Lecture Notes in Artificial Intelligence 2182

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Matthias Klusch   Franco Zambonelli (Eds.)

Cooperative Information Agents V

5th International Workshop, CIA 2001
Modena, Italy, September 6-8, 2001
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saabrücken, Germany

Volume Editors

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

Franco Zambonelli
Università di Modena e Reggio Emilia, Dipartimento di Scienze dell'Ingegneria
Via Vignolese 905, 41100 Modena, Italy
E-mail: franco.zambonelli@unimo.it

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Cooperative information agents V : 5th international workshop ; proceedings / CIA 2001, Modena, Italy, September 6 - 8, 2001. Matthias Klusch ; Franco Zambonelli (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2182 : Lecture notes in artificial intelligence)
ISBN 3-540-42545-4

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

ISBN 3-540-42545-4 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Typesetting: Camera-ready by author
Printed on acid-free paper SPIN: 10840517 06/3142 5 4 3 2 1 0

Preface

These are the proceedings of the Fifth International Workshop on Cooperative Information Agents, held in Modena, Italy, September 6-8, 2001.

Information agent technology has become one of the major key technologies for the Internet and the World Wide Web. It mainly emerged as a response to the challenges of cyberspace from both the technological and human user perspective. Development of information agents requires expertise from different research disciplines such as Artificial Intelligence (AI), advanced databases and knowledge base systems, distributed information systems, information retrieval, and Human Computer Interaction (HCI). The fifth international workshop on Cooperative Information Agents (CIA) continued the tradition by capturing the intrinsic interdisciplinary nature of the above research area by calling for contributions from different research communities, and by promoting open and informative discussions on all related topics.

In keeping with tradition, the workshop featured a sequence of regular and invited talks of excellence given by leading experts in the field. This year the topics of the talks are mainly on the challenges of information agents in the upcoming age of ubiquitous and pervasive computing. These challenges are in particular due to the necessity of an efficient utilization, evolution, and trust management of information agents for user-oriented information search, provision, and visualization in networked computing environments with small, mobile, and embedded devices. A different issue concerns the potential of agent-based support of massive distributed data warehousing worldwide.

As a novelty, the CIA 2001 workshop issued two awards for best paper and best system innovation to acknowledge particularly significant advances in research and development respectively, in the area of information agents. Regarding the *CIA System Innovation Award* each nominee, as signaled by the program committee during the review process, was requested to demonstrate a running prototype at the workshop to the jury of the award. Moreover, each paper submitted to CIA 2001 was considered for nomination for the *CIA Best Paper Award* by the program committee. Both awards were given at the end of the workshop.

Based on the success of former workshops and the fact that the workshop is now recognized as a well-established event in the area of agent-based computing, this year's organizers decided to hold the CIA workshop as a stand-alone event for the second time in its history. In past years, the event was mainly co-located with other major conferences in the domains of agent technology (Autonomous Agents 1998, ICMAS 2000), and Artificial Intelligence (IJCAI 1999). The CIA 2001 event was held in collaboration with the 28th conference on Very Large Databases (VLDB) to continue providing a bridge between the areas of research in databases and intelligent agents.

CIA 2001 featured 5 invited, 13 regular, and 12 short papers selected from 63 submissions. This represented an 15% increase in the number of submissions compared with the last year's workshop, which confirms the success of the CIA workshop and its force of attraction, independent of any co-location. The result of the peer-review of all contributions is included in this volume, rich in interesting, inspiring, and advanced work in 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), and 1860 (2000), respectively.

Acknowledgements. First of all, we are very much indebted to our sponsors whose financial support made this event possible and contributed to its success, from both the scientific point of view (invited speakers and CIA Awards) and that of the local organization (social program and printed material). The sponsors of the CIA 2001 were the following.

- NASA Goddard Space Flight Center, USA
- NOKIA Research Center, USA
- Swiss Life AG, Switzerland
- AgentLink II, European Network of Excellence for Agent-Based Computing
- AutoDesk Inc., USA
- OTConsulting, Italy
- IFMAS International Foundation for Multi-Agent Systems.

Secondly, we are grateful to the authors and invited speakers for contributing to this workshop as well as to all members of the program committee and external reviewers for their very careful, critical, and thoughtful reviews of all submissions.

Finally, our special thanks goes to the members of the local organization committee in Modena – Giacomo Cabri, Letizia Leonardi, and Maurizio Vincini – for their hard work in providing CIA 2001 with a remarkably comfortable location at the brand new Faculty of Engineering of the University of Modena and Reggio Emilia, and an exciting social program.

September 2001 Matthias Klusch and Franco Zambonelli

Program Committee

Ricardo Baeza-Yates	(University of Chile, Chile)
Sonia Bergamaschi	(University of Modena and Reggio Emilia, Italy)
Hans-Dieter Burkhard	(Humboldt University of Berlin, Germany)
Cristiano Castelfranchi	(NRC Rome, Italy)
Brahim Chaib-draa	(Laval University, Canada)
Yves Demazeau	(Leibniz CNRS, France)
Rose Dieng	(INRIA, France)
Frank Dignum	(University of Utrecht, The Netherlands)
Peter Edwards	(University of Aberdeen, UK)
Dieter Fensel	(Free University of Amsterdam, The Netherlands)
Barabara Hayes-Roth	(University of Stanford, USA)
Mike Huhns	(University of South Carolina, USA)
Yasuhiko Kitamura	(Osaka City University, Japan)
Sarit Kraus	(University of Maryland, USA)
Victor Lesser	(University of Massachusetts, USA)
Robert A. Meersman	(University of Brussels, Belgium)
Werner Nutt	(Heriot-Watt University Edinburgh, UK)
Andrea Omicini	(University of Bologna, Italy)
Aris Ouksel	(University of Illinois at Chicago, USA)
Ana Paiva	(IST - TU Lisbon, Portugal)
Todd Papaioannou	(DALi Inc., USA)
Rosalind W. Picard	(MIT Media Lab, USA)
Agostino Poggi	(University of Parma, Italy)
Ulrich Reimer	(Swiss Life AG, Switzerland)
Ulrike Sattler	(RWTH Aachen, Germany)
Heiko Schuldt	(ETH Zurich, Switzerland)
Sandip Sen	(University of Tulsa, USA)
Amit Sheth	(University of Georgia, USA)
Carles Sierra	(CSIC AI Research Lab, Catalonia, Spain)
Munindar Singh	(North Carolina State University, USA)
Von-Wun Soo	(National Tsing Hua University, Taiwan)
Leon Sterling	(University of Melbourne, Australia)
Katia Sycara	(Carnegie Mellon University, USA)
Robert Tolksdorf	(TU Berlin, Germany)
Robert Trappl	(Austrian Research Institute for AI, Austria)
Jan Treur	(Free University of Amsterdam, The Netherlands)
Christian Tschudin	(University of Uppsala, Sweden)
Gerhard Weiss	(TU Munich, Germany)
Mike Wooldridge	(University of Liverpool, UK)
Makoto Yokoo	(NTT Communication Science Lab, Japan)
Ning Zhong	(Maebashi Institute of Technology, Japan)

Co-chairs

Matthias Klusch (DFKI, Germany), General Chair
Franco Zambonelli (University of Modena and Reggio Emilia, Italy)
Larry Kerschberg (George Mason University, USA)
Toru Ishida (Kyoto University, Japan)
Onn Shehory (IBM Research, Israel)

Local Organizing Committee

Franco Zambonelli (University of Modena and Reggio Emilia, Italy)
Giacomo Cabri (University of Modena and Reggio Emilia, Italy)
Letizia Leonardi (University of Modena and Reggio Emilia, Italy)
Maurizio Vincini (University of Modena and Reggio Emilia, Italy)

External Reviewers

Kemafor Anyanwu
Felix Brandt
Pere Garcia Calvés
Ying Ding
Partha Sarathi Dutta
Joris Hulstijn
Andreas Gerber
Claudia V. Goldmann
Claire Green
Markus Hannebauer
Peter Kropf
Gabriela Lindemann-v.Trzebiatowski
Federica Mandreoli
Jordi Sabater Mir
Masayuki Okamoto
Borys Omelayenko
Giovanni Rimassa
Matteo Somacher
Arnon Sturm
Leon van der Torre
Maurizio Vincini
Niek Wijngaards

Table of Contents

Personal Assistance: Interaction and Avatars

Information Search and Recommendation

Data Warehousing and Mining

Collaborative Information Agents: Systems and Applications

Trading Internet Agents: Auctions

Trading Internet Agents: Strategies, Negotiation, and Design

Issues of Collaboration and Coordination

Information Agents for Mobile and Wireless Environments: Practical Issues and Directions

Invited Contribution

Interactive Integration of Information Agents on the Web

Yasuhiko Kitamura[1], Teruhiro Yamada[2*], Takashi Kokubo[3**], Yasuhiro Mawarimichi[1***], Taizo Yamamoto[2†], and Toru Ishida[3]

[1] Osaka City University, Osaka 558-8585, Japan
kitamura@kdel.info.eng.osaka-cu.ac.jp
http://www.kdel.info.eng.osaka-cu.ac.jp/~kitamura/
[2] Laboratories of Image Information Science and Technology, Japan
[3] Kyoto University, Kyoto 606-8501, Japan

Abstract. World Wide Web contains a vast amount of different information stored in a huge number of distributed Web sites. Search engines and information agents have been developed to facilitate efficient information retrieval tasks from the Web. By integrating multiple search engines and information agents as an interoperable system, we increase the value of each of them. In conventional collaborative systems, the integration process is designed by system designers and is concealed from the end users.

This paper proposes an interactive multiagent-based interface called Multiple Character-agent Interface (MCI) where animated character-agents interact with each other and with the user for assisting in information retrieval. By using the MCI, even a novice user can create a team of information agents and can self-customize the agents through the interactions with them. We here report the architecture of MCI and two prototype systems based on MCI, Venus and Mars, which is a cooperative multiagent system for information retrieval, and Recommendation Battlers, which is a competitive multiagent system for information recommendation.

1 Introduction

World Wide Web (WWW) provide a means for disseminating and sharing information on the Internet and has been widely used for doing business, education, research, advertisement and so on. The amount of information stored on the Web is increasing day by day, but the more the information is stored, the more difficult to find. Search engines are the most popular tools to find Web pages. A search engine returns a list of URLs responding to the query keyword(s) submitted by the user, but it often returns too many URLs, which include a fair number of unrelated ones, to be processed by a human user.

[*] Presently with SANYO Electric, Tokyo 113-8434, Japan.
[**] Presently with NTT Docomo, Kanagawa 239-8536, Japan.
[***] Presently with NTT Advanced Technology, Kanagaww 210-0007, Japan.
[†] Presently with NTT West, Osaka 530-6691, Japan.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 1–13, 2001.

A number of information agents have been developed to replace or reinforce search engines [11,12]. For example, Letizia [14], WebWatcher [10], and WebMate [5] learn the preference or interest of user by monitoring the history of Web browsing or searching performed by the user, and recommend suitable Web pages for the user or refine search keywords. Ahoy! [19] analyzes and filters the output of general-purpose search engine and returns a domain-specific output such as individual's homepages. Fab [2] is an information recommendation system that incorporates the collaborative filtering method.

Combining or integrating search engines and/or information agents adds more value to each system. Meta-search engines such as MetaCrawler [18] and SavvySearch [8] integrate the output of multiple search engines and succeed to improve the performance. BIG [13] is an information agent that intelligently and efficiently gathers information from multiple sources considering the trade-off between the quality of information and the constraints like time and cost. RETSINA [20] consists of three types of reusable agents; interface agents, task agents, and resource agents. An interface agent interacts with the user to receive a query from the user and returns results. A task agent solves domain-specific tasks through exchanging information with other agents. A resource agent provides access to a heterogeneous information source. Other multi-agent based systems, such as federated system [7], InfoSleuth [3], and LARKS[12], incorporate information brokering or information matchmaking mechanism.

In conventional collaborative systems such as mentioned above, the way to coordinate information agents or information resources is specified by the system designers and concealed from the users. Hence, the users are just allowed to submit a query to a fixed interface and to receive results from the system, but not allowed to change the combination of information agents nor the collaboration mechanism.

For example, RETSINA agents are reusable and their interface is open to the system designers but not to the user, so the user can just get access to the system only through the interface agent. In the federated system, the process of coordination among agents is specified in the ACL (Agent Communication Language) such as KQML [6]. The ACL provides an open interface to information agents but not to human users because it is difficult for a human user to communicate directly with agents using the ACL. Hence, neither system is designed to provide an open interface to the end user.

Each user has different demands or preferences for information retrieval. Some user may like some search engine and others may not. Hence, rather than just a ready-made collaborative system, we would like to have a framework where we can easily make a team of favorite information agents that work together and customize them flexibly.

This paper proposes the Multiple Character-agent Interface (MCI) as shown in Fig. 1 where multiple character agents, each of which represents an information agent that resides at a server, collaborate with each other on a client machine. We can view a character agent as the head of information agent whose body resides in a server. The collaborative actions for retrieving information are performed by character agents on the client machine and displayed to the user. The user can get direct access to the agents by clicking and talking, so he/she can submit the request to each agent and can customize the agent's actions. The user can also make a favorite team of agents by calling them out to the client machine.

The advantages of MCI are as follows. The collaboration process among information agents is open to the user. The user can understand how the process goes and what happens in the system including some erroneous or inappropriate actions caused by an agent. In a conventional collaborative system, the user may not be able to notice such erroneous actions because the collaboration process is concealed from the user. In the MCI, the user can not only notice the error, but also fix it by tuning the corresponding agent or by replacing it with other appropriate agent. Hence, the user can customize the team of information agents through visible interactions with them.

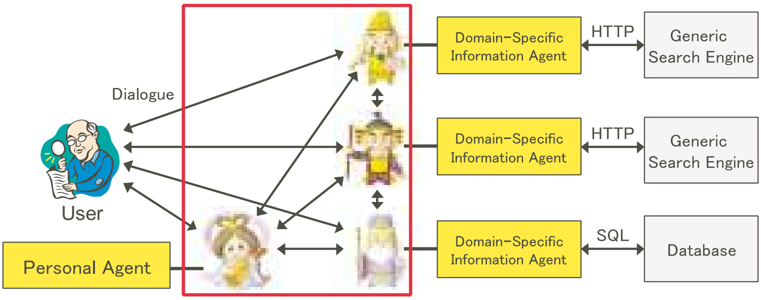

Dialogue
User
Personal Agent
Domain-Specific Information Agent
Domain-Specific Information Agent
Domain-Specific Information Agent
HTTP
HTTP
SQL
Generic Search Engine
Generic Search Engine
Database

Fig. 1. Overview of Multiple Character-agent Interface.

Andre and Rist propose a similar system employing multiple character-agents [1], but their work mainly emphasize the advantage of multiple characters as a presentation media. Agents in their system reside and perform in a client machine whereas, in our system, agent's bodies reside independent servers in a distributed manner and agent's heads interact in a client machine.

Section 2 addresses the architecture and the implementation issues of the Multiple Character-agent Interface. Then, two prototypes that use MCI are introduced. Section 3 describes Venus and Mars which is a cooperative multi-agent system consisting domain-specific information agents and a personal agent. In Section 4, we introduce Recommendation Battlers, a competitive multi-agent system in which two character-agents competitively recommend restaurant information to the user.

2 Multiple Character-agents Interface

We show the system architecture of Multiple Character-agent Interface in Fig. 2. MCI consists of multiple character-agents; the body of each agent resides in a different server. As an autonomous agent mentioned in [16], each agent recognizes actions taken by the user or other agents through data from sensors, interprets the actions, and responds through its actuator. Agent behavior is controlled by a program written in Q [9], which is an interaction design language being developed by Kyoto University for linking autonomous agents and humans.

Table 2 shows major sensor and actuator commands of a typical MCI agent. Other optional commands are available according to the functionality of agent.

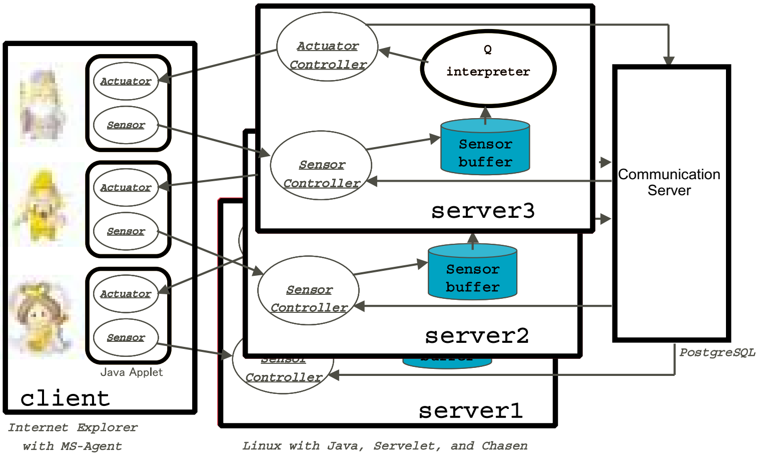


Fig. 2. Architecture of Multiple Character-agents Interface.

Here is an example of the behavior of an MCI agent as specified in Q.

```
((?feel)
   (!speak "Hello.")
   (!display_dialogue_box "prompt" :title "May I help you?"))
```

This rule specifies that the agent speaks “Hello.” and shows a dialogue box with “May I help you?” when it is clicked.

```
((?find "time" :hour $hour)
   (if (and (>= hour 18) (< hour 6))
      (!speak "Good Night!"))
   (if (and (>= hour 6) (< hour 12))
      (!speak "Good Morning!"))
   (if (and (>= hour 12) (< hour 18))
      (!speak "Good Afternoon!")))
```

Sensor Commands:	
`(?feel)`	The agent checks whether it is clicked.
`(?hear $utterance [:from agent])`	The agent hears an utterance from other agents or its user.
`(?find $database_name :parameter parameter)`	The agent get information from the specified database.
Action Commands:	
`(!show agent)`	The agent appears.
`(!hide agent)`	The agent disappears.
`(!speak utterance [:to agent])`	The agent utters `utterance` to other agent.
`(!play_animation action)`	The agent performs the animated `action`.
`(!point direction)`	The agent points to the specified `direction`.
`(!display_dialogue_box type [:title title])`	The agent shows an dialogue box.
`(!present url)`	The agent shows a Web page at `url` on its browser.
`(!search keyword)`	The agent submits `keyword` to a search engine.

Table 1. Major sensor and actuator commands of MCI agent.

This rule specifies as the agent checks the time and then speaks the greeting words, "Good Morning!," "Good Afternoon!," or "Good Night!" according to the time.

To make an agent perform a complex behavior, rules are grouped in scenes as follows.

```
(scene1                          // Scene 1
  ((otherwise)
    (!show)                      // The agent appears
    (go scene2)))                // and goes to Scene 2.
(scene2                          // Scene 2
  ((?hear "Hello!")              // If the agent hears "Hello,"
    (!speak "Hello!"))           // he says "Hello."
  ((?feel)                       // If the agent is clicked,
    (!speak "I am itchy.")       // he says "I am itchy."
    (go scene3)))                // and goes to Scene 3.
(scene3                          // Scene 3
  ((?feel)                       // If the agent is clicked,
    (!speak "Please stop it.")   // he says "Please stop it."
    (go scene4)))                // and goes to Scene 4.
```

At the initial scene (Scene 1) the agent appears and moves to Scene 2. At Scene 2, he says "Hello." if he hears "Hello." If he is clicked, he says "I am itchy." and moves to Scene 3. At Scene 3, as the action is different from the one at Scene 2, he says "Please stop it." if he is clicked.

Information agents, which reside in distributed servers, are integrated through the integration of their characters (heads) on a client machine. Once an agent is called to the client machine, it begins to interact with the user and other agents in a plug-and-play fashion.

Character-agents are implemented on the MS-Agent platform and controlled through Java Applets and Java Scripts running within Internet Explorer. By using multiple frames as shown in Fig. 3, we can realize multiple characters that interact with each other and the user. The MCI consists of a control frame and multiple agent frames. When the system is initiated, agent manager, user manager, and dialogue manager are downloaded in the control frame. Agent manager manages agent frames when agents are invoked or stopped. User manager manages users by using the Cookie mechanism. Dialogue manager controls the order of utterances in order not to make multiple agents talk at the same time. For a request of showing an agent, the agent manager downloads character controller, command receiver and transmitter from the agent server in an agent frame. Character controller is written in Java Script and controls a character agent. This corresponds to sensor and actuator modules in Fig. 2. Command receiver and transmitter are Java Applets connecting the character (head) and the server (body).

The user's actions (clicks or utterances) to an agent are sensed through its sensor applet (character controller) and forwarded to the sensor controller in the agent's server through the command transmitter. Actions taken by other agents are sensed through the communication server and the command receiver. Action commands from the agent's server is forwarded to the character through the command receiver and the character controller.

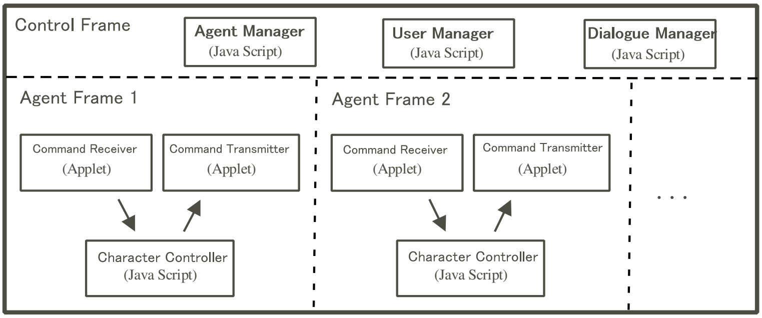

Control Frame
Agent Manager (Java Script)
User Manager (Java Script)
Dialogue Manager (Java Script)
Agent Frame 1
Command Receiver (Applet)
Command Transmitter (Applet)
Character Controller (Java Script)
Agent Frame 2
Command Receiver (Applet)
Command Transmitter (Applet)
Character Controller (Java Script)
. . .

Fig. 3. Implementation of Multiple Character-agents Interface.

By adopting the above multiagent architecture, we can easily add or remove agents on the client machine. This allows a wide a variety of collaborative tasks for information retrieval to be achieved by making various teams of agents. Moreover, the failure of one agent does not affect the other agents seriously, so the performance of the whole team degrades only locally. While the agents should be autonomous to a large extent, some actions should be tightly controlled. To this end we implemented the dialogue manager that prevents them from speaking at the same time.

3 Venus and Mars: A Cooperative Multiagent System for Information Search

Search engines are the most widely used tools for retrieving information from the Web. A drawback of conventional search engine is that it tends to return too many URLs with low precision as the amount of information increases on the Web. One approach to deal with this drawback is to build domain-specific search engines that can deal with some specific topics with high precision [19]. Furthermore, a platform where such domain-specific search engines or agents collaborate with each other looks promising.

Venus and Mars (VandM) [22] is an information retrieval system in which multiple character-agents work cooperatively to assist the user. As shown in Fig. 4, VandM consists of three types of character-agents. Kon-san is the information agent that locates Web pages about cooking recipe. When the user submits a Japanese query that include keywords about recipe, such as “I would like to eat a pork dish,” Kon-san extracts one or more keywords about recipe (“pork”) and submit the keyword(s), with keyword spices[1] for the recipe domain, to a general-purpose search engine[2]. It then receives search results from the search engine and shows them to the user through the Web browser. In case it receives too many results, it automatically asks for additional keywords to reduce the number of results.

Cho-san has knowledge about cooking ingredients and health. Responding to an utterance that includes keywords related to cooking ingredients or health, it utters comments about the relations between cooking ingredients and health, such as “Leeks are good for colds.”

Pekko is the personal agent; it initially appears on the client machine and calls other agents. It chats with the user, and monitors the user’s search history. When needed, it suggests some search keywords to Kon-san on behalf of its user referring to the user’s history.

A snapshot of Venus and Mars is shown in Fig. 5. A typical cooking recipe search is given below.

(1)Pekko: “May I help you?”
(2)User: “I would like to eat a pork dish.”
(3)Kon-san “OK. Let me locate some recipes that use pork. How do you like this?” (He shows a recipe page on the browser.)

[1] Keyword spices are domain-specific keywords to improve the performance of general-purpose search engine. [15] discusses how to discover keyword spice semi-automatically by using a machine learning technique. For example, tablespoon $\vee$ ($\neg$tablespoon $\wedge$ recipe $\wedge$ $\neg$home $\wedge$ $\neg$top) $\vee$ ($\neg$tablespoon $\wedge$ $\neg$recipe $\wedge$ pepper $\wedge$ $\neg$pan) is discovered for the recipe domain.

[2] http://www.goo.ne.jp/

Nickname	Pekko	Kon-san	Cho-san
Type	Personal Agent	Information Agent	Information Agent
Function	Chatting with the user. Learning the user's preference.	Searching WWW pages concerning cooking recipes.	Providing comments about combination of cooking ingredients and health.
Knowledge	User's profile	Keywords about recipes, ingredients, and seasoning.	Database about cooking ingredients and health.

Fig. 4. Character-agents in Venus&Mars.

(4)Cho-san "Pork is good for the digestion. It is a good way to get vitamin B."
(5)Kon-san "The number of search results is over 1000. Let me reduce the number. What kind of seasoning do you like?"
(6)Pekko: "I know Kitamura-san likes Japanese food."
(7)Kon-san: "OK. Let me find some Japanese recipes that use pork. How do you like this?" (He shows another recipe page on the browser.)
(8)Pekko: (Responding to a click from the user) "May I help you?"
(9)User: "I want a recipe that is good for recovering from a cold."
(10)Kon-san: "?"
(11)Cho-san: "Leeks are good for colds."
(12)Kon-san: "OK. Let me locate recipes that use leeks. How do you like this?" (He shows a recipe page that mentions leeks on the browser.)

In VandM, agents collaborate with each other in two ways. In the above dialogue steps, (5) to (7), Pekko assists Kon-san in reducing the number of search results. In utterance (5), Kon-san asks for a tip on seasoning, Pekko answers "I know Kitamura-san likes Japanese food." on behalf of the user by referring to his interaction history. Of course, if the user does not like Pekko's suggestion, he/she can correct Pekko's utterance by indicating his true preference directly to Kon-san through the dialogue box. Pekko recognizes this correction and updates the user's preferences stored in its database.

In the above dialogue steps, (9) to (12), Cho-san assists Kon-san. In this case, Kon-san cannot answer the request "I want a recipe that is good for recovering from a cold." because it has no knowledge about health. On the other hand, Cho-san has knowledge about cooking ingredients and health, so it makes the comment "Leeks are good for colds." Kon-san takes the comment as a clue to initiate a new search with the keyword of leek. This type of collaboration shows a potential of VandM for realizing various type of information search by adding agents to the team. For example, if we add a restaurant recommendation agent to the team, the user's request for a recipe with salmon may also result in a recommendation of local restaurants specializing in salmon.

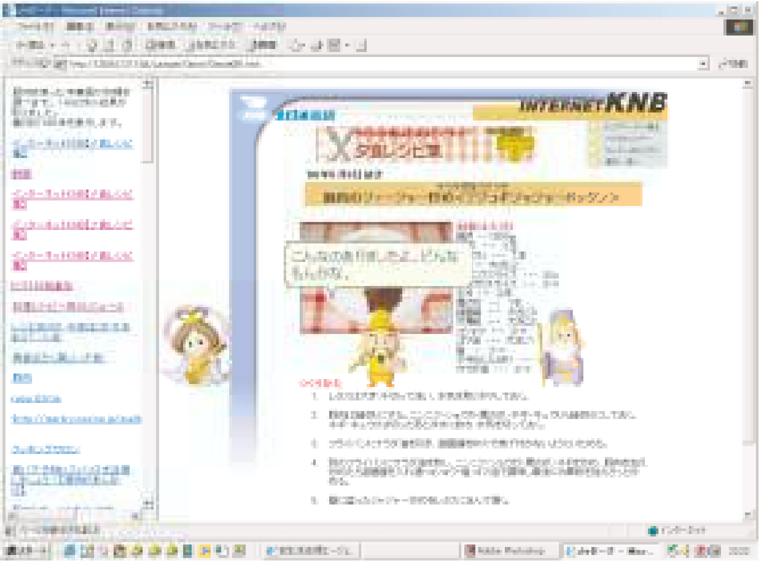


Fig. 5. A snapshot of Venus&Mars.

4 Recommendation Battlers: A Competitive Multiagent System for Information Recommendation

With the growth of e-commerce market, a large number of shopping sites are open on the Internet. Recently comparison shopping sites, such as www.dealtime.com, are also available. Such a site collects information from multiple shopping sites and provides a list of commodities sorted in order of the price. Though it is a very convenient tool to find the cheapest price, it does not provide direct interactions between buyers and sellers. We here provides a framework where an information user (buyer) and providers (sellers) directly interact with each other through information agents. This scheme gives a new style of e-commerce where multiple sales agents directly negotiate with the customer.

Recommendation Battlers(RB)[17] is a multiagent system for information recommendation in which two character-agents competitively recommend information. It consists of two restaurant recommendation agents; Peedy and Genie. Responding to a request from the user, Peedy collects relevant restaurant pages from Yahoo! Gourmet[3] and extracts their addresses and average cost. It then calculate the distance from the nearest station by referring to the addresses. Genie also do the same process but it uses Kansai Walker[4]. The user and agents interact on a client machine as shown in Fig. 6. A typical dialogue is given below.

(1)User: "I would like to find a good noodle restaurant near Shinsaibashi station."
(2)Genie: (After collecting information from the Kansai Walker) "Do you like Restaurant A? The average cost is around 2,000 Yen and the distance from Shinsaibashi station is 90m."
(3)User: "I like it."
(4)Peedy: (After collecting information from the Yahoo! Gourmet) "Do you like Restaurant B? The average cost is around 2,500 Yen and the distance from Shinsaibashi station is 80m."
(5)User: "No, I don't like it."
(6)Peedy: "Do you like Restaurant C? The average cost is around 2,200 Yen and the distance from Shinsaibashi station is 80m."
(7)User: "Good."
(8)Genie: "I sure that you like Restaurant D. The average cost is around 1500 Yen and the distance from Shinsaibashi station is 70m."
(9)Peedy: "I have no more restaurant to recommend."

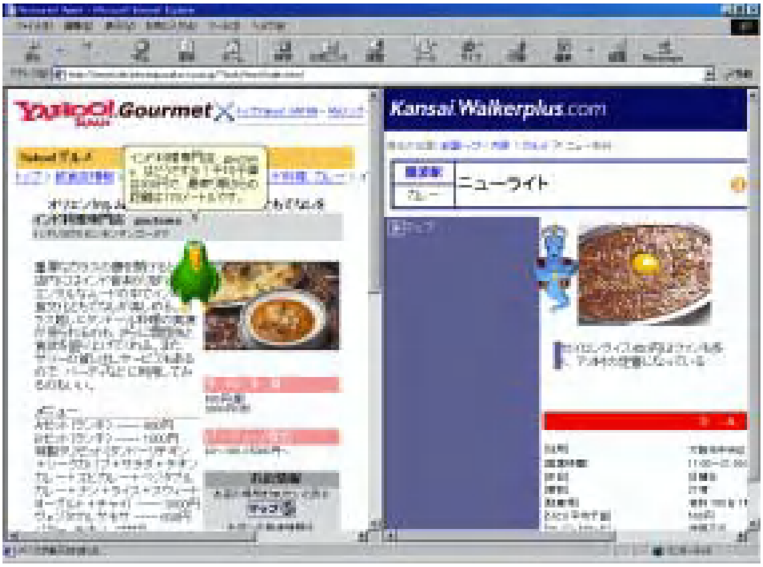


Fig. 6. A snapshot of Recommendation Battlers

[3] http://gourmet.yahoo.co.jp/
[4] http://www.walkerplus.com/kansai/gourmet/

In contrast to VandM, where agents cooperate, agents in RB compete with each other. Each monitors the other agent's proposals and the user's responses. It then proposes a new item that is more appropriate given the course of dialogue estimating the preference of user by using a rational proposal mechanism [17].

For example, in utterance (2), Genie proposes Restaurant A with the average cost of 2,000 yen and the distance of 90m. After the user accepts the proposal, the opponent Peedy proposes Restaurant B which looks better from his perspective because the distance is nearer than that of Restaurant A. However, the user does not accept the proposal, so Peedy proposes another Restaurant C. Through interactions with the user, Genie and Peedy continue to propose restaurants estimating the preference of user until one of them has nothing to propose.

5 Future Study and Conclusion

In this paper, we proposed the MCI where the user performs information retrieval tasks interacting with multiple information agents, and introduced two prototypes, Venus and Mars and Recommendation Battlers, using the MCI. Collaborative tasks performed in these prototypes are rather simple and a lot of future work still remains to build more advanced collaborative information agents.

Capability for collaboration The MCI agents start to collaborate with each other once they are called on the client machine in a plug-and-play fashion. Collaborative actions performed by agents in our current prototypes are simple such as association or extension of search keywords. To realize more advanced collaborative features, we need to incorporate collaborative mechanisms using coordination and negotiation techniques, which have been studied in the field of multi-agent systems [21], into our systems.

Capability for presentation In our current prototypes, collaboration among agents is presented as a conversational activity. Conversation among agents performed on a computer display is volatile and it may not be suitable for presenting a complex collaborative activity. Hence, we need to improve the presentation skill, for example, by creating a virtual space where agents can interact with not only other agents but also virtual objects. Such skills have much concern with the work being done by Andre and her colleagues [1].

Capability for interaction In our current prototypes, agents interact with the user mainly by using natural language, but just use a simple technique such as keyword extraction from an utterance in Japanese. Natural language is the most natural way of communication for human users, especially for novice users such as children and old people. For more complex interactions with the user, more advanced features of natural language processing are needed to be applied to our agents.

Current Internet-based information systems depend heavily on the Web technology. Since we can put not only text, but also image and audio, on a Web page, the Web is highly expressive. The XML technology will further enhance the value of current Web system. On the other hand, the Web information system looks static because it just provide information in a page by page way. Character agents enhance the Web technology in another way and make it look more dynamic and interactive. To this end, a number of works have been done in the academic field [4] and some are commercially used such as Extempo [5], Haptek [6], Virtual Personalities [7], Artificial Life [8] and so on. The MCI provides a framework where multiple character agents are integrated as a collaborative system. This scheme may lead to a new generation of agent-based information integration systems after the XML age.

Acknowledgement

This paper reports the progress of a joint project of Kyoto University, Osaka City University, SANYO Electric, NTT West, and NTT Comware at LIST (Laboratories of Image Information Science and Technology) subsidized by NEDO (New Energy and Industrial Technology Development Organization). I thank Masahiko Kyosaka, Kiyomasa Okumura, Yuji Tomono, Hiroshi Morimoto, Mutsumi Ikeda, and Toshiki Sakamoto for their contribution to the development of the system.

References

1. Andre, E., Rist, T.: Adding Life-Like Synthetic Characters to the Web. Cooperative Information Agents IV, Lecture Notes in Artificial Intelligence 1860. Springer (2000) 1–13
2. Balabanovic, M., Shoham, Y.: Fab: Content-Based, Collaborative Recommendation. Communications of the ACM, **40**(3) (1997) 66–72
3. Bayardo, R.J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewics, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D.: InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. In Proceedings ACM SIGMOD International Conference on Management of Data (1997) 195–206
4. Cassell, J., Sullivan, J., Prevost, S., Churchill, E. (eds.): Embodied Conversational Agents. The MIT Press (2000)
5. Chen, L., Sycara, K.: Web Mate: A Personal Agent for Browsing and Searching. In Proceedings of the Second International Conference on Autonomous Agents (1998) 132–139
6. Finin, T., Labrou, Y., Mayfield, J.: KQML as an Agent Communication Language. In Software Agents, AAAI Press (1997) 291–316
7. Genesereth M.R.: An Agent-Based Framework for Interoperability. In Software Agents, AAAI Press (1997) 317-345

[5] http://www.extempo.com

[6] http://www.haptek.com

[7] http://www.vperson.com

[8] http://www.artificial-life.com

8. Howe, A.E., Dreilinger, D.: Savvy Search: A Metasearch Engine That Learns Which Search Engines to Query. AI Magazine, **18**(2) (1997) 19–25
9. Ishida, T.: Interaction Design Language Q: The Initial Proposal. In Proceedings 15th Annual Conference of Japanese Society for Artificial Intelligence, 2A2-03 (2001)
10. Joachims, T., Freitag, D., Mitchell, T.: WebWatcher: A Tour Guide for the World Wide Web. In Proceeding of the 15th Joint Conference on Artificial Intelligence (1997) 770–775
11. Klusch, M.: Intelligent Information Agents, Springer (1999)
12. Klusch, M.: Information Agent Technology for the Internet: A Survey. Journal on Data and Knowledge Engineering, **36**(3) (2001)
13. Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T., Zhang, S.X.: BIG: An agent for resource-bounded information gathering and decision making. Artificial Intelligence, **118**(1-2) (2000) 197–244
14. Lieberman, H.: Letizia: An Agent That Assists Web Browsing. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (1995) 924–929
15. Oyama, S., Kokubo, T., Yamada, T., Kitamura, Y., Ishida, T.: Keyword Spices: A New Method for Building Domain-Specific Web Search Engines. In Proceedings 17th International Joint Conference on Artificial Intelligence (2001) (in press)
16. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall(1995)
17. Sakamoto, T., Mawarimichi, Y., Kitamura, Y., Tatsumi, S.: Competitive Information Recommendation System Using Multi-Characters Web Information. In Proceedings 15th Annual Conference of Japanese Society for Artificial Intelligence, 1F1-01 (2001)
18. Selberg, E., Etzioni, O.: Multi-Service Search and Comparison Using the MetaCrawler. In Proceedings of the 4th International World Wide Web Conference (1995) 195–208
19. Shakes, J., Langheinrich, M., Etzioni, O.: Dynamic Reference Sifting: A Case Study in the Homepage Domain. In Proceedings of Sixth International World Wide Web Conference (1997)
20. Sycara, K., Zeng, D.: Coordination of Multiple Intelligent Software Agents. International Journal of Cooperative Information Systems **5**(2&3) (1996) 181–211
21. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press (1999).
22. Yamada, T., Kokubo, T., Kitamura, Y.: Web Information Integration Using Multiple Character Interface. In Proceedings 15th Annual Conference of Japanese Society for Artificial Intelligence, 1F1-05 (2001)

Improving Communication in 3D Virtual Environments by Means of Task Delegation in Agents

Ricardo Imbert, Angélica de Antonio, and Javier Segovia

Facultad de Informática, Universidad Politécnica de Madrid
Campus de Montegancedo, s/n – 28660 Boadilla del Monte, Madrid, Spain
{rimbert, angelica, fsegovia}@fi.upm.es

Abstract. User-to-User Communication within Internet-based 3D Virtual Environments is usually difficult mainly due to the trade-off between the capability and the complexity of the interface. In this paper we make a proposal for interface management based on the delegation of some communication tasks on Intelligent and Emotional Agents.

1 Believability, Usability, and Delegation

We can probably place the continuous and successful growth of the Internet on the easiness it provides to share information, and the enhancement it supposes in human communication. On this respect, communication over the Internet has become more and more sophisticated as the technology has evolved. A text-based chat line has been good enough for a long time for synchronous communication, given that technology only allowed fast text interchange.

Obviously, a picture is worth a thousand words, and the increase in the line bandwidth allows complementing the written message with 2D and even 3D content. The natural evolution of this 3D content has led to Virtual Environments (VE). In these VE, users are provided with a virtual representation for their interaction, a character commonly called *avatar*, which reinforces the plain textual message.

In this point a new problem arises: with these new media, users are forced to manage a higher degree of complexity in their interaction. A user of a VE has to decide on the reactions, facial expression and body movements (i.e., complex behaviors) of their virtual representation, all while they keep on writing or talking. The result is that real-time complete – believable – communication gets very demanding, and the user gives up managing their avatar and turns back to the plain message. In short, we have only reached a nicer chat line.

The **Amusement Project**[1] has sought for VEs in which the users can easily perform complex, interesting tasks, cooperating and interacting with other users, in a believable way. It copes with the problem of providing believability in the interaction but minimizing any negative effect on the VE usability by integrating personalized agents (both autonomous and semi-autonomous) and avatars within the environment.

[1] Amusement Project has been partially funded by the European Union, Esprit Project 25197

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 14–19, 2001.

The approach of providing believability to the characters behavior by attaching them an intelligent agent has been tried before, but mostly centered in strictly autonomous characters. It is reasonably manageable to provide an autonomous character with realistic/believable behaviors; the problem arises when the character's cannot just show a set of pre-recorded behaviors in a given sequence, but its behavior depends on every moment on several factors, such as the user's intentions, the rest of the avatars behavior or some past events.

One of the most promising approaches to achieve believability, mostly explored for autonomous characters, is to provide the avatar with a personality and emotions [10] [11], what implies that it should have a – more or less complex – emotional model. This model is essential if the avatar is to have some autonomy, in order to react properly, to deliberate and reason, and then to show proper behaviors.

The approach presented in this paper goes a step further: we want avatar-agents in which we can delegate some functions (the less interesting or the most tiring ones from the users' point of view), but avatar-agents that will be managed by a real user in real time (semi-autonomy vs. pure bots or pure puppets). Our approach is to provide the pair avatar-agent with an Internal Model to allow it to generate autonomously the same behaviors that the user expected, letting the user free to interact.

2 The Virtual Environment and Its Inhabitants

It is commonly accepted that fancy graphics may be less important than how meaningful the interactions in which the user engages are [8]. However, a poor interface usually gives the wrong impression of a poor interaction. There is the balance to be reached between these two factors.

2.1 The Avatars

We can think of our representation in the VE as a duality mind-body, where the mind is the part related to the intelligent, personalized agent, and the body has to do with the avatar itself. In this section we will deal with the body, and in a following one we will deal with the mind.

The first important decision to be made is regarding the avatar's appearance: is it better to use human-like or non-human-like avatars? The big problem with human-like avatars is that users usually expect from them not only believable behaviors, but also realistic behaviors, what implies a certain risk. We have found that it is possible to make non-verbal communication (NVC) simpler without losing expressive power and believability by using of a more simplified and well-structured image than a human avatar. This avatar might even be more believable, and could be perceived more clearly and faster [1]. This idea and its potential have also been noticed in some experiments on NVC using very realistic human-like avatars [5].

Our choice was a non-human appearance for the avatars, more exactly a very schematic rabbit, consisting simply of a ball with two ears, two stationary eyes and two legs and big feet. The reasons: the obvious limitations in the speed and capabilities of the PCs employed by users to link up to the Internet made it necessary to use simplified forms for both characters and objects; schematisation makes it easier to convey and understand the properties of the avatars [2][3][1]; the choice of a well-known animal for the characters adds the characteristics of the original form to the avatar; the schematisation of avatars imposes a solid framework upon the objects around them, making it possible to free oneself to a certain extent from the constraints of the natural laws necessary in a more realistic design [2]; and finally, a schematised form can convey moods and personality much more easily.

2.2 The Environment

The appearance of the world must be consistent with the form of the avatars dwelling it. It must also be the place or space where events happen, and where the objectives of users are satisfied. In our case, the environment looks like a world of wild rabbits where the terrain consists of ground with scattered plants and rocks of the same size or larger than the avatars.

The basic aim of our environment is for users to meet; consequently the design of the world is circular like a traditional maze consisting of cubicles or areas connected by gates and separated from each other by elements that hide the view between them partly or completely. In the centre of the environment there is a large zone acting as a forum or general meeting point. The users start out in the zones that are farthest from the forum and explore the world until they gradually converge in the forum.

2.3 The Interface

The interface used for the system is based on the Active World[2] browser. It consists mainly in two sections, one for navigating the WWW, and another for visualizing the virtual world. The virtual world and the web window are linked, so an object or event in the virtual world may launch the search and the display of a certain web page which is related to it.

The user is allowed to write text in a chat line, and to click on the buttons of the interface, and it delegates all the external behavior and movement of the avatar (something experimental, not common in 3D interfaces) on the attached personalized agent.

3 The Mind of the Avatar-Agent

As far as the mind is concerned, the actions that an avatar can perform (and therefore are eligible to be automated) can be classified into two categories: *expressions*, which can be in turn separated into verbal and non-verbal expressions; and *tasks*, which can be divided into reflex tasks and conscious tasks. Some reflex tasks, such as breathing, blinking and having a dynamic glance, contribute to increase the avatar's appearance of life [14], and, since they are reflex, it seems obvious that the user must be freed

[2] http://activeworlds.com

from the management of these tasks. Given that they are highly related to the personality and mood of the avatar, a proactive attitude is required to manage them.

Besides, when a conscious action has to be automated, it must be performed as the user would do it, according to their current mood, their personality traits, attitudes... Again, a personalised and proactive control must be provided.

Finally, the avatar must always show an external expression that is coherent with its internal psychological model. The management of this external expression is also a good candidate be automated, letting the avatar learn from the user's behaviour to provide the right expression in every moment.

In all of these situations, when an avatar has to select an action to perform, this decision must be made according to several factors, as it is shown in Fig. 1.

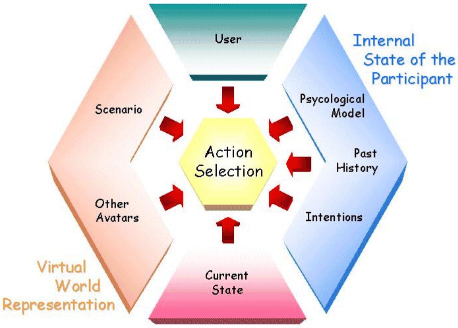


Fig. 1. Influential factors in an agent's action selection.

- The *internal state of the participant*, where individual characteristics of the avatar are defined and initialised.
- The *representation of the VE*, defined in terms of other avatars and objects.
- The *current state of the scenario*, with a suitable representation of what is happening around (for instance, the state of a game which is being played).
- The *user explicit commands*. Commands are actions that should be performed by the avatar as soon as possible (e.g. open the door).

Within the internal state of the participant, we distinguish a *psychological model*, the intentions entrusted to the avatar, and the *past history*. The last one is of a great importance when a high degree of autonomy is given to the avatar. The past history will be useful to maintain a coherent behaviour and to have some memory to make intelligent decisions.

The psychological model is the component that makes possible to build avatars as social individuals with personality and emotions. The body language of an avatar and how it carries out every action will be largely influenced by its personality. Some other authors have already proposed the use of personality models for autonomous agents, but very few have dealt with this aspect in avatars, that is, characters that receive directions from a user, and whose decisions are always conditioned by the user's wish. And, even more, those who have done it, scarcely explore the explicit interaction from the avatar to its own user.

The Internal Model that we propose consists of *personality traits*, to mark out the general lines of every participant's behavior, and distinguish an individual from all others [6]; *moods*, to show the emotional state of a participant in a fixed moment, distinguishing between the classical mutually exclusive moods proposed by most of the authors [4][9], and non-exclusive moods, which complement and qualify the mutually exclusive moods; *attitudes*, to determine the behaviour of a participant in their relationship with other participants, and *intentions*, to express the goals entrusted by the user to their avatar. All these components have been widely explained in previous works [7].

Traits, moods and attitudes are not independent, but they must be closely connected. Thus, personality traits have influence, in a higher or lower degree, over moods and attitudes, whereas attitudes also have influence over moods.

4 The Link between the Environment and the Avatar-Agent: The Awareness System

The awareness system is made of three interconnected components:

- *Events and Goals*: In our system, each avatar-agent has got an Event Perceiver that will allow it to react semi-autonomously to the external events.

  An avatar-agent is semi-autonomous, meaning that it takes its own decisions of what to do after an event, for example moving the head towards the nearby avatar or going away, but under certain circumstances it may question the user about the next action. This is a way of enriching the Group and Contextual Awareness perceived by the user, that so far has only being fed by the movements of the other avatars. The avatar will behave accordingly to the answer of the user.

- *Non Verbal Communication*: The Non Verbal Communication gestures that the avatars may perform can be classified into six groups: gazes, listening, reactions, movements and communication gestures.

  In order to simplify the animation database, gazes, listening and reaction gestures are considered to be directional, i.e. the avatar is pointed in one direction and the head and ears in another. The other gestures occur in front of the avatar.

- *Camera Management*: One of the main problems with Group, Contextual and Peripheral Awareness in virtual environments concerns the use of the camera. The lack of proprioception between users and their semi-automated avatars requires the system to provide some sort of feedback about the gestures made by avatars. A camera always placed outside the avatar usually provides such feedback, but in this position a considerable amount of the feeling of immersion obtained from a subjective viewpoint is lost.

  The automation of avatars' narrative aspects envisaged in the system offers an interesting solution to the problem of managing the viewpoint used to observe the story. The camera can also be automated and situated in different viewpoints according to the narrative situation in which users and their avatars are located.

This solution has the advantage of enabling camera language to be used in a virtual world, with its inherent narrative detail. This includes and controls the feedback received by users about their own gestures and the impact they generate.

5 Conclusions

One of the more attractive qualities for communication of VE is to provide with a bigger capability for user immersion than traditional systems. The disadvantage is the complexity of the interface management whenever high believability is sought.

The proposed approach of automating through a personalised agent some tasks, quite important for the improvement of the communication, although less interesting for the user to be directly managed, is an alternative to be taken into account to increase VE usability without affecting the immersion itself.

In fact, that has been the empirical perception of the users who have experimented our approach. They have preferred a VE with this alternative against "traditional" VE or the usual chat line. Besides, it has been made evident that a key for obtaining believability is an adequate avatar personalisation, that is, a precise internal model for the avatars.

References

1. Arnheim, R.: Art and Visual Perception: A Psychology of the Creative Eye. University of California Press, Berkeley, Los Angeles (1974)
2. Culhane, S.: Animation. From Script to Screen. St. Martin Press, New York. (1988)
3. Eisner, W.: Comics & Sequential Art. Spanish edition: Norma Editorial, Spain (1985)
4. Ekman, P., Friesen, W.: Facial Action Coding System. Consulting Psychol. Press (1978)
5. Guye-Vuillème, A., Capin, T., Pandzic, I., Magnenat, N., Thalman D.: Non-verbal Communication Interface for Collaborative Virtual Environments. The Virtual Reality Journal, Vol. 4. Springer-Verlag, Berlin Heidelberg New York (1999)
6. Hayes-Roth, B., van Gent, R., Huber, D.:Acting in Character. In: Trappl, R., and Petta, P. (eds.): Creating Personalities for Synthetic Actors. Springer-Verlag Lecture Notes in Artificial Intelligence (1997) 92 – 112
7. Imbert, R., de Antonio, A.: The Bunny Dilemma: Stepping between Agents and Avatars. TWLT 17-CEvoLE 1 Learning to Behave. Workshop I: Interacting Agents. Proceedings of the 17th Twente Workshop on Language Technology. Enschede, The Netherlands (2000)
8. Maes, P.: Artificial Life Meets Entertainment: Lifelike Autonomous Agents. In: Communications of the ACM, Vol. 38, No 11 (1995) 108–114
9. Pelachaud, C., Badler, N., Steedman, M.: Generating Facial Expression for Speech. In: Cognitive Science, No 20 (1995) 1 – 46
10. Reilly, W.: A Methodology for Building Believable Social Agents. In: Proceedings of the First International Conference on Autonomous Agents, Marina del Rey (1997) 114–121
11. Reilly, W., Bates, J.: Natural Negotiation for Believable Agents. Technical Report CMU-CS-95-164. Carnegie Mellon University (1995)
12. Rousseau, D., Hayes-Roth, B.: Improvisational Synthetic Actors with Flexible Personalities. Report No. KSL-97-10. Knowledge Systems Laboratory. Department of Computer Science. Stanford University, California (1997)

Wizard of Oz Method for Learning Dialog Agents

Masayuki Okamoto[†], Yeonsoo Yang[†⋆], and Toru Ishida[†‡]

[†]Department of Social Informatics, Kyoto University
Yoshida Hommachi, Sakyo-ku, Kyoto, 606-8501 Japan
[†]CREST, Japan Science and Technology Corporation
{okamoto,soo}@kuis.kyoto-u.ac.jp, ishida@i.kyoto-u.ac.jp
http://www.lab7.kuis.kyoto-u.ac.jp/

Abstract. This paper describes a framework to construct interface agents with example dialogs based on the tasks by the machine learning technology. The Wizard of Oz method is used to collect example dialogs, and a finite state machine-based model is used for the dialog model. We implemented a Web-based system which includes these functions, and empirically examined the system which treats with a guide task in Kyoto through the experimental use.

1 Introduction

There are many synthetic interface agents which introduce some Web sites through a text-based dialog. For example, Jennifer in Extempo [1] sells cars virtually, and Luci in Artificial Life [2] introduces her own Web site. They play the role of guides, and prevent users from clicking all links of these sites. The number of such kind of interface agents is increasing.

However, when a designer decides to make an agent, he/she should design an internal model per domain, and he/she should implement it. It costs very much to construct each agent in different way.

In this paper, we propose a framework to construct task-oriented dialog agents from example dialogs semi-automatically. For the dialog model of each agent, we use a finite state machine (FSM). For collecting the example dialogs, we use the Wizard of Oz (WOZ) method [4] which is originally used for prototyping of dialog systems.

2 Learning Process

This section describes the learning process of the agent.

For constructing a learning dialog agent, the following elements are needed:
(a) The internal model and learning mechanism of which the agent consists.
(b) The environment and method for collecting dialog examples of good quality.

[⋆] She is currently working at Toshiba Corporation.
[1] http://www.extempo.com/
[2] http://www.artificial-life.com/

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 20–25, 2001.

We use a finite state machine for the dialog model or structure of the agent, and the WOZ method for collecting dialogs.

There are some FSM-based systems. In particular, the VERBMOBIL uses a layered architecture including FSM [1]. Our approach differs from usual approaches. We try to provide a simple method to construct systems with limited scenarios and with the assist by human, though each agent is used in a narrow domain or task.

Figure 1 shows the conceptual process of constructing agents.

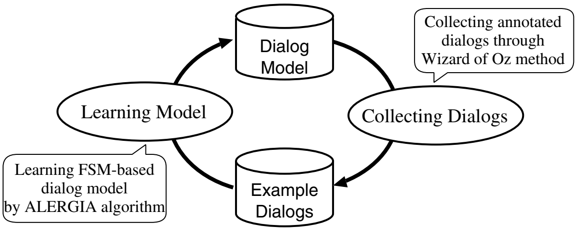

Dialog Model
Collecting annotated dialogs through Wizard of Oz method
Learning Model
Collecting Dialogs
Learning FSM-based dialog model by ALERGIA algorithm
Example Dialogs

Fig. 1. Conceptual process

Finite State Machine-based Dialog Model

We consider each dialog as a series of utterances. Each utterance consists of (a) the speaker, (b) the content, and (c) the utterance tag decided based on the task design.

The dialog model is constructed by the *ALERGIA* algorithm [2], which was originally used for the grammatical inference area. We try to use it for an actual dialog systems. The algorithm works as follows.

First, a tree-formed FSM is constructed from the examples. Then, the compatibility of each pair of two states is examined. If the probability of any their suffixes are compatible, the two states are merged. Finally, a generalized FSM is constructed.

In this paper, each utterance tag means an FSM symbol, the learned FSM means a dialog model, and the original example dialogs mean the plausible sets of a user's inputs and the agent's outputs. Figure 2 shows the dialog structure.

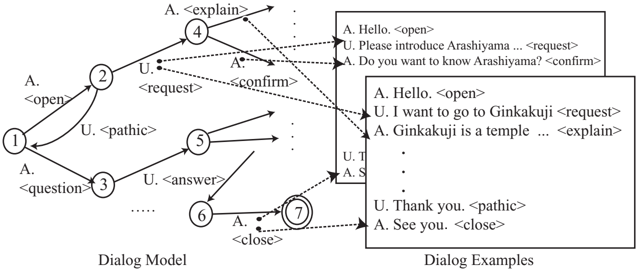


Fig. 2. Dialog model and example dialog

We introduce the keyword score of each words to recognize the user's utterance. When a user inputs an utterance, the nearest example utterance is calculated by the simple dot product of word score vectors.

Applying Wizard of Oz Method to Collect Example Dialogs

The *Wizard of Oz (WOZ)* simulating [4] is a method that a user and a person called *Wizard* who behaves as if he/she were a system talk together. It is because a human-human dialog should not be applied to human-computer dialog interfaces. There are differences in utterances used when a user thinks the partner of the communication is a computer and utterances used when he/she thinks the partner is a person [3]. This method is usually used for prototyping a system.

We apply the WOZ method to develop learning interface agents. This framework has the following two features:

Example-based development Instead of dialogs which developers *guess*, dialogs which users and the Wizard *actually talk* are used for the systems. The range of actions which a person does are limited if the situation is established clearly [6]. Therefore, the example-based approach is suited to the construction of dialog agents.

Human-assisted development The role of the Wizard is to assist the learning process as well as supplementing functions of the system. At the beginning, the system with a few dialog examples has little intelligence. As the system progresses, the role of Wizard is replaced by the system, and the system can be evolved to the real interface agent. Therefore, we consider the WOZ as the human-assisted method for learning interface agents[3].

3 WOZ-based Agent System

We implemented a support system with the mechanism described in Section 2.

The system architecture is shown as Figure 3. Each component of the architecture has the functions and contents as below:

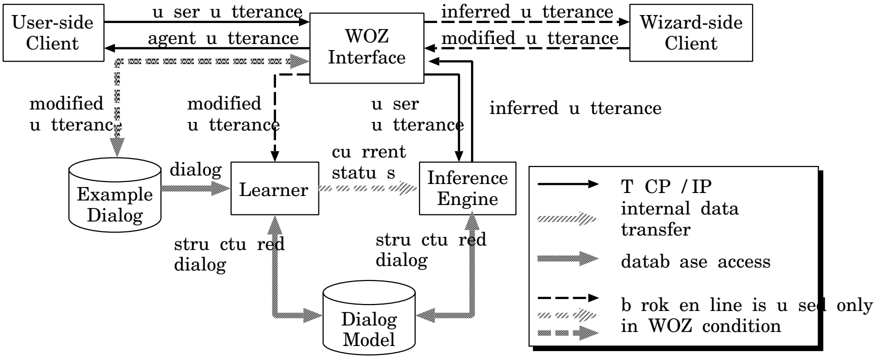
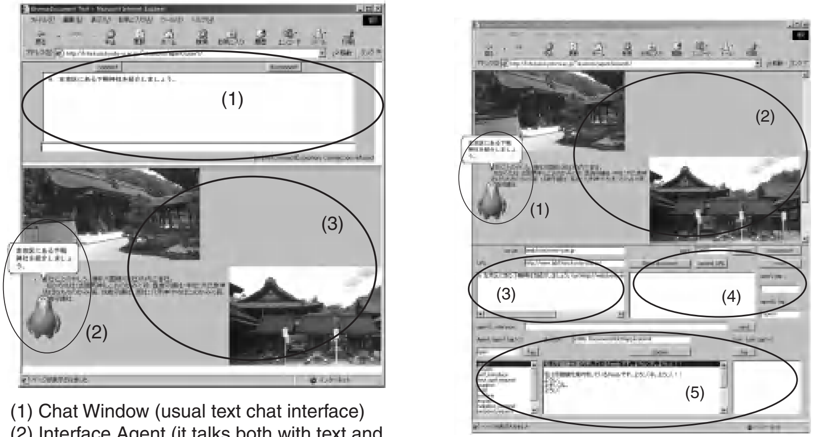


Fig. 3. Architecture of learning interface agent

User-side Client It is used by a user. The current version (Figure 4(a)) is a chat system which is expanded with an interface agent of Microsoft Agent[4]. The agent talks both with text and speech. When the agent talks about a topic, a Web page related to the dialog will be shown in the Web browser.

[3] In the WOZ condition, both the Wizard and the system play the role of the agent.

[4] http://msdn.microsoft.com/msagent/



(1) Chat Window (usual text chat interface)
(2) Interface Agent (it talks both with text and voice)
(3) Web Page Related to Contents (When a Web page is associated to the agent's utterance, the page pops up at the same time the agent speaks)

(a) User-side client

(1) Interface Agent
(2) Web Page Related to Contents (the same page the user sees)
(3) Log Window (chat log)
(4) Inferred Message Window (inferred utterances by the system)
(5) Preset Dialogs and Utterance Tags (the Wizard can input or modify the utterance)

(b) Wizard-side client

Fig. 4. Screenshot of each client

Wizard-side Client The Wizard uses a client (Figure 4(b)) which is the extended version of the User-side Client. It has two additional features. (a) The Wizard can associate a related Web page with a dialog. With this feature, the agent can explain each topic on the corresponding Web page. (b) The Wizard can select an utterance from inferred utterances by the system or preset utterances from the menu.

WOZ Interface It controls the dialog stream among each client, the Learner, and the Inference Engine. When the Wizard-side Client is disconnected, the whole system works as an individual agent system.

Example Dialog It has the collected dialogs. Each utterance is annotated by the Wizard.

Dialog Model It has the current FSM and all utterances of which the FSM consists. The FSM is constructed by the Learner, and referred by the Learner and the Inference Engine.

Learner It constructs an FSM from annotated dialogs in the Example Dialog, and updates each word score of utterances in the WOZ condition according to the utterances from the WOZ Interface.

Inference Engine It infers next utterance from an input from the user. Then, it sends the utterance to the Wizard in the WOZ condition, otherwise it sends the utterance to the user. In the WOZ condition, it also receives the current status (including current FSM state) from the Learner.

The system flow is as below:

1. When a user inputs an utterance, the Inference Engine infers the next utterances and sends them to the Wizard-side Client via the WOZ Interface. If there is not the Wizard, the ‘best’ candidate is sent to the user.

2. When the Wizard selects a candidate, the inferred utterance and its utterance tag are sent to the Learner and the Example Dialog.
3. If there are not any proper candidates, the Wizard has to select an utterance from the preset utterances or input one, and annotate both the user's utterance and the agent's utterance. The Wizard can add a URL to the utterance if the Wizard wants to show a Web page to the user.
4. In any cases, the utterance is spoken by the interface agent. If a Web page is associated with the utterance, the page is shown on the User-side client.

4 Empirical Evaluation : Kyoto Guide Task

This section shows an agent Peedy which introduces tour spots in Kyoto, Japan.

Setting

In this task, the agent talks with a user along to the following flow:

First, Peedy introduces itself. Then, he asks the user what he/she wants to know (e.g., temples, shrines, and so on). If there are a request and any correspoonding example dialogs, he explains the tour spot; otherwise, he suggests a tour spot. Finally, there are not any requests, he closes the dialog.

We collected 42 dialogs consists of 410 utterances by the agent and 167 utterances by users through the WOZ method. The example includes 103 tour spots. After the ALERGIA algorithm (the parameter $\alpha = 0.975$), we obtained an FSM-based model with 127 states and 380 transitions. Figure 5 shows an example dialog between the agent system with this model and a user. We observed 9 dialogs among the system with the learned dialog model and student users.

A. (means agent) Hello, I'm Kyoto tour guide, Peedy. Nice to meet you. ⟨open⟩
A. I was a guide of ... Where do you want to go sightseeing? ⟨self introduction⟩
U. (means user) Please tell me Nijo Castle. ⟨request TS⟩
A. Nijo Castle is ... ⟨explain TS⟩
(A Web page related to Nijo Castle appears in user's browser)
U. I see. ⟨response⟩
A. Do you want to go to another place? ⟨lead to another TS⟩
U. I want to buy my souvenir. ⟨request TS⟩
A. Can I introduce a famous shrine in Kyoto? ⟨suggest TS⟩
U. Yes. ⟨request TS⟩
A. I'll introduce Kamo-Betsurai shrine in Kita-ku. ... ⟨explain TS⟩
U. I see. ⟨response⟩
......
A. How about today's guide? Is it good? ⟨lead to closing⟩
U. Yes. ⟨accept closing⟩
A. See you. ⟨close⟩

Fig. 5. Example dialog (the original dialog is in Japanese)

Lessons Learned

We found the following empirical result about our system from the actual use:

- The task and domain
  There were many tour spots which are not in the example learning set. It means the domain is larger than what we assumed.
- The learning algorithm and inference process
  There were two kinds of users. The first kind of persons requested different

tour spots one after another, and they often input only noun words as using a search engine. The other kind of persons replied to the agent's each answer explanatorily. Some dialogs resulted in complete failures.
When the agent spoke a wrong utterance, some users asked the agent again. It means they could not find what the agent did not understand, e.g., the agent did not know the spot, or the spot is known in another name.

5 Conclusion

In this paper, we proposed a framework to construct a learning dialog agent from example dialogs. For the dialog model of the agent, we used the FSM constructed by the ALERGIA algorithm. For collecting the example dialogs, we use WOZ method. We implemented a Web-based system and empirically examined the system which treats with a guide task in Kyoto through the experimental use.

Future work includes extending the mechanism to improve dialog management, and to treat with much knowledge and more difficult situations. Many dialog failures are from users' embarrassment that the user could not suppose the agent's internal state. The next version needs to show their status more than the current version. In our example-based approach, all knowledge is in the example. When we design agents for wider domains, it is necessary to treat with the knowledge and example dialogs independently.

We also consider that the tour-guide agent in Kyoto will be applicable in Digital City Kyoto [5].

Acknowledgement

We would like to thank associate professor Yasuhiko Kitamura at Osaka City University for his devoted advice. The work is partially supported by the New Energy and Industrial Technology Development Organization for Senior Support System Project, by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (A), 11358004, 1999, and by the Japan Science and Technology Corporation for Core Research for Evolutional Science and Technology Digital City Project.

References

1. J. Alexandersson, E. Maier and N. Reithinger, "A Robust and Efficient Three-Layered Dialogue Component for a Speech-to-Speech Translation System," *Proc. EACL-95*, pp. 188–193, 1995.
2. R. C. Carrasco and J. Oncina, "Learning Stochastic Regular Grammars by Means of a State Merging Method," *Proc. ICGI-94*, pp. 139–152, Springer-Verlag, 1994.
3. J. M. Carroll and A. P. Aaronson, "Learning by Doing with Simulated Intelligent Help," *Communications of the ACM*, Vol. 31, No. 9, pp. 1064–1079, 1988.
4. N. M. Fraser and G. N. Gilbert, "Simulating Speech Systems," *Computer Speech and Language*, Vol. 5, No. 1, pp. 81–99, 1991.
5. T. Ishida, J. Akahani, K. Hiramatsu, K. Isbister, S. Lisowski, H. Nakanishi, M. Okamoto, Y. Miyazaki and K. Tsutsuguchi, "Digital City Kyoto: Towards A Social Information Infrastructure," *Proc. CIA-99*, pp. 23–35, Springer-Verlag, 1999.
6. B. Reeves and C. Nass, *The Media Equation*, Cambridge University Press, 1996.

Supporting User-Profiled Semantic Web-Oriented Search

Luigi Palopoli, Domenico Rosaci, Giorgio Terracina, and Domenico Ursino

DIMET - Università "Mediterranea" di Reggio Calabria
Via Graziella, Località Feo di Vito, 89100 Reggio Calabria, Italy
{palopoli,rosaci,terracina,ursino}@ing.unirc.it

Abstract. This paper proposes a technique which allows to select Web sources on the basis of a semantic reconstruction of user's preferences. A conceptual model, called SDR-Network, and a probabilistic Description Logic, called DL_P, are exploited to this end.

1 Introduction

In order to foster the development of new Web-based information-distribution systems, it is relevant to be able to obtain a *user-based* view of "available" information. The exponential increase of the size and the formats of remotely accessible data imposes to find suitable solutions to the problem. Often, today's information access tools are not able to provide the right answers for a user query but, rather, provide large supersets thereof (e.g, in Web-search engines). The problem basically relies on such tools being based on syntax-driven retrieval techniques, whereas semantic-oriented search would be needed.

This paper provides a contribution in this setting. The basic idea is to use suitable representations of both the available information sources and the user interests in order to be able to match as appropriately as possible, user information needs, as expressed in her/his query, and available information. To this end, we exploit two formalisms: *(1)* a conceptual model, called the SDR-Network [8], which is used in order to represent information source content and extract semantic relationships possibly holding among such contents and represent user information preferences, in the form of a user profile; *(2)* a probabilistic variant of description logics, called DL_P [6], in order to reason about source contents and user preferences. Space limitations do not allow us to include a detailed description of SDR-Networks and DL_P, for which the reader is referred to [8,6].

Our objective is to obtain a reconstruction of the semantics of a set of available information sources as filtered using actual user interests encoded in a user profile. In order to do that, the user provides an initial set of sources she/he is interested in. The semantic contents of this set of sources is then derived. This is done by first "translating" each available information source into a corresponding SDR-Network. Using these SDR-Networks, we compute: *(i)* a set of intrasource knowledge patterns, represented as DL_P assertions, by which relationships holding among concepts presented within one source are made explicit;

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 26–31, 2001.

(ii) a set of intersource properties [1,2,3,4,8] denoting elementary relationships holding among data presented in distinct sources. The set of intersource properties and the set of intrasource knowledge patterns are needed in order to be able to reconstruct the "cumulative" concept associated to a certain term found in several information sources of interest and to relate it to user preferences. This is done by using an inference mechanism associated to DL_P formulae to derive a set of "complex knowledge patterns" describing properties relating available information and user interests, as expressed in a set of terms she/he provides. This set of complex knowledge patterns is used to obtain the starting user profile. Such profile is not fixed, though. Indeed, it is expected that the set of interesting sources may change over time. In such cases, the user profile is updated in order for it to properly represent the modified situation.

Querying works as follows. Let P be the current user profile (querying can take place independently and in parallel with profile updating soon after the initial profile is constructed) and let Q be a query submitted by this user. In order to answer Q, the following steps are carried out: *(i)* the query Q is submitted to a traditional (syntax-based) information retrieval tool; let S_{Synt} be the set of returned sources; *(ii)* a structural filtering on S_{Synt} is carried out by analyzing structural properties of the portions of information sources matching Q; structural properties pertain the relative topology of the associated SDR-Networks; this filtering step produces a subset S_{Struct} of S_{Synt}; *(iii)* a further, profile-based, filtering is performed on S_{Struct} in order to discard those information sources whose semantic content does not match specific user's information needs, as expressed in the query Q; *(iv)* a score is assigned to thus selected information sources, classifying them on the basis of their overall correspondence with P and Q.

It is important to point out that the presented technique is not intended to serve the user in generalized Web searches. This issue is indeed important and we shall pursue this generalization in the future. In order to extend our technique to Web-wide searches we need to define a light-weight version of our SDR-Network and to devise "on-the-fly" translations of both data sources into "light" SDR-Networks and SDR-Networks into DL_P assertions.

2 Extracting Complex Knowledge Patterns from SDR-Networks

As pointed out in the Introduction, our technique for semantic retrieval requires the construction of a profile for each user and the derivation of correlations existing between the user profile and source content, as encoded in a set of DL_P assertions, and reasoning with it about relationships relating source content and user profiles. These assertions will be referred to as knowledge patterns in the following. (We refer to [7] for details about reasoning with DL_P). The method consists of three steps: *(i)* translating each involved SDR-Network into a set of DL_P entities, relationships and assertions (*intrasource knowledge patterns*); *(ii)* deriving *basic intersource knowledge patterns* by exploiting intrasource knowledge patterns and synonymies of concepts in sources derived by applying the technique of [8]; *(iii)* extracting *complex knowledge patterns* by combining two or more knowledge patterns for obtaining further, more complex, ones.

3 Construction and Updating of the User Profile

In this section we describe how a user profile can be obtained “from scratch” from an initial set $S = \{S_1, \ldots, S_n\}$ of information sources provided by the user. These can be databases, XML documents, semi-structured information sources, and so on. User profiles are represented by SDR-Networks.

3.1 Constructing the Initial User Profile

The process consists of the following steps: *(i)* Translating all information sources of S from the original formats into SDR-Networks. This task is carried out with the support of rules described in [8]. *(ii)* Deriving the set KP_S of complex knowledge patterns relative to S; this step is carried out exploiting the inference process referred to in Section 2. *(iii)* Deriving the set $IKP_S \subseteq KP_S$ including those knowledge patterns possibly interesting for the user. The user is asked to indicate the set CS of concepts she/he is interested in and IKP_S is obtained as those knowledge patterns of KP_S that subsume some of the concepts of CS. *(iv)* Constructing the user profile by translating the set IKP_S into an SDR-Network; to this end, translation rules described in [7] must be applied.

Our technique returns also a set SS of the form $\{\langle Net_{S_i}, L_i \rangle\}$, where Net_{S_i} is the SDR-Network associated to the information source S_i, and L_i is the set of concepts of CS also represented in Net_{S_i}. In the following, the set SS will be referred as ISC_Set (Interesting Source and Concept Set).

3.2 Updating the User Profile

In this section we describe a semi-automatic technique for updating the user profile P and the associated ISC_Set when a new information source becomes available. Note that, if at some time the user is no longer interested in a certain concept C, the updating of the profile consists just in removing from P the node associated to C and all its arcs, whereas the updating of the set ISC_Set is carried out by removing every occurrence of C from it.

The task of evaluating if a new information source S_i is interesting for a user U is carried out by a function σ. This receives the profile P of a user U, the ISC_Set SS associated with U, the set CS of interesting concepts of U and the SDR-Network Net_{S_i} to evaluate. σ verifies if S_i is interesting for U and, in the affirmative case, adds Net_{S_i} to SS and updates P accordingly, returning the (possibly) modified SS and P. σ evaluates the interest of U for Net_{S_i} in two steps: first it determines the set of lexical similarities between those nodes of P directly corresponding to concepts of CS, and the nodes of Net_{S_i}[1]; then, for each lexical similarity, it checks if a corresponding semantic similarity holds. If σ derives at least one semantic similarity between nodes of P and Net_{S_i}, it adds Net_{S_i} to SS and updates P. σ is defined as $\xi(P, SS, CS, \theta(P, Net_{S_i}, \lambda(P, CS, Net_{S_i})))$

The function λ. On the basis of the content of a standard thesaurus, such as WordNet [5], λ returns pairs of lexically similar nodes selected from the cartesian product $NP \times NS_i$, where NP is the the set of nodes of P directly corresponding to concepts expressed in CS, and NS_i is the set of nodes of Net_{S_i}.

The function θ. θ returns a pair $\langle Net_{S_i}, SSP\rangle$, where Net_{S_i} is the (possibly modified) SDR-Network and SSP is the set of pairs of nodes determined to be *semantically* similar. Given two sets of nodes NS_1 and NS_2, we say that $NS_1 \sqsubseteq NS_2$ if, for each node N_a in NS_1, there exists a lexically similar node N_b in NS_2 and, for each node N_b in NS_2, N_b is lexically similar to at most one node in NS_1. For each pair $[N_P, N_S] \in LSP$ such that $N_P \in P$ and $N_S \in Net_{S_i}$, θ examines the sets NS_P and NS_S of nodes directly connected by an arc to N_P and N_S, resp. NS_P and NS_S correspond to the sets of nodes which are directly involved in the definition of the semantics of N_P and N_S. Three cases are to be considered *(i)* $NS_S \sqsubseteq NS_P$, in which case N_S and N_P are considered semantically similar. *(ii)* $NS_P \sqsubseteq NS_S$ and $NS_S \not\sqsubseteq NS_P$. In this case, we can assume that N_S is an hyponym of N_P. Thus, since we are interested in semantic similarity relationships, Net_{S_i} is modified in order to have in it a new node N_S^f semantically similar to N_P. Then, all nodes directly connected to N_S, having a lexically similar node in NS_P, are connected to N_S^f and old connections to N_S are removed. In addition an arc $\langle N_S, N_S^f, [0,1]\rangle$, denoting an *"is-a"* relationship, is added to Net_{S_i}. *(iii)* $NS_S \not\sqsubseteq NS_P$ and $NS_P \not\sqsubseteq NS_S$. In this situation θ verifies the "similarity degree" of the two sets by computing the percentage of nodes of NS_S which are lexically similar to nodes of NS_P. N_S and N_P are considered semantically similar only if this percentage is above a certain given threshold th.

The function ξ. ξ receives the profile P of one user U, the ISC_Set SS associated to U, the set CS of concepts interesting for U and the pair $\langle Net_{S_i}, SSP\rangle$ returned by θ. First ξ examines SSP; if this is empty, it stops, otherwise it updates P and adds to SS the pair $\langle Net_{S_i}, L_i\rangle$, where Net_{S_i} is the SDR-Network determined to be interesting for U and L_i is the set of concepts of CS having a corresponding node in P semantically similar to a node of Net_{S_i}. In order to update P, ξ first derives complex knowledge patterns about concepts of P and Net_{S_i}. Then, ξ selects from derived patterns those of the form $L_1 \dot{\leq}_{W_{\langle L_1, L_2\rangle}} L_2$, where L_1 is a class expression inferred from either Net_{S_i} or P whereas L_2 is a DL_P entity representing a node of P corresponding to one of the concepts of CS. Finally, the updated P is obtained by translating the selected patterns into an SDR-Network. ξ returns such modified SS and P.

[1] Recall that each SDR-Network node is identified by a name; therefore the lexical similarity considers the node names.

4 Semantic Web-Oriented Search

Suppose that a user U, having a profile P and a set IS of information sources of interest, submits a query Q upon IS. The proposed technique for searching sources in IS that are semantically relevant to Q consists of four steps. The function realizing these four steps is as follows:

$$S_{Filter} = \phi(P, Q, IS) = \chi(P, Q, \phi_{Profile}(P, Q, \phi_{Struct}(Q, \phi_{Synt}(Q, IS))))$$

The function ϕ_{Synt}. ϕ_{Synt} activates a standard search tool on IS for obtaining the set of information sources syntactically relevant to Q. It returns the set $NetS_{Synt}$ of the SDR-Networks corresponding to these information sources.

The function ϕ_{Struct}. ϕ_{Struct} receives the set $NetS_{Synt} = \{Net_{S_1}, \ldots, Net_{S_n}\}$ of the SDR-Networks syntactically relevant to Q, as returned by ϕ_{Synt}, and carries out a structural check on each of them. It is as follows:

$$\phi_{Struct}(Q, NetS_{Synt}) = \bigcup_{i=1..n} \phi'_{Struct}(Net_{S_i}, \zeta(Q, Net_{S_i}))$$

In order to comprehend the behaviour of ϕ'_{Struct}, consider that a user query Q can be seen as a set of terms related to each other by AND, OR and NOT connectives. It is always possible to represent Q in CNF, as $Q = Q_1 \vee ... \vee Q_l \vee \neg Q_{l+1} \vee ... \vee \neg Q_m$; here each Q_i is of the form $T_{i_1} \wedge ... \wedge T_{i_n}$, where each term T_{i_j} can represent either a concept or an instance of a concept. However, we are interested in a semantic analysis of Q; therefore, it is necessary to find, for each SDR-Network Net_{S_i}, the concepts associated to those terms in Q which correspond to instances in the source corresponding to Net_{S_i}. This task is carried out by the function ζ which receives the query Q and an SDR-Network Net_{S_i} and returns a CNF boolean expression E_i derived from Q by *(i)* substituting each term of Q being an instance of a concept with the corresponding concept of Net_{S_i}; *(ii)* deleting negative literals from Q; this is possible because the sources returned by the standard search tool contain neither concepts nor instances corresponding to negative literals of Q. As a consequence, E_i has the form $E_i = E_{i_1} \vee ... \vee E_{i_l}$, where each E_{i_j} is of the form $C_1 \wedge ... \wedge C_n$ and C_k, $1 \leq k \leq n$ is a concept of Net_{S_i}. The function ϕ'_{Struct} carries out a structural check on the SDR-Network Net_{S_i}, as follows:

$$\phi'_{Struct}(Net_{S_i}, E_{i_1} \vee ... \vee E_{i_l}) = \begin{cases} \{Net_{S_i}\} & if(\bigwedge_{j=1..l} \phi''_{Struct}(Net_{S_i}, E_{i_j})) = true \\ \emptyset & \text{otherwise} \end{cases}$$

ϕ''_{Struct} returns true if all the concepts included in E_{i_j} are strongly related to each other. Here we assume that a concept C_h is strongly related to a concept C_k if, in Net_{S_i}, the nodes corresponding to C_h and C_k are connected through a path in which the sum of semantic distance coefficients of the arcs constituting the path il less than 2 (we call this kind of path *neighbor-path*)[2].

The function $\phi_{Profile}$. $\phi_{Profile}$ receives the set $NetS_{Struct} = \{Net_{S_1}, \ldots, Net_{S_m}\}$ of the SDR-Networks returned by ϕ_{Struct} and carries out a profile-based check on each of them. It is encoded as:

$$\phi_{Profile}(P, Q, NetS_{Struct}) = \bigcup_{i=1..m} \phi'_{Profile}(P, Net_{S_i}, \zeta(Q, Net_{S_i}))$$

[2] In the following, in order to simplify the notation, we indicate by C_k both the concept in S_i and the corresponding node in Net_{S_i}.

Here, the function ζ is as described above. The function $\phi'_{Profile}$ receives a user profile P, an SDR-Network Net_{S_i} and the CNF boolean expression E_i associated to Net_{S_i} and Q and verifies if at least one of those nodes of Net_{S_i}, having a corresponding concept in E_i, is semantically similar to a node of P. In the affirmative case, $\phi_{Profile}$ returns $\{Net_{S_i}\}$, otherwise it returns $\emptyset$. In order to carry out its task, $\phi_{Profile}$ exploits the technique for verifying the existence of a semantic similarity between a node of an SDR-Network associated to an information source and a node of a user profile described in Section 3.2, when discussing about the function θ.

The function χ. χ receives a user profile P, a query Q and the set $NetS_{Profile}$ of SDR-Networks returned by $\phi_{Profile}$. χ carries out two tasks: *(i)* It first assigns a score to each SDR-Network Net_{S_i} of $NetS_{Profile}$ according to how much it matches the user profile P and the query Q. The score is computed as the ratio between the number of nodes of Net_{S_i} whose corresponding concepts are present in $\zeta(Q, Net_{S_i})$, and the number of nodes of P whose corresponding concepts are present in $\zeta(Q, Net_{S_i})$. *(ii)* classifies the SDR-Networks of $NetS_{Profile}$ on the basis of their scores. χ returns the ordered list $S_{Filter} = \{\langle IS_1, sc_1\rangle, \ldots, \langle IS_l, sc_l\rangle\}$, where IS_i is the information source corresponding to the SDR-Network Net_{S_i} of $NetS_{Profile}$ and sc_i is the score of Net_{S_i} w.r.t. P and Q computed by χ.

References

1. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.
2. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *Transactions on Data and Knowledge Engineering*, 13(2), 2001.
3. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. of International Conference on Management of Data (SIGMOD 2001)*, Santa Barbara, California, USA, 2001. ACM Press.
4. J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proc. of International Conference on Very Large Data Bases (VLDB 2001)*, Roma, Italy, 2001. Forthcoming.
5. A.G. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
6. L. Palopoli, D. Saccà, and D. Ursino. DL_P: a description logic for extracting and managing complex terminological and structural properties from database schemes. *Information Systems*, 24(5):403–425, 1999.
7. L. Palopoli, G. Terracina, and D. Ursino. Inferring complex intensional knowledge patterns from heterogeneous semi-structured information sources. Submitted for publication. Available from the authors.
8. L. Palopoli, G. Terracina, and D. Ursino. A graph-based approach for extracting terminological properties of elements of XML documents. In *Proc. of International Conference on Data Engineering (ICDE 2001)*, pages 330–340, Heidelberg, Germany, 2001. IEEE Computer Society.

Recommending a Trip Plan by Negotiation with a Software Travel Agent

Von-Wun Soo and Shu-Hau Liang

Department of Computer Science
National Tsing Hua University
101, Section 2 Kuang Fu Road, Hsinchu,
Taiwan 300, Republic of China
{soo, shown}@cs.nthu.edu.tw

Abstract. A trip plan is a compromise between the personal preferences and constraints of a traveler and the spatial, temporal, physical, cost constraints of the visiting destinations and environmental support. To recommend a trip plan to a traveler, the travel agent must find a reconciled solution among these constraints. In this paper, we demonstrate a design of a software travel agent who could recommend a trip plan by conducting the negotiation with human travelers. The recommendation and negotiation dialogue for the travel agent is driven by the mechanisms of resolution of constraint violation. In order to ensure the coherence and continuity of the plan negotiation process, we adopt a minimal alternation principle in the mechanism of the resolution of constraint violation. Finally, we illustrate with scenarios with trip plan recommendation for a tourist to Taipei city by negotiation with the software travel agent.

1 Introduction

Recommending a good trip plan for a particular traveler is often not a trivial task for a travel agent. A travel agent has to satisfy not only the customer's personal preferences and constraints, but also the complex spatial, temporal, physical and cost constraints imposed by the transportation methods among visiting spots and various environmental supports such as the availability of hotels. However, a traveler may not often express clearly his or her preferences over a trip plan at the outset unless they get more information about the visiting destinations and itinerary about the trip. Therefore, a travel agent alone cannot always decide a satisfactory trip plan for a traveler without the involvement of the travelers. Communication and negotiation with the traveler is necessary to recommend a satisfactory trip plan for the traveler.

By a trip plan we mean the details about the itinerary as well as the methods of transportation and cost estimation for the whole trip. We are particularly interested in understanding how a travel agent could conduct such kinds of recommendation and negotiation dialogues with the user and come out with a satisfactory trip plan for the user. In this paper, we focus on the techniques of recommendation by negotiation with the user and by satisfying user's personal preferences and domain constraints of various kinds. The recommendation and negotiation dialogue of a travel agent is primarily driven by the mechanisms of resolution of constraint violation.

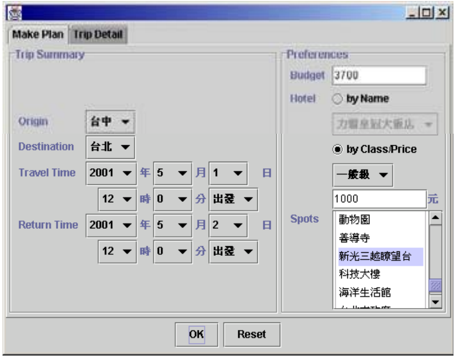
M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 32–37, 2001.

Ndumu et al. [4] identifies a number of important issues and challenges for the creators of personal travel assistant (PTA) systems that are based on a multi-agent architecture. Another work is the trading agent competition [6] for a travel agent to bid and purchase travel components at different market places in building travel packages for its clients.

2 The System Overview

Our test bed is built upon JADE [7]. The system is a simple travel application composed of a travel agent and a user interface agent

The user interface agent simply provides a user with many friendly interfaces. One is an interaction form (Figure 1), by which a user can specify his preferences for the trip and request the travel agent to arrange such a plan. When the travel agent completes the request, a resulting trip plan is returned and displayed in the plan displaying area. If a user is not satisfied with the plan, he may press the "modify" button to the right of the plan displaying area and a modification dialog box (Figure 2) will pops up to help the user to modify the plan.



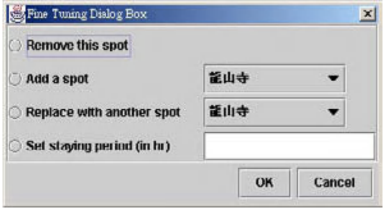
Trip Summary

Origin: 台中
Destination: 台北
Travel Time: 2001 年 5 月 1 日 12 時 0 分 出發
Return Time: 2001 年 5 月 2 日 12 時 0 分 出發

Preferences

Budget: 3700
Hotel: ○ by Name 力霸皇冠大飯店
● by Class/Price 一般級 1000 元
Spots: 動物園 善導寺 新光三越瞭望台 科技大樓 海洋生活館

Fig. 1. The interaction form



○ Remove this spot
○ Add a spot 龍山寺
○ Replace with another spot 龍山寺
○ Set staying period (in hr)

Fig. 2. The modification dialog box

The travel agent is the service center of the system, which provides several services including building a trip plan and modifications of an existing plan. A travel agent consists of a trip planner, a context user model, an inconsistency manager and a travel database about the locations and attributes of possible visiting spots, hotels and other traveling options. The context user model keeps the likes and dislikes of a user. Unlike user models of the long-term preferences, it is only concerned about the favors and disfavors in the current dialogue context. Whenever a service request is received, the travel agent first updates the context user model according to the request. Then it employs the trip planner to accomplish the user's request. The behaviors of the trip planner are constrained by both context user model and the domain constraints. If the trip planner fails to accomplish its tasks for the user by over-constraining the trip plan or specifies infeasible personal constraints, the travel agent will then invoke the inconsistency manager to resolve the failure by proposing a way for the user to relax his constraints. Details of the trip planner and the inconsistency manager are described in section 4 and 5.

The preferences for a trip that a user can specify include an origin and a destination of the trip, the departure and return time for the trip, the upper bound of the budget, a preferred hotel which can be specified by either the name, the class or the price limit, and a set of preferred spots, namely tourist attractions, and the time period to stay at each designated spot.

A valid trip plan must satisfy all the domain constraints. They are: 1) budget constraint, the total expenses of a planned trip should not exceed the budget specified by the user; 2) time constraint: all activities scheduled in the plan should be before the return time of the trip and after the departure time; 3) necessary component constraint: this specifies a minimal set of components in a trip plan. Any valid plan should contain a round-trip transportation ticket, hotel reservations if the trip exceeds more than one day, and at least one spot in the trip plan; 4) the constraint of minimal staying period: the staying period at a spot must exceed a particular amount of time. The particular amount of time reflects the reasonable minimal period in which one can enjoy visiting the spot; and 5) preferences satisfaction constraint: besides the hard constraints above, the travel agent must also try his best to satisfy users' preferences and personal constraints while arranging a trip plan.

3 Planning a Trip

The trip planner uses a three-stage planning process to build a trip plan that satisfies the user. The nature of the process is to construct the skeleton first, then fill the skeleton with the required components, and finally augment the trip plan if necessary.

At the first stage, the travel agent prepares a round-trip transportation ticket and hotels for the nights according to the user's preferences, and a number of day slots. A day slot serves as a template for a day, which consists of an empty schedule, and a set of parameters, including the start time, the end time, the starting location of the day, and the hotel for the night if necessary. After preparing the skeleton plan, the travel agent estimates the total expense so far and calculates the remaining budgets for the next stage.

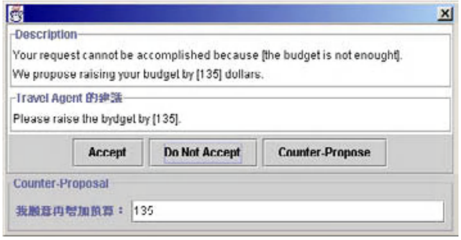
The travel agent routes the designated spots and schedules them into the skeleton plan at the second stage. It iteratively selects the spot that is nearest to the last location of a day slot, appends it to the end of the day slot, and sets the staying period to the minimal one. When all the designated spots are appended and nothing violates the domain constraints, the travel agent elongates the staying periods to reach a proper staying period of each spot, where the proper period is the recommended one proposed by the travel agent.

At the final stage, if the trip plan is not full of spots, the travel agent may recommend more spots into the trip plan as long as it causes no constraint violation.

4 Resolution of Constraint Violation and Infeasible Constraints

The trip planner sometimes fails to derive a valid trip plan that satisfies users' preferences. The reason of the failure maybe that, the users have over constrained the trip plan, namely, they have specified too many constraints or an infeasible user constraint for the travel agent to construct a trip plan.

By an infeasible user constraint, we mean the one that is directly inconsistent with the application content, or the one that does not comply with the limitation of the constraints. Specifying a non-existing hotel and specifying a staying period that does not exceed the minimal requirement are examples. Infeasible user constraints are detected at the stage of the skeleton plan formation, and the inconsistency manager simply proposes the alternatives complying with the application content and the limitations to the users.



Your request cannot be accomplished because [the budget is not enought].
We propose raising your budget by [135] dollars.

Please raise the bydget by [135].

我願意再增加預算： 135

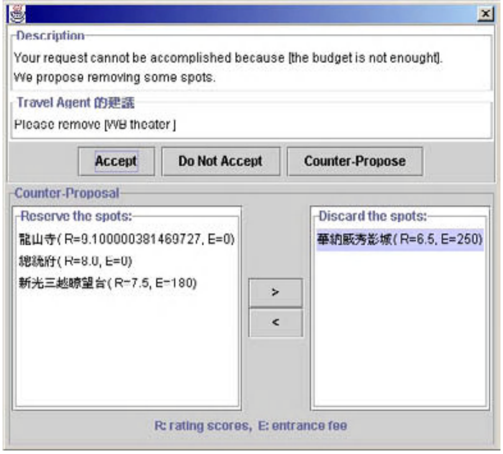
Fig. 3. The travel agent proposes raising the budget by 135 to resolve the out-of-budget situation

For the over-constraining problem, the inconsistency manager inspects the user's specifications for the trip and derives several candidate solutions to the problem. It recommends the user to adopt the candidate with minimal impact on the other parts of the plan first. If the user does not accept a proposal, the travel agent then proposes the next solution. The process is carried out by a negotiation protocol in which the travel agent can propose, and the user can also express his preferences in a counter-proposal by using the resolution dialog box (Figure 3). The resolution methods include raising the budgets, postponing the return time of the trip, changing the hotel, shortening the visit time of the spots, removing the visiting spots, changing the destination of the trip and so on.

5 Results

Here we demonstrate the results by a scenario run to illustrate the interaction between a travel agent and a user.

It starts with a user requesting a trip plan, in which the origin is HsinChu, the destination is Taipei, the departure time is 5/1 10:00AM, the return time is 5/1 19:00AM, the upper bound budget is 1200NTD, and 4 points of interest including SKM skyscraper, the Presidential Hall, WB Theater and Long-Shan Temple. When arranging such a plan, the travel agent found that the budget is not enough. It first proposes the user raising the budget by 135, but the user does not accept the proposal (Figure 3). Then it proposes removing an expensive spot, WB Theater, and the user accepts the proposal (Figure 4).


Description

Your request cannot be accomplished because [the budget is not enought].
We propose removing some spots.

Travel Agent 的建議

Please remove [WB theater]

Counter-Proposal

Reserve the spots:

龍山寺(R=9.100000381469727, E=0)
總統府(R=8.0, E=0)
新光三越瞭望台(R=7.5, E=180)

Discard the spots:

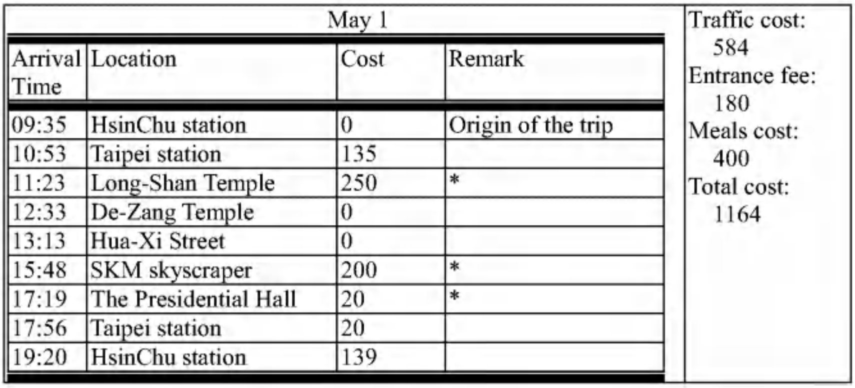
華納威秀影城(R=6.5, E=250)

R: rating scores, E: entrance fee

Fig. 4. The travel agent proposes removing WB Theater to resolve the out-of-budget situation

Since the violation of the budget constraint is resolved, the travel agent continues processing the request and returns a resulting trip plan as summarized in Table 1.

Table 1. The resulting trip plan. The cost includes the transportation cost and the entrance fee and a star (*) in the remark means the spot is designated by the user.



May 1				Traffic cost: 584 Entrance fee: 180 Meals cost: 400 Total cost: 1164
Arrival Time	Location	Cost	Remark	
09:35	HsinChu station	0	Origin of the trip	
10:53	Taipei station	135		
11:23	Long-Shan Temple	250	*	
12:33	De-Zang Temple	0		
13:13	Hua-Xi Street	0		
15:48	SKM skyscraper	200	*	
17:19	The Presidential Hall	20	*	
17:56	Taipei station	20		
19:20	HsinChu station	139		

6 Conclusion

In this paper, we argue that recommendation cannot be sometimes satisfactory in many situations unless the recommender actually negotiate with the recommendee. We have implemented a prototype system to demonstrate how a trip plan could be recommended by negotiation with a software travel agent and the negotiation dialogues are driven by the mechanisms of resolution of constraint violation. The travel agent could have several opportunities to recommend a travel component to the traveler during the negotiation process. The recommendation could also base on collaborative filtering methods or based on travel agent's rationality. For example, the travel agent might recommend some spots that are ranked higher based on the users' consensus or based on the convenience or profit of the travel agent itself. This requires more sophisticated design and is out of scope of our discussion. However, we assume the travelers be cooperative, therefore the complicated argumentation dialogues do not apply in the domain. In the future work, we could focus on more on the maintaining a coherent dialogue model in order for the travel agent to conduct more intelligent and coherent negotiation and recommendation with traveling users.

Acknowledgment. This research is supported in part by MOE Program for Promoting Academic Excellence of Universities under grant number 89-E-FA04-1-4

References

[1] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes D., and Sartin, M. (1999) Combining Content-based and Collaborative Filtering in an On-Line Newspaper, ACM SIGIR Workshop on Recommender Systems Berkeley, CA, 1999.

[2] Balabanovic M., and Shoham Y. Combining (1997) Content-Based and Collaborative recommendation. Communication of the ACM 1997.

[3] Jonsson, A. A model for dialogue management for human computer interaction, http://www.ida.liu.se/labs/nlplab/sds/public/

[4] Ndumu, D. T, Collis, J.C. & Nwana, H. S.(1998) *Towards Desktop Personal Travel Agents*. BT Technology Journal, 16(3) 1998, 69-78.

[5] Parsons, S. and Jennings, N. R. (1996) Negotiation through Argumentation – A Preliminary Report, Proceedings of the second International conference on Multi-Agent Systems, 1996.

[6] Wellman, M. P., Wurman, P. R., O'Malley, K., Bangera, R., Lin, S. D., Reeves, D. and Walsh, W. E. (2001) Designing the Market Game for a Trading Agent Competition. IEEE Internet Computing, March and April 2001.

[7] Bellifemine, F., Poggi, A., Rimassa, G., *JADE - A FIPA-compliant agent framework*, CSELT internal technical report. Part of this report has been also published in Proceedings of PAAM'99, London, April 1999, pagg.97-108.

CoWing: A Collaborative Bookmark Management System

Rushed Kanawati[1] and Maria Malek[2]

[1]LIPN-CNRS UMR Q 7030 , 99 Av. J. B. Clément, F-93430 Villetaneuse
rushed.kanawati@lipn.univ-paris13.fr
[2]LAPI-EISTI, Av. du Parc, F-95011 Cergy
maria.malek@eisti.fr

Abstract. In this paper we describe a new *distributed* collaborative bookmark system, called *CoWing* (for COllaborative Web IndexING system). The goal of the *CoWing* system is to provide a group of organized users with a computerized-support that enable them to share their experiences in managing their bookmark repositories. The *CoWing* system is composed of a set of assistant agents, called *Wings*, and a central agent that manages the user's organization. A *Wing* agent assists each user. This agent performs two main tasks: learning the user's strategy in classifying her/his bookmarks and interacting with other *Wing* agents in order to fetch new bookmarks that match the local user information need.

Keywords: Collaborative Information Agents, Bookmark, Hybrid Neural/CBR classification.

1 Introduction

Almost all World Wide Web (The web hereafter) browsers available today provide users with some *bookmaking* facility. A bookmark system enables users to save addresses of web pages (i.e. URL) or sites that users judge relevant to their information need. Typically a bookmark is a record that holds the following information: the URL of the indexed page, the page title, and some other data such as bookmark creation date, last visit date and user-provided description of the indexed page. Web usability studies show that bookmark lists are far from representing an effective personal information space [2, 4]. One approach for transforming bookmark collections into effective information spaces consists on allowing group of users to share their bookmark lists as well as their experiences in managing these lists [8, 13, 14].

Recently, a number of works have addressed the issue of developing collaborative or shared bookmark repositories. Almost all-existing systems have a centralized architecture. In [8] authors show that such an architecture does not match main requirements of collaborative applications (i.e. privacy protection, short response time, availability, tailorability, etc). In this paper we present a full-distributed

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 38–43, 2001.

collaborative bookmark system called *CoWing*. The *CoWing* system provides an infrastructure that enables an organized group of users to share their bookmark in an implicit way. By implicit, we mean that users are not required to do extra work to share their experience. The only additional work is to define other's access rules to their own repositories. A role-based access control service is also provided in order to ease this extra task. The basic idea of the system is the following: a personal agent, called a Wing agent assists each user. This agent observes the user behavior in managing her/his bookmark repository. Collected information (bookmarks and bookmark classification strategies) are then exchanged among the different *Wings* allowing each agent to recommend new bookmarks to the associated user. The *CoWing* system is detailed in section 2. First a quick overview is given in section 2.1. then the main services implemented in the systems are discussed: User's bookmark classification learning and recommendation computation services. Related work is presented briefly in section 3. A conclusion is given in section 4.

2 The CoWing System

2.1 System Overview

The *CoWing* system provides a group of *organized* users with a computerized-support that enable them to share their experiences in managing bookmarks. The overall architecture of the *CoWing* system is described on figure 1.

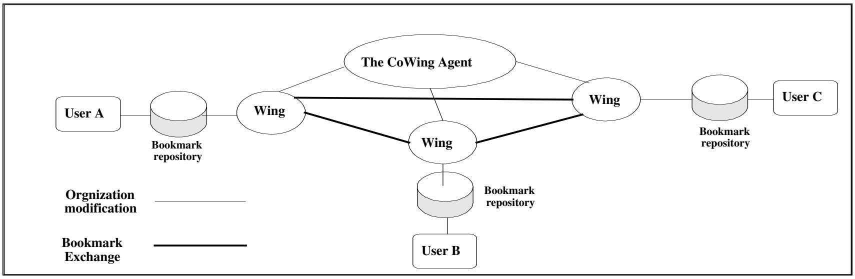


Fig. 1. The *CoWing* system architecture involving three users: A, B and C.

The system is composed of a central *CoWing* agent and a *Wing* agent per registered user. The *CoWing* agent acts as *Wing* agent registry. It provides *Wings* with each other addresses. In addition it provides *Wing* agents with a description of the users organization hierarchy (see section 2.3). A *Wing* agent performs two main tasks: *Assisting* the user in classifying new bookmarks in her/his own bookmark repository and *recommending* the user with bookmarks that have been added by other users, and which are *relevant* to the him. Each user manages her/his own hierarchy of bookmarks just as in single-user settings. However, users are required to set access rules that define which personal bookmarks or bookmark folders to share with whom. An easy to use role-based access control system is provided for that purpose. A *WING* agent observes the associated user behavior in order to *learn* how the user organizes her/his bookmarks. An unsupervised classifier system is used for this purpose (see section 2.2). *Wing* agents exchange bookmark folders according to a predefined protocol and each agent uses its own classification knowledge in order to compute correlation between local folders and received ones (see section 2.3).

2.2 Learning to Classify

Each *WING* agent uses a case-based reasoning (CBR) classifier in order to learn the user's bookmark classification strategy [12]. CBR is a problem solving methodology that is based on reusing past experiences for solving problems in order to solve new problems. A case is classically composed of two parts the problem part and the solution part. To solve a new problem the system retrieves from its *memory* (called also the case base) all cases that the problem part is *similar* to the problem to solve. Solutions proposed by retrieved cases can be adapted to propose a solution to the new problem. In our application, the problem part of a case is composed of set of the attributes of a bookmark, the solution part is the folder identifier in which the bookmark is filed by the user.

The used classifier memory model, called *PROBIS*, is based on the integration of a prototype-based neural network and a flat memory devised into many groups, each of them is represented by a prototype [12]. *PROBIS* contains two memory levels, the first level contains prototypes and the second one contains examples. The first memory level is composed of the hidden layer of the prototype-based neural network. A prototype is characterised by :

1. The prototype's co-ordinates in the m-dimensional space (each dimension corresponding to one parameter), these co-ordinates are the *centre* of the prototype.
2. The prototype's influence region, which is determined, by the region of the space containing all the examples represented by this prototype.
3. The class to which belongs the prototype (i.e. a bookmark folder)

The second memory level is a simple flat memory in which examples are organised into different zones of similar examples. These two levels are linked together, so that a memory zone is associated with each prototype. The memory zone contains all examples belonging to this prototype. A special memory zone is reserved for atypical examples. These are examples that do not belong to any prototype. The classifier system operates either in *learning* mode or in *classification* mode. The system can switch from one mode to another at any moment. Before the *first learning phase*, the system contains neither prototypes nor zones of examples. Examples for training are placed initially in the atypical zone. Prototypes and associated zones are then automatically constructed. An incremental prototype-based neural network is used to construct the upper memory level. Particular and isolated examples are kept in the atypical zone whereas typical examples are transferred to the relevant typical zones. This memory organisation helps to accelerate the classification task as well as to increase the system *generalisation capabilities*. In addition adding a new example is a simple task, the example is added to the appropriate memory zone and the associated prototype is modified. The learning procedure is the following:

1. If the new example does not belong to any of the existing prototypes, a new prototype is created. This operation is accomplished by adding a new hidden unit to the neural network. The co-ordinates of this prototype and the *radius* of the influence region is initialised to a maximal value (this is a system parameter). A new memory zone is also created and linked to the prototype. The new example is added to the new memory zone.
2. If the new example belongs to a prototype whose class value is the same as the example, the example is added to the associated zone of the second level memory. The prototype co-ordinates are modified to fit better the new.

2.3 Leaning to Recommend

The bookmark recommendation computation is performed as follows. Each *WING* agent maintains locally two data structures: an *agenda* and a *folder correlation matrix (FCM)*. The agenda is a dictionary structure where keys represent identifiers of peer *WING* agents to contact and values are next contact dates. Hence *Agenda[i]* gives the next contact date with agent *i*. The *FCM* is a *mXn* matrix where *m* is the number of folders in the local repository and *n* the number of peer agents known to the local agent. An entry *FCM[i, j]* is a couple $<f^{j}_{k}, cor_{ij}>$ where f^{j}_{k} is a folder identifier maintained by user u_j and cor_{ij} is the correlation degree between the folder f^{j}_{k} and the folder f^{i}_{k} maintained by local agent. Correlation between two folders f_1 and f_2 is given by the number of bookmarks contained in folder f_2 that are classified in folder f_1 divided by the total number of bookmarks in f_2. In the *FCM* matrix, an entry $FCM[i,j]= <f^{j}_{k}, cor_{ij}>$ is computed by taking the folder f_k from the agent *j* repository that have the *maximum* correlation value with folder *i* belonging to the local repository. Given a *WING* agent A, the bookmark recommendation process is made by executing the following algorithm:

```
1   For each B agent in Agenda do
2     If Agenda[B] is over then         ; it is time to contact
3       send B a bookmark request       ; the B agent
4       receive from B: V and ND        ; V is A's view on B repository
5       Agenda[B]= ND;                  ; ND the next contact date
6       For each f in V                 ; f folders in view V
7           <i,c>=computeCorrelation(f) ; i is the local folder with
8           If FCM[i,B].cor < c then    ; highest correlation with f
9               FCM[i,B]= <f,c>         ; c is correlation of i and f.
10      If FCM[i,B].cor > δ then        ; δ is a minimum correlation
           recommend to add bookmarks   ; threshold
           in f to the local folder i
```

The function *computeCorrelation* (line 7 in above algorithm) finds the folder *i* in the local repository that have the highest correlation value with a folder *f* as defined above. The function proceeds as follows. For each bookmark b_i in f the local neural/CBR classifier is applied. For each bookmark, the classifier responds by the identifier of a local folder. The folder that has been selected the most will be the returned folder. Notice that the correlation relation is not symmetric since correlation is computed by using local classifiers (the classifier is different from one agent to another) and by using information contained in the local agent view on the repository of the other agent.

3 Related Work

Few systems are proposed in the literature to cope with the problem of collaborative bookmark management. Almost all-commercial systems are based on implementing a central shared URL repository that allows users to store and retrieve URLs. Examples of shared bookmark systems are the *GAB* system [14], *KnowledgePump* [7], *Pharos* [3]. The GAB system offers a service that allows merging different user bookmark repository in a virtual centralized bookmark. However no recommendation mechanism is implemented. It is up to the users to navigate in the merged repository to find bookmarks they are interested in. A comparable approach is also implemented in the *PowerBookmarks* systems [13]. Both *KnoweldgePump* and *Pharos* provide users with the possibility to share a centralized bookmark repository. Both systems provide also customization service in order to recommend users with bookmarks that are more interesting for them in given folder. Recommendation computation is made by applying a collaborative filtering mechanism that is based on matching the characteristics of bookmarks added and accessed by each user. Most similar to our work is the *RAAP* system [5]. In *RAAP* the system also learns by using a classical classifier how users classify bookmarks and use this information to recommend people with new bookmarks. However, *RAAP* has the disadvantage of being built on a centralized repository. It provides a poor access control model. Related also to our work is the *Yenta* system [6]. *Yenta* is presented by its authors as a matchmaking system. It aims at discovering matchmaking between people based on comparing shared interests. The principal of *Yenta* could be easily applied to built a collaborative bookmark system. The accent is put on distributing the computation of the matchmaking function.

4 Conclusion

In this paper we have presented *CoWing*: a new full distributed collaborative bookmark management system. The *CoWing* system addresses mainly the resources discovery problem. It provides a mean that allow users to share their bookmarks, in a personalized way without asking users to do extra task except for defining others access control on their own repositories. Experiments made on *synthetic data* show that our approach is valid. However, we believe that some enhancements should in order to make the system operational in real work settings. One important issue concerns the cold start problem [8, 14]. The applied recommendation computation approach makes the hypothesis that users have organized their bookmarks in a hierarchy of folders. Each folder has some semantic sense. While lots of users do use hierarchical bookmark structures, some still using flat organization structures [2]. Another related problem is the witnessed low overlapping between different user bookmark repository [11].

References

1. D. Abrams, R. Baecker, and M. Chignell. (1998). Information Archiving with Bookmarks: Personal Web space Construction and Organization. *In Proceedings of ACM Conference on Human Computer Interactions (CHI'98),* Los Anglos, 18-23 April pp. 41-48.
2. Abrams, D. (1997) *Human Factors of Personal Web Information Spaces*. MS Thesis, Department of Computer Sciences, University of Toronto, 1997.
3. Bouthors V., and Dedieu O. (1999). Pharos, a Collaborative Infrastructure for Web Knowledge Sharing. *In Proceedings of the third European Conference On Research and Advanced Technology for Digital Libraries (ECDL'99)* Abiteboul S., and Vercoustre A. Eds), LNCS No 1696, Paris September, 1999, pp. 215-233
4. A. Cockburn, B. McKenzie. What Do Web Users Do? An Empirical Analysis of Web Use. In International Journal on Human-Computer Studies, 2000.
5. J. Delgado, N. Ishii and T. Ura. Intelligent Collaborative Information Retrieval. In proceedings of Progress in Artificial Intelligence, IBERAMIA'98, LNAI 1484, 1998
6. Foner, L.N. (1999) Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System, Ph.D. Thesis, MIT June 1999.
7. Glance, N., Arregui, D., and Dardenne M. (1999) Making Recommender Systems Work for Organizations. *In Proceedings of PAAM'99*, London April 1999.
8. R. Kanawati, M. Malek. Informing the design of shared bookmark systems. In proceedings of RIAO'2000, Paris, 2000.
9. R. M. Keller, S. R. Wolf; J. R. Chen, J. L. Rabinowitz and N. Mathe. A Bookmaking Service for Organizing and Sharing URLs. In proceedings of the 6th International Conference on the World Wide Web, Santa Clara, CA, April 1997.
10. W.S. Li, Q. Vu, D. Agrawal, Y. Hara and H. Takano. PowerBookmarks: A system for Personalizable Web Information Organization, Sharing and Management. *In proceedings of the 8th International World Wide Web Conference (WWW'8),* Toronto, Canada, 1999.
11. Maarek, Y.S., Ben Shaul, I.Z. (1996), Automatically Organizing Bookmarks per Contents. *In Proceedings of the 5th International World Wide Web Conference*, Paris, 6-8 May.
12. M. Malek. Hybrid approaches Integrating Neural Networks and case based reasoning: from Loosely Coupled to Tightly Coupled Models. In K.P. Sankar, S. D. Tharam and S. Y. Daniel (Eds) Soft Computing in Case-based Reasoning. Spinger, 2000 pp.73-94
13. Mathe, N. and Chen, J. R. (1998); Organizing and Sharing Information on the World-Wide Web using a Multi-agent System; *In proceedings of ED-MEDIA'98 Conference on Educational Multimedia and Hypermedia,* Freiburg, Germany, June 1998.
14. Wittenburg, K., Das, D., Hill W., and Stead L. (1997) Group Asynchronous browsing on the World Wide Web. *In proceedings of the 6th International Conference on the World Wide Web (WWW'6)*, 1997

Context Aware Agents for Personal Information Services

Enric Plaza and Josep-Lluís Arcos

IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia (Spain)
Vox: +34-93-5809570, Fax: +34-93-5809661
{enric,arcos}@iiia.csic.es
http://www.iiia.csic.es

Abstract. We present a society of personal information agents that work for a community of users and that are aware of the physical and social context of their users. We show how context-awareness is a feature that allows the agents to improve their performance when they work with limited resources in information spaces with a large amount of information. The use of context information allows the agents to focus their information search and, as a result of this, increase the quantity and quality of information delivered to the user. Moreover, we propose an implemented agent architecture with context-aware capabilities. We discuss this architecture in detail, focusing on the capability of exploiting the windows of opportunity provided by the awareness of the users' activity. The agents use these windows of opportunity to furnish the user with information and advice in the situation where it can be most useful. In particular, we show how context-aware information agents can assist a community of attendees to big conferences and fairs. In this application, an information agent gathers relevant information based on a model of specific interests of a user. Given the multiplicity of interesting events, and their distribution in time and space, an information agent has to deliver the gathered information in a few hours and comply to the schedule constraints. Finally we report some experimentation results to illustrate how context-awareness improve the service a society of information agents provides to a community of users in the conference application.

1 Introduction

The goal of personal information agents (PIAs) is to gather relevant information based on a model of the specific interests of a user. The current research in information agents covers several approaches such as cooperative and non-cooperative information agents, information agents with adaptation capabilities, mobile agents, or information agents for e-commerce [4]. Our research is focused in a society of personal information agents that interact among them for pursuing user interests. That is, we are using a dialogical approach for modeling the information gathering process. Moreover, we are interested in personal information agents that work in environments with a large amount of information and

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 44–55, 2001.

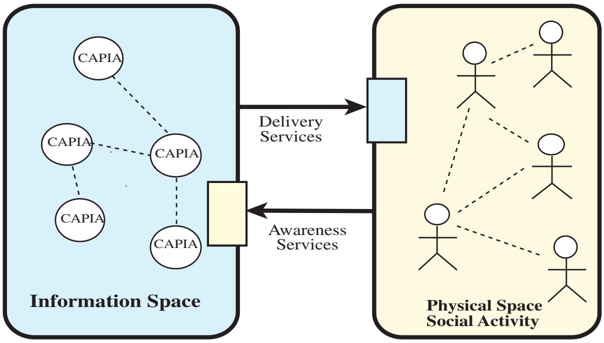


Fig. 1. A schema of the dual space of information agents and human users with the mediation services between them.

limited resources for dealing with this information. Specifically, in those environments the PIAs are not able to cover all the search space and the design of search strategies are required.

Our proposal is that personal information agents should be able not only of searching relevant information and of making contact with other interesting agents on behalf of the user; PIAs should be aware, as far as possible, of the physical and social context of the user in order to focus their information search and provide to the users information in the place where and at the time when that information is more relevant and useful. In this paper we present context-aware information agents that function in a particular physical and social setting, that of persons attending to conferences and fairs. Limiting our research effort to this specific context allows us 1) to design concrete PIAs that help users in some tasks they typically perform in conferences and fairs, and 2) develop context-awareness mechanisms that support those tasks.

In the next section we present the issues that have to be addressed in order to develop a society of interacting information agents aware of the physical and social context of their users. In Section 3 we introduce a specific application where users participate in a conference and a collection of PIAs, with a (partial) awareness of the users' physical and social context, provide information to the users. In Section 4 we present an implemented agent architecture with context-aware capabilities. In Section 5 we report the experiments performed with our agent architecture in the conference application. Finally, in Section 6 we compare our work with existing literature and summarize our proposal.

2 A Framework for Context-Aware Agents

Our framework is composed by a collection of context-aware personal information agents (CAPIAs) working in an information space and a collection of human users interacting in a same physical space. A useful way to visualize this distinction is the *dual space* schema depicted in Figure 1. Human users, on the right hand of Figure 1, are in a location, interacting with other persons (that might be users or not) in the context of social activities. Information agents, on the left hand of Figure 1, inhabit an information space where they interact with other agents and gather information in the interest of the users.

Moreover, we have *mediation services* connecting the information space of agents and the physical space of human users. Specifically, we are currently using two mediation services, namely an *awareness service* and a *delivery service* (see Figure 1).

2.1 Awareness and Delivery Services

The *awareness service* takes charge of pushing information from the physical space to the information space. Specifically, the awareness service provides to CAPIAs a real-time information about the physical location movements of users. The specific data provided depends on the particular sensors available in the awareness service for a particular application. For instance, in the conference center application the awareness service provides a real-time tracking of attendees location as well as the group of other attendees nearby a given attendee—see in Section 3 the features of the awareness service in the Comris Conference Center application.

Concerning the *delivery service*, it offers mediation and brokerage capabilities (subscribed by the human users) for delivering information from the information space to the physical space. Specifically, the delivery service provides the channels for delivering the information gathered by the CAPIAs to their corresponding users. For instance, in the conference center application the delivery service allows to send information as audio output by means of a wearable computer and HTML pages by means of screen terminals scattered through the conference building.

2.2 Agents Requirements

The society of agents has to be able to communicate using a common ontology for a specific application, and they have to share a collection of interaction protocols appropriate for that application. Our approach is to use the notion of *agent-mediated institution* [6] to specify the ontology and interaction protocols to be used by a society of agents for a particular application.

In addition to support the ontology and interaction protocols of an agent-mediated institution the agents should be able to manage with context awareness information. That is to say, a context-aware agent should be able to react dynamically when a new physical context information is received from the awareness service. Moreover, since the future physical and social context of the user is not known, a desired feature of CAPIAs is the capability of gathering information that may become relevant in a future context. For instance, in the conference center application, when an attendee is at a specific exhibition zone the CAPIAs use the knowledge provided by the conference about the physical distribution of booths for trying to anticipate the next movement of the attendee.

In our framework, context-aware personal information agents (CAPIA) are based on the distinction between two kinds of information valuation, namely *interestingness* and *relevance*. Information *interestingness* measures the intersection of a given information with the user model a CAPIA has for the tasks it is charged with. That is, *interestingness* : $\mathit{Info} \times U_M \mapsto e_I$ where Info is a given information; U_M is the user model; and e_I is the estimation of the interest that the user has in Info.

However, depending on the physical and social context of the user and on the time some information may be more or less *relevant* for the user on each particular point of time. Information *relevance* measures this intersection of a given information with the time and the context of the user. That is, *relevance* : $\mathit{Info} \times \mathit{time} \times \mathit{UC} \mapsto e_R$ where Info is a given information; U_C is the user context; and e_R is the estimation of the relevance of Info in U_C. For instance, in the conference application when an attendee is nearby to an exhibition booth, the information related to the booth is estimated as more relevant. Another example of increase of relevance is when a conference event is close to start: a CAPIA has a time constraint for deciding if that event is useful for the user interests.

3 The Comris Conference Center

This section introduces the framework of context-aware information agents in a particular application, that is to say in the physical location and the social activity context of a Conference Center in the Comris[1] project [8]. We view the Conference Center (CC) as an agent-mediated institution where a society of information agents work for, and are aware of, the attendees of a conference [1, 7]. The ontology of the CC institution defines the conference activities that take place in the CC. Examples of conference activities are exhibition booths and demo events, plenary and panel sessions, etc. The ontology also defines the roles that a person takes in different locations while performing different activities, e.g. speaker, session chair, attendee, organization staff, etc. Other important elements defined by the CC ontology are the different locations of the conference such as the exhibition areas, the conference rooms, and the public areas—i.e. halls, cafeterias, and restaurants. This information is used by the agents for reasoning about the movements of users in the conference. The schedule of conference events is also defined in the CC ontology.

Finally, the CC ontology supports the definition by each user of the "instruction set" that her CAPIA should follow. The instruction set is entered by the conference attendee using a WWW browser while registering and basically includes i) an interest profile (specifying the topics with weights the attendee is interested in), ii) those tasks the user commissions the PIA to do in her behalf

[1] COMRIS stands for Co-Habited Mixed-Reality Information Spaces. More information is available at URL `<http://arti.vub.ac.be/~comris/>`.

(e.g. if she is interested or not in making appointments); and iii) the delivery modes that the CAPIA will use to communicate with her.

We implemented two types of CAPIAs in the conference center application: CAPIAs representing interests of attendees and CAPIA advertisers. There is a CAPIA for each attendee, a CAPIA advertiser for each exhibition booth, and a CAPIA advertiser for each paper session. The goal of CAPIA advertisers is convince people for attending to the conference event they are representing.

3.1 Delivery Service

The delivery service in Comris allows the users to receive information in two ways: by means of a wearable computer with text and audio output and by screen terminals scattered through the Conference Center. The wearable computer is used to convey short messages that are relevant for the user with respect to her current physical and social surroundings. The user can walk to a terminal if she wishes to have more information about this message or other recent messages she has received. When the user approaches a screen the wearable computer detects this terminal's identifier, and then it sends this identifier to the user's CAPIA. Once the CAPIA is aware of this situation, the agent sends to that screen the report of the performed tasks and the report of ongoing tasks.

The delivery service comprises several components. The first component is the natural language generation (NLG) component. The NLG component receives the message sent by a CAPIA and generates an english sentence explaining the message content and taking into account the current attendee context and the sentences previously generated. Then, when the message has to be delivered as audio, the sentence structure is sent to a speech synthesis component that produces the actual audio heard by the user. Similarly, there are components that transform CAPIA's messages into HTML or VRML in order to be delivered to the screen terminals.

3.2 Awareness Service

The awareness service keeps track of the whereabouts of the attendees in the Conference Center. In the Comris CC the detection devices are a network of infrared beacons (marking the different rooms, places and locations in the CC) and the wearable computers the attendees carry. The Comris wearable computer detects the infrared beacons and thus informs the awareness service of the location of its user. Moreover, the wearable device possesses an infrared beacon, allowing the detection of other persons, wearing a parrot, located nearby. In order to have access to this information, each CAPIA in the information space "subscribes" its user to the awareness service. As a result, the CAPIA receives messages about the changes in location of that person and a list of other people close to that person. When the CAPIA interacts with other CAPIAs (representing other conference attendees), and decides that those CAPIAs are interesting persons, subscribes those persons to the awareness service. Consequently, the CAPIA is aware of the location of the most interesting persons for its user, and detects for instance when one of these persons is in the same location as the user—a most relevant situation to push to its user the information concerning that person that is interesting *and* nearby.

Tasks. The tasks that the COMRIS Conference Center supports are the core of the activity in the CAPIAs. It is important to remark here that, in order to perform these tasks, the information agents use *both* the CC ontology and the awareness service to infer the situation of the user. That is to say, knowing that the user is in a particular place, the current time, and the activity scheduled by the Conference for that place at that time, the information agent can infer the social activity in which the user is involved.

We will briefly summarize the tasks performed by COMRIS CAPIAs and the scenes they are involved in.

• *Information Gathering*: is responsible of establishing initial conversations with other CAPIAs for estimating the interestingness of the attendees or conference events they represent. We say that the information gathering task constructs the *interest landscape* of a given attendee. The interest landscape holds all the information considered as useful for the interest of the attendee and is used and refined in the other tasks. In CAPIA advertisers, this task has been specialized for attracting persons that might be interested in the conference events (exhibition booths or conference sessions) they represent.

• *Appointment Proposal*: in this task, using the interest landscape, the CAPIAs try to arrange an appointment between two attendees. First, CAPIAs negotiate a set of common topics for discussion (the meeting content). When they reach an agreement, CAPIAs negotiate on the appropriate meeting schedule.

• *Proximity Alert*: in this task an attendee is informed that she is physically near to another person with similar interests —or near an exhibition booth or a thematic session with similar topics.

• *Commitment Reminder*: this task is responsible of checking if attendees are aware of their commitments. The CAPIA uses context to determine that the user may be unaware of a commitment, e.g. if she is not near the location of an appointment (or a bus scheduled to leave) a few minutes before.

For each task several *activities* are launched in the CAPIA. For instance, when an agent in COMRIS is discussing about appointments with several CAPIAs, each thread of interaction is managed by a distinct activity.

4 Personal Information Agents

The architecture of CAPIAs has to fulfill adequately the requirements posed by context awareness. Essentially, the fact that an agent is aware of the context implies that i) the changes in the context cause immediate changes in the internal inference, and ii) some activities the CAPIA is pursuing cannot be completed until a satisfactory context is reached. On the other hand, it's not possible for an agent to wait until a context is reached and then start the relevant activity. Since the CAPIA needs to perform some interaction with other agents, and those agents may delay too much their answers, the agent could have to wait too long and see the context change again, losing its window of opportunity. Because of that, the CAPIAs has to anticipate the work on activities that can arise as future candidates. Thus, the architecture of CAPIAs has been designed for dealing with a high number of concurrently competing activities focusing on the most promising activities at any point in time. The architecture of CAPIAs consists on five components, as shown in Figure 2, and are detailed below.

The *Communication Component* is responsible of communicating the agent with the society of agents. Thus, the communication component registers the agent into the conference application and manages the incoming and outgoing messages. Messages incoming from the awareness service will be consumed by the awareness component. Messages incoming from the delivery service will be consumed by the delivery component. Messages incoming from other CAPIAs or from the conference will be consumed by the activities in the decision component. Moreover, the communication component is the responsible of translating the internal representation structures into the communication language defined by the institution and vice versa—in COMRIS the communication component translates from/to the XML-based FIPA-compatible language defined for message content to/from the internal frame-based representation.

The *Awareness Component* is responsible of processing the messages incoming from the awareness service and provide the *context model* to the agent. The context model holds the information provided by the awareness service and information elaborated by the agent from this basic context information. For instance, in COMRIS the physical location of the attendee and the current time is used, with the schedule of the conference, for inferring the social activity of the user. Moreover, the context model stores the history of context changes. This context trace is used in COMRIS agents for changing dynamically the relevance of an information and for generating the information to be delivered to the attendee. For instance, the second time an attendee coincides in a paper presentation her relevance is increased.

The *Delivery Component* is responsible of generating the information to be presented to the attendee in the format adequate to the functionalities provided

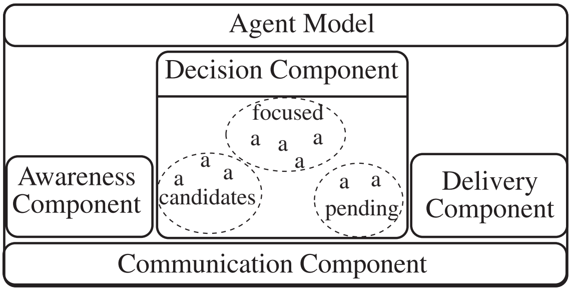


Fig. 2. The CAPIA architecture components.

by the delivery service. In COMRIS the users receive information in two ways: by means of a wearable computer and by screen terminals scattered through the Conference Center. These two complementary channels implies that the delivery component has to generate different information for each delivery channel. For the wearable output, it is important to generate short messages selecting the most relevant information. In contrast, in the screen terminals there is more room for reporting the agent activity and the main issue here is how to organize the information.

The *Agent Model* is the component that holds i) the attendee's interests and instructions and, ii) the interest landscape—i.e. the models about the other attendees. The interest landscape stores the profiles and interests of other attendees and a record summarizing the history of the interactions with their CAPIAs. In COMRIS the attendee's interests and instructions is a static information that can be partially tuned using the screen terminals. An attendee can inspect her interest landscape in the screen terminals and change the valuation performed by the CAPIA influencing the future behavior of the CAPIA. The models about the other attendees are built from the interaction with the other CAPIAs and are influenced by the decisions taken by the attendee when the attendee accepts or declines the suggestions received from the CAPIA.

The *Decision Component* is responsible of performing the tasks on behalf of attendee's interests. The Decision Component manages the collection of ongoing activities by dividing them in three groups (see Figure 2):

- *Focused* activities are those pursued in an interleaved mode. When a focused activity in order to proceed needs an answer from another CAPIA or requires a specific context, the activity is demoted to the *pending* group.
- *Pending* activities are those waiting for the answer of another CAPIA or waiting for a specific attendee's context. When the communication component receives a message for a given pending activity or the attendee's context satisfies the pending conditions of an activity, that activity is promoted to the *candidate* group.
- *Candidate* activities are those competing for promotion to the focused group. This group is composed by new activities (generated either internally or by foreign requests) and activities previously members of focused or pending groups.

component is to decide which candidate activities have to be promoted to the focused group and which focused activities are demoted to candidates. The decision component uses the context model provided by the awareness component to make those changes. For instance, in COMRIS we have associated to each activity an activation record ρ with five values $\rho = \langle T, S, e_I, e_R, e_C, a_V \rangle$ where T is the type of the activity (information gathering, appointment proposal, proximity alert, or commitment reminder); S is the starting time (the time when the activity started); e_I is the estimated interestingness of the information the activity is elaborating; e_R is the estimated context relevance of the information; e_C is the estimated confidence that the activity will succeed; and a_V is the activation value.

Table 1. Comparison of experimentation results with different interest intersection ratios.

	100 %		50 %		25 %		10 %	
	Tot.	%	Tot.	%	Tot.	%	Tot.	%
Without Context	1080	43.2	540.32	43.22	270.59	43.3	108.8	43.23
Context Awareness	1080	43.2	609.87	48.79	441.37	70.62	236.95	94.78

The estimated confidence e_C is a number assessing the likelihood of success in an activity (for instance, for an appointment proposal activity the confidence that an agreement can be reached). e_C is calculated using features such as the time delay in receiving answers or the flexibility in accepting counter-proposals. The activation value a_V of a given activity is a number that aggregates 1) the interestingness of the information e_I with 2) the relevance of the information e_R (given the current context and the activity type T) and 3) the estimated confidence e_C that the activity will succeed. The activation values of all ongoing activities are used by the decision component, with a threshold mechanism, for determining which activities are promoted to focused and which other are demoted to candidates.

5 Experimentation Results

We have performed several experiments for evaluating how context awareness improve the information provided by the CAPIAs. For our experimentation purpose we have used a context simulator that provides context information about attendees movements without requiring the existence of attendees physically walking around a conference building. The context simulator allows us to generate different sets of context traces varying several testing conditions.

We performed experiments with populations of 2.000 attendees participating in a conference. The conference was designed with 420 different locations: 400 exhibition booths distributed along 8 exhibition areas; 10 seminar rooms with 10 parallel presentations in each of them; and 10 public areas such as halls, cafeterias, and restaurants. The duration of the conference was 6 hours and the attendees remained in the same physical place from 5 to 15 minutes (with an average of 10 minutes). Notice that attendees visit about 40 different locations as average. Each attendee participating in the conference was represented by one CAPIA—i.e. 2.000 CAPIAs representing attendees. Moreover, a CAPIA advertiser was assigned to each exhibition booth and each paper session for trying to attract people to the event the agent is representing—i.e. 410 CAPIAs more. Finally, we also restricted the amount of information that each CAPIA is able to manage: 3 different information sources (conference events or interactions with other CAPIAs) per minute. This constraint limited the amount of different information an agent handled during the 6 hours of the conference experiments to 1080, i.e. around the 40 % of the total amount.

Given the previous conference scenario we performed experiments using four experimentation settings with different interests intersection between attendees.

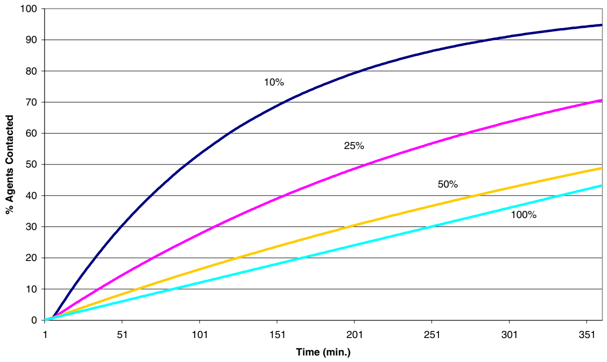


Fig. 3. Experimentation Results.

A first setting with 100 % of interest intersection (everybody interested in the whole conference), a second setting with 50 % of interest intersection, a third setting with 25 % of interest intersection, and the last setting with 10 % of interest intersection (attendees only interested in the 10 % of the conference).

The experiments were performed following two modes: with and without context information for guiding the CAPIAs decisions.

The results of the experiments shown that without the context information the CAPIAs are able to identify and deliver to the attendees only the 43 %, in average, of the whole conference information interesting by users (see first row of Table 1). Notice that the absolute number of final messages delivered to users depends on the experimentation setting but the ratio with respect to the potential interesting information is the same.

The results of the experiments performed using the awareness service shown that the CAPIAs improved their results (see second row of Table 1). When the interest intersection between attendees is 100 % there is no difference with respect to not using context information. The explanation is trivial because when everybody is interested in the whole conference, the context information is not relevant. Nevertheless, when the interest intersection of attendees decreases the use of context information arises as a crucial capability of the agents for focusing the information search (see comparison in Table 1).

A detailed analysis of the performance of the CAPIAs along the conference time (see Figure 3) shows that as soon an attendee went to a place interesting for her, the CAPIAs started to exploit the context information allowing a fast identification of interesting information. When the number of delivered information increases the rate of new information decreases (see in Figure 3 how the slope of the curves decreases). This phenomenon can be explained by the fact that the value of context information decreases because many of the attendees detected by the awareness service have already been reported in previous occasions by the CAPIA.

Another important experimentation result was the analysis of when the information was delivered to the users. It is clear that using the awareness service the CAPIAs deliver more messages to their users —but an important issue is how these delivered messages are distributed along the conference period. An analysis of the first mode (without using context information) shows that the distribution of messages is uniform along the conference period. An analysis of the second mode shows that when CAPIAs use the awareness information for focusing on the most promising activities, they can take advantage of the opportunities generated by the attendee's context. Then, the messages delivered to attendees are better distributed along the conference period. This is because in the second mode CAPIAs use awareness information to guide its evaluation of the most promising activities. For instance, if a CAPIA is aware that a certain attendee is nearby its user, the agent increases the activation value aV of the activities related to that attendee. Therefore the CAPIA has more chances to exploit windows of opportunity offered by its user's activities than an agent working on the first mode.

Summarizing, the experiments shown that the use of context awareness allows CAPIAs to deal with high amount of potential information without degrading significantly their competence. When an agent fully uses the awareness service to guide its work, as it does in the second mode, it's less affected by size problems since it can dynamically adapt its focus to those CAPIAs that are relevant given the concrete surroundings of its user's activity.

6 Discussion

The literature on information agents is increasingly wide in number and scope. Recent books on intelligent information agents (like [5] and [4]) show a variety of research addressing societies of cooperating information agents, adaptive information agents, mobile agents, and applications to e-commerce. The focus of this paper is on context-awareness, a new feature desirable for certain applications of information agents. It is novel in the sense that the tasks that a CAPIA pursues or abandons, the information that eventually gathers, and the specific information that the user of such an agent receives, all these activities are biased and mediated by the perception that the agent has on the physical and social activity of the user. We have introduced a framework for analyzing the relationship between the information space inhabited by CAPIAs and the physical space inhabited by users. In this framework we define two mediation services between physical and information spaces, namely delivery and awareness services.

The research on context awareness is also increasing and is encouraged by the new possibilities that offers the mobile communications. The paper from Dey and Abowd [2] presents an overview about context and context-aware applications. The same authors presented a work regarding to a conference assistant [3].

Nevertheless, the focus of their research is on the relationship between context-awareness and wearable computing.

We have also presented the COMRIS application where context-awareness of a society of agents is supported to improve the information received by attendees to a conference or a fair. Concerning COMRIS we have focused on the society of information agents and we have only described other parts of the application as needed for understanding the requirements posed on CAPIAs and the tasks that those agents perform on the Conference Center application. See the COMRIS website at `http://www.iiia.csic.es/Projects/comris/` for more information. We have proposed and implemented an architecture for context-aware agents, emphasizing the need for such an architecture to be able to react to a changing surroundings, to exploit the windows of opportunity offered by the users' moving throw changing contexts, and to push the information they have gathered in situations where the users can consider that meaningful and relevant. Finally, the scenario and experiments have been used to illustrate in more detail how is the relationship between a society of agents and a community of users and why the ability of context-awareness is useful for situations as such of a Fair or a Conference Center.

Acknowledgements. The research reported in this paper is partly supported by the ESPRIT LTR 25500-COMRIS *Co-Habited Mixed-Reality Information Spaces* project.

References

1. J. L. Arcos and E. Plaza. Instituting interactions among agents. In *Interaction Agents Workshop*. AAAI Press, 1998.
2. A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. In *Proceedings of CHIA'00 workshop on Context-Awareness*, 2000.
3. A. K. Dey, D. Salber, G. D. Abowd, and M. Futawaka. The conference assistant: Combining context-awareness with wearable computing. In *Proceedings of the 3rd International Symposium on Wearable Computing ISWC'99*, 1999.
4. M. K. (Eds). *Intelligent Information Agents*. Springer Verlag, 1999.
5. M. Klusch, O. M. Shehory, and G. W. (Eds). *Cooperative Information Agents III*, volume 1652 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1999.
6. P. Noriega and C. Sierra. Towards a formal specification of complex social structures in multi-agent systems. Number 1624 in Lecture Notes in Artificial Intelligence, pages 284–300. Springer-Verlag, 1999.
7. E. Plaza, J. L. Arcos, P. Noriega, and C. Sierra. Competing agents in agent-mediated institutions. *Personal Technologies*, 2:212–220, 1998.
8. W. Van de Velde. Co-habited mixed reality. In *Proceedings of IJCAI'97 workshop on Social Interaction and Communityware*, 1997.
9. W. Van de Velde, S. Geldof, and R. Schrooten. Competition for attention. In M. P. Singh, A. Rao, and M. S. Woooldridge, editors, *Intelligent Agents IV*, number 1365 in Lecture Notes in Artificial Intelligence, pages 297–311. Springer-Verlag, 1998.

Data Warehouse Quality and Agent Technology

Matthias Jarke[1, 2]

[1] RWTH Aachen, Informatik V (Information Systems), Ahornstr. 55
52074 Aachen, Germany
jarke@informatik.rwth-aachen.de
[2] GMD-FIT Institute of Applied Information Technology, Sankt Augustin, Germany

Abstract. Based on a brief review of agent-based information brokering and data warehouse architectures, we discuss the importance of data quality in data warehousing and the role distributed and multi-faceted knowledge processing as offered by agent technologies can play in such settings. We present a comprehensive methodology for addressing these issues, developed in the European DWQ project, and report on some industrial experiences concerning the relevance of this methodology to large-scale data warehousing efforts.

1 Information Brokering and Data Warehouses

After the basic online transaction processing (OLTP) infrastructure is in place in many organizations, not least through standardized enterprise resource planning tools such as SAP/R3, the focus of interest is now broadening in at least three directions:

- to the integration of a *broader range of multimedia data sources* inside and outside the organization,
- to a *broader range of clients* with diverse interest and capability profiles as well as situational parameters, and
- to the conversion of the massive *experiential data* created by transaction processing into *knowledge* relevant for organizational learning and action.

A wide range of information flow logistics architectures is being proposed under labels such as supply chain management and business-to-business e-commerce. In such architectures, databases can be considered the short- and medium-term intermediate stores of information whereas data warehouses serve for long-term memory, knowledge creation and management. Information brokering is, then, a basic loading and unloading service between the storage and transportation of data. On the one hand, information brokering can thus be considered a subtask of data warehousing related to linking information sources to the data warehouse. However, one could also say that a data warehouse is a long-term information broker between the operational part and the reporting/planning part of a company, supporting the Controlling department. This choice also has profound implications for the – in my opinion still open – question what role agent technologies can play in advanced data warehousing.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 56–75, 2001.

1.1 Agents in Information Brokering

The role of agent-based technologies in information brokering has been discussed at least for a decade (e.g. [PLS92, JP98, PTU99]). Two cases can be distinguished: in *information retrieval*, the response to an information request consists of a set of documents (often annotated with relevance information), while in *information extraction* where actual data is provided using the original source documents only as evidences. There is by now fairly wide (though not universal) agreement that neither information extraction nor information retrieval can be handled adequately on a purely syntactic level [KS98]. At the semantic level, domain ontologies are broadly considered useful for the matching of terminologies between the sources, the broker, and the customer agents. At a pragmatic level, the de-contextualization of source information from the context of their creation and the re-contextualization of brokered information into the customer's context of use (e.g. work context, personal interests and competencies, device context and mobile-local context) is equally important.

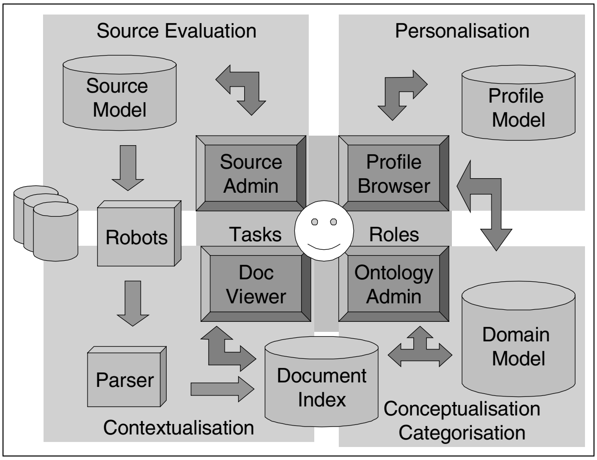


Fig. 1. Broker's Lounge [JaKN01]

As an example of how these ideas are applied in a commercial system, consider the basic architecture of the Broker's Lounge platform developed in GMD-FIT of which commercial spin-offs are used, e.g., for competition monitoring in the steel industry and for targeted access to research funding information. Broker's Lounge is a configurable environment whose information brokering processes can be composed from the components shown in figure 1 in many different ways [JKN01]. Basic structuring and detailed ontology are handled by tools for information categorization and conceptualization. The created concepts are used firstly to create parsers that index retrieved documents and allow visually attractive access to them, but also serve as the conceptual basis for search agents/internet robots who look for the relevant documents, or alert to changes in already found ones. On the customer side, an elaborate user model must be matched both against the viewing facilities and against the domain ontologies to ensure the personalization of information delivery.

The key success factor in such environments, at least according to our experiences, is a kind of light-weight approach where not too much investment in the mentioned tasks is needed before success beyond the quality of standard search engines is reached. In contrast, data warehouses – being a kernel corporate resource-- require a careful, long-term oriented design (but, of course, evolution of sources and adaptation to changing customer requests remain important as well).

1.2 Agents in Traditional Data Warehouse Architectures

The traditional data warehouse (DW) architecture, advocated both in research and in the commercial trade press, is shown in figure 2. Physically, a data warehouse system consists of databases (source databases, materialized views in the data warehouse), data transport agents that ship data from one database to another, and a repository which stores metadata about the system and its evolution. In this architecture [CDH+94, KB98], heterogeneous information sources are first made accessible in a uniform way through extraction mechanisms called *wrappers*, then *mediators* [WG95] take on the task of information integration and conflict resolution. The separation between wrapper and mediator agents is a considered design decision, reflecting the separation between service wrappers and request brokers in middleware systems such as CORBA.

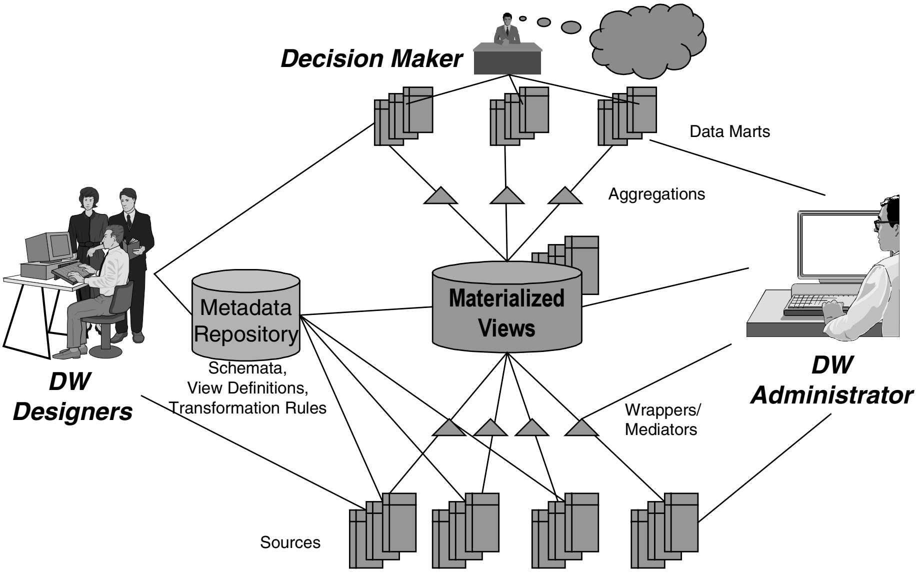
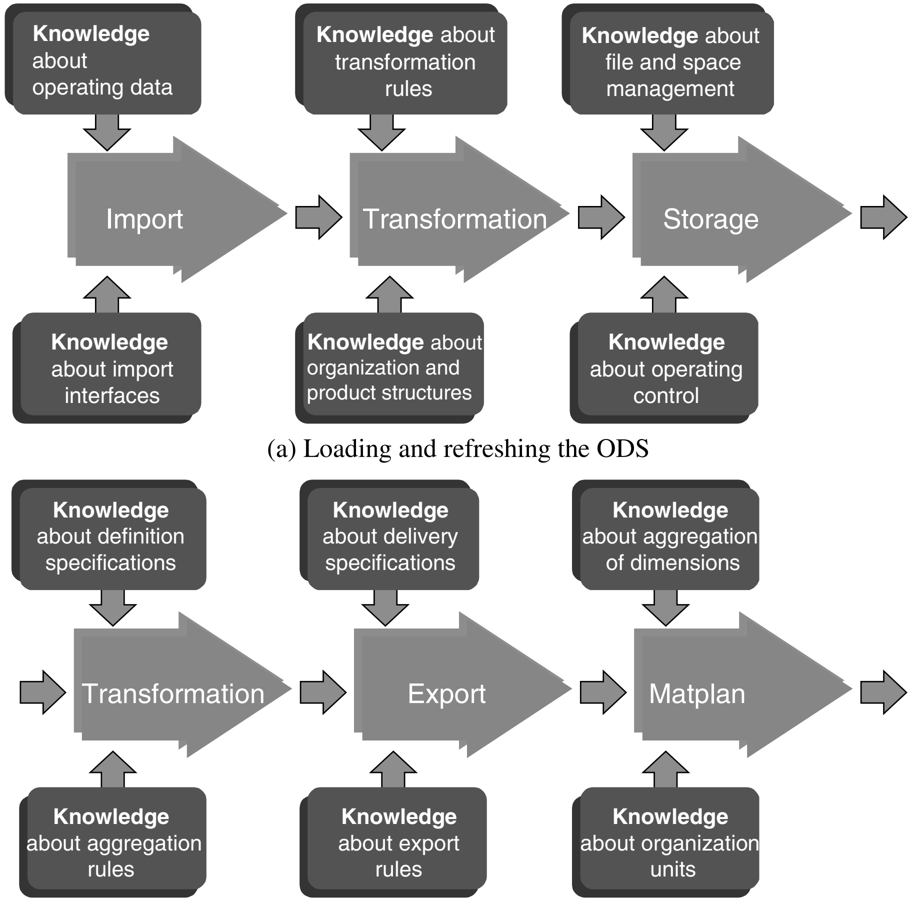


Fig. 2. Traditional data warehouse architecture

The resulting standardized and integrated data are stored as materialized views in the data warehouse. These base views (often called ODS or operational data store) are usually just slightly aggregated. To customize them for different analyst users, *data marts* with more aggregated data about specific domains of interest are constructed as second-level caches which are then accessed by data analysis tools ranging from query facilities through spreadsheet tools to full-fledged data mining systems.

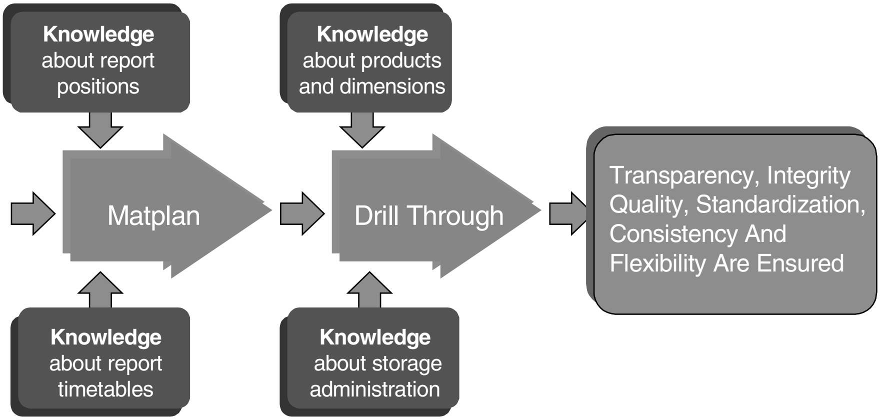
Almost all current research and practice understand a data warehouse architecture as a stepwise information flow from information sources through materialized views towards analyst clients, as shown in figure 2. For example, projects such as TSIMMIS [CDH+94] or Squirrel [Hull 97] all focus on the integration of heterogeneous data via wrappers and mediators, using different logical formalisms and technical implementation techniques. The Information Manifold project at ATT Research [LRO96] is the only one providing a conceptual domain model as a basis for integration.

Our key argument is that the architecture in figure 2 covers only partially the tasks faced in data warehousing and is therefore unable to even express, let alone support, a large number of important quality problems and management strategies followed in data warehouses. To illustrate this point, consider figure 3 which shows the host of knowledge sources used in the central data warehouse a large German bank has developed for its financial controlling environment (cf. [SBJ00] for details).


Knowledge about operating data
Knowledge about transformation rules
Knowledge about file and space management
Import
Transformation
Storage
Knowledge about import interfaces
Knowledge about organization and product structures
Knowledge about operating control
Knowledge about definition specifications
Knowledge about delivery specifications
Knowledge about aggregation of dimensions
Transformation
Export
Matplan
Knowledge about aggregation rules
Knowledge about export rules
Knowledge about organization units

(a) Loading and refreshing the ODS

(b) Preparing a host of specialized multi-dimensional models


Knowledge about report positions
Knowledge about products and dimensions
Matplan
Drill Through
Transparency, Integrity Quality, Standardization, Consistency And Flexibility Are Ensured
Knowledge about report timetables
Knowledge about storage administration

(c) Personalizing information delivery and access to background information

Fig. 3. Sixteen knowledge sources for a real-world data warehouse

Despite the fact, that this data warehouse talks “only” about financial figures, there is a host of semantic coherency questions to be solved between the different accounting definitions required by tax laws, stock exchanges, different financial products, and the like. At the same time, there are massive physical data integration problems to be solved by re-calculating roughly 40.000 multi-dimensional data cubes with 800 GB on a daily basis to have close to zero-latency information for top management. In light of such problems, many architectures discussed in the literature appear somewhat naive.

The key to solving these enormous problems in a flexible and evolvable manner is enriched metadata management, used by different kinds of interacting software components. In the following section, we shall present our approach how to organize this.

2 Metadata Structuring in the DWQ Approach

The main argument we wish to make is the need for a *conceptual enterprise perspective*. To explain, consider figure 4. In this figure, the flow of information in figure 2 is stylized on the right-hand side, whereas the process of creating and using the information is shown on the left.

Suppose an analyst wants to know something about the business -- the question mark in the figure. She does not have the time to observe the business directly but must rely on existing information gained by operational departments, and documented as a side effect of OLTP systems. This way of gathering information implies a bias which needs to be compensated when selecting OLTP data for uploading and cleaning into a DW where it is then further pre-processed and aggregated in data marts for certain analysis tasks. Considering the long path the data has taken, it is obvious that also the last step, the formulation of conceptually adequate queries and the conceptually adequate interpretation of the answers present a major problem to the analyst.

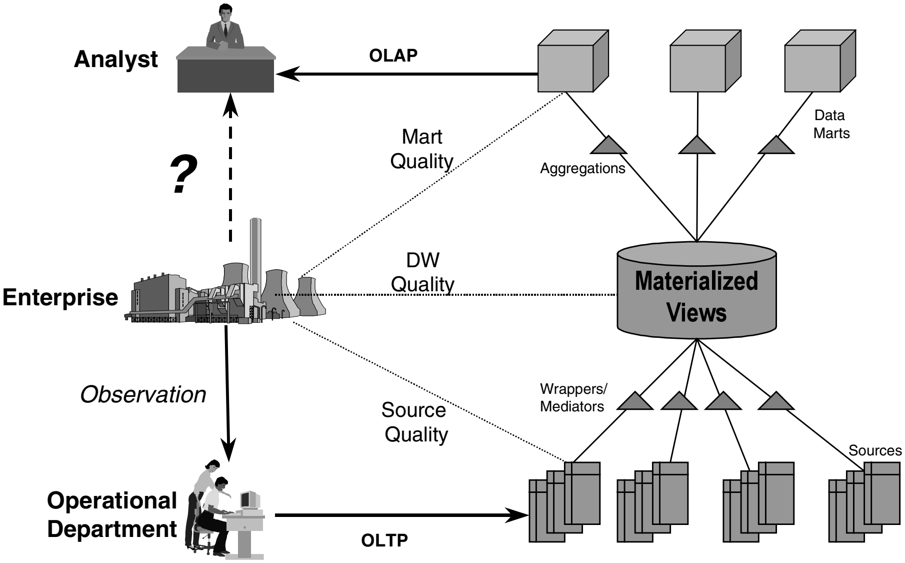


Fig. 4. An enterprise-oriented critique of the DW architecture

The traditional DW literature only covers two of the five steps in figure 4. Thus, it has no answers to typical practitioner questions such as "how come my operational departments put so much money in their data quality, and still the quality of my DW is terrible" (answer: the enterprise views of the operational departments are not easily compatible with each other or with the analysts view), or "what is the effort required to analyze problem X for which the DW currently offers no information" (could simply be a problem of wrong aggregation in the materialized views, could require access to not-yet-integrated OLTP sources, or even involve setting up new OLTP sensors in the organization).

An adequate answer to such questions requires an explicit model of the conceptual relationships between an enterprise model, the information captured by OLTP departments, and the OLAP clients whose task is the decision analysis. We have argued that a DW is a major investment undertaken for a particular business purpose. We therefore do not just introduce the enterprise model as a minor part of the environment, but demand that *all other models are defined as views on this enterprise model*. Perhaps surprisingly, even information source schemas define views on the enterprise model -- not vice versa as suggested by figure 2!

In [CDL01], the advantages of having an explicit conceptual perspectives are summarized as follows:

- *Declarative and system independent approach*: A conceptual level in the architecture for information integration is the obvious means for pursuing a declarative approach to information integration. As a consequence, all the advantages deriving from making various aspects of the system explicit are obtained. The conceptual level provides a system independent specification of the relationships between sources, and between sources and the enterprise model.

- *High level of abstraction in user interface*: One of the most tangible effects of conceptual modeling has been to break the barrier between user and system by providing a higher level of abstraction in the design. Moreover conceptual models are naturally expressed in graphical form, and graphical tools that adequately present the overall information scenario are key factors in user interfaces.
- *Incremental approach*: One criticism often raised to the declarative approaches to information integration is that it requires a reconciled view of the data, which can be very costly to obtain. However, having a conceptual level does not impose to fully develop it at once. Rather, one can incrementally add new sources or new elements therein, when they become available, or when needed, thus amortizing the cost of integration.
- *Documentation*: While in the procedural approach the information about the interrelationships between sources is hard-wired in the mediators, in a declarative approach it can be made explicit. The importance of this clearly emerges when looking at large organizations where the information about data is widespread into separate pieces of documentation that are often difficult to access and not necessarily conforming to common standards. Conceptual modeling for Information Integration can thus provide a common ground for the documentation of the enterprise data stores and can be seen as a formal specification for mediator design.
- *Maintenance*: Classical advantages of conceptual level in the design phase are advantages also in the maintenance phase of the Information Integration System (sources change, hence design never ends).

All the advantages outlined above can be obtained simply by having the appropriate linguistic (graphical) tools for expressing the conceptual model, as well as the mappings to the other components of the architecture.

Another set of features that one can obtain from the introduction of the conceptual level are related to the ability to *reason over the conceptual representation*. Such a possibility, which is fully supported by our approach, can be used in the accomplishment of several activities concerning both the design and the operation of the system. For example, one can use reasoning to check the conceptual representation for inconsistencies and redundancies; to maintain the system in response to changes in the information needs; to improve the overall performance of the system. In particular, we are pursuing the goal of characterizing the quality of data and to use such a characterization to improve quality of services by relying on reasoning support on the conceptual representation.

Our first step towards this goal was to capture the observations illustrated in figure 3 and abstracted in figure 4 in a metadata model – the schema of an enhanced meta database supporting our approach. Over the past few years, we have developed such a meta model in three steps, looking at the overall architecture (which perspectives and submodels are needed?), the question of how to integrate quality models with the architecture, and the question of representing data warehouse management processes over this architecture and quality model. In the following, we briefly discuss the first two of these. As for the processes, we do not discuss them at the meta level, but present a specific methodology in the next section.

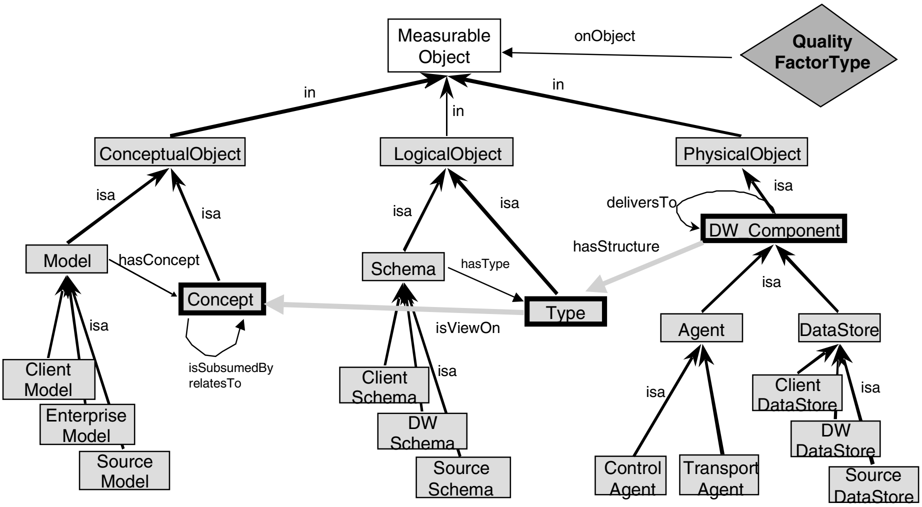


Fig. 5. Meta model of data warehouse architecture [JJQV98]

The architecture model shown in figure 5 distinguishes three perspectives:

- the already discussed conceptual perspective in which models of both client interests and available sources are linked to the enterprise model underlying the data warehouse
- the logical perspective in which the interlinking of data schemas, views and queries takes place, and
- the physical perspective which describes the (software) agents responsible for the physical transfer, transformation, and storage of the data.

Note that the logical perspective has been the one in which most database research has been invested, whereas the physical perspective is the one on which most commercial tools are focusing. Obviously, there are close relationships between the three perspectives which will be further discussed in the methodology section. Formally, these relationships are stored in the meta database and can be reasoned about by metadata query facilities and, if necessary, by more powerful description logic reasoners.

The different perspectives are closely related to specific aspects of data warehouse quality. A detailed discussion of data quality issues is beyond the scope of this paper (cf. [Wang00] for an overview). Figure 6 may give an impression how the main quality factors identified in the literature can be associated to the objects and processes of the conceptual, logical, and physical perspective. Looking back to the example in figure 3, the reader can now rather easily figure out how concern for certain potential quality problems has influenced the choice of knowledge bases in the meta model accompanying the bank's data warehouse process.

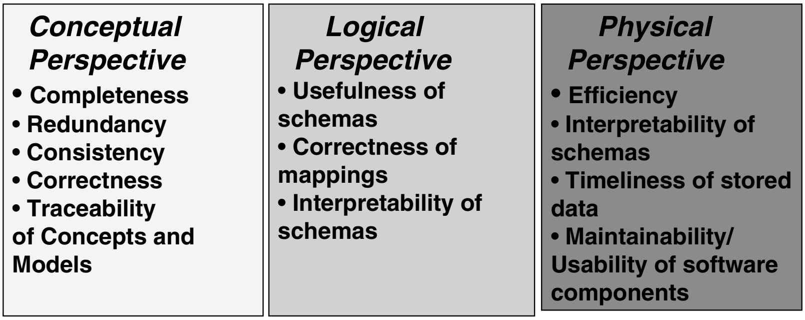

Conceptual Perspective
• Completeness
• Redundancy
• Consistency
• Correctness
• Traceability of Concepts and Models
Logical Perspective
• Usefulness of schemas
• Correctness of mappings
• Interpretability of schemas
Physical Perspective
• Efficiency
• Interpretability of schemas
• Timeliness of stored data
• Maintainability/ Usability of software components

Fig. 6. Data quality factors by meta model perspective

In designing and operating a data warehouse, such quality factors are used in at least two different ways: firstly, to monitor and possibly improve the actual data quality during operations, as illustrated in the example of figure 3. However, the quality factors above are rather abstract. They need, on the one hand, to be focused on the actual quality goals of the stakeholders involved, and, on the other hand, to be made more specific by mapping them down to quality questions and specific measurement techniques to be stored in the database.

Thus, perhaps surprisingly, we can reuse the ideas of using specific agents for data extraction and analysis for the case of analyzing the quality of data and data streams. This is reflected in our quality meta model which is shown in figure 7. The organization of this meta model is not accidental but reflects a broader quality management paradigm – the so-called goal-question-metric (GQM) approach [OB92] – which has proven quite successful in a broad range of software engineering applications.

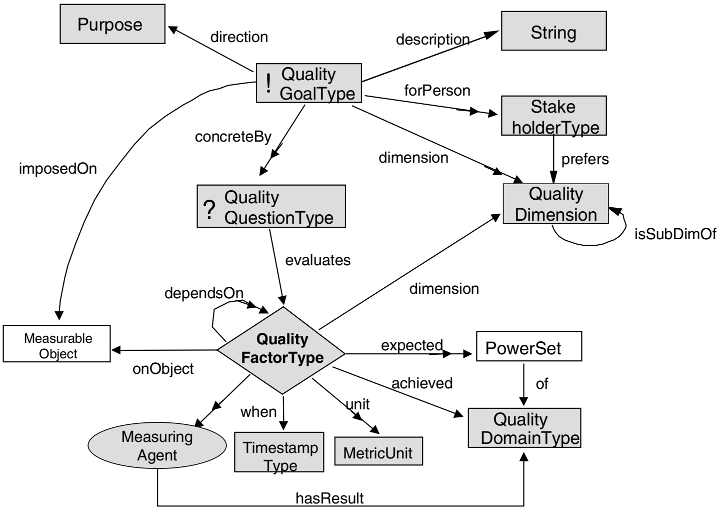


Fig. 7. Quality meta model

3 A Concept-Centered Data Warehouse Methodology

A large body of literature addresses the problems introduced by the data warehouse approach, such as the trade-off between freshness of data warehouse data and disturbance of OLTP work during data extraction; the minimisation of data transfer through incremental view maintenance; and a theory of computation with multi-dimensional data models. For an overview, see [JLVV99].

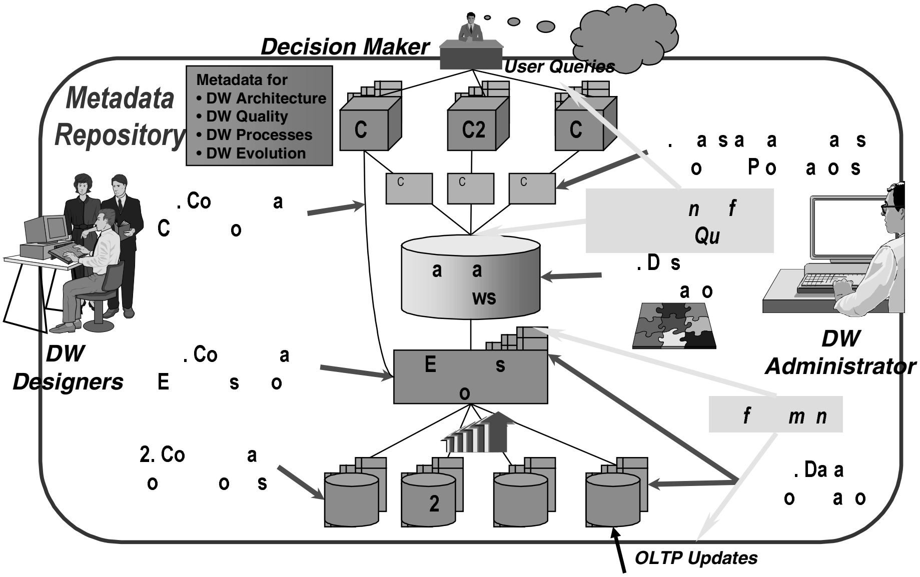
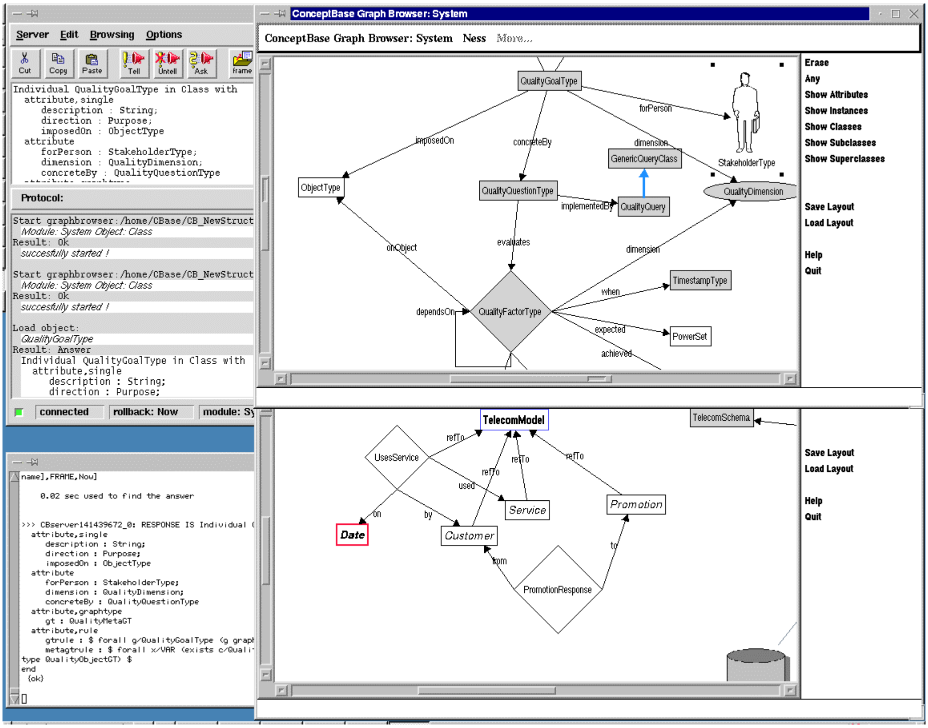


Fig. 8. DWQ Data Warehouse Development Process

The DWQ method consists of the six steps shown in figure 8. In this figure, the objects of the classical DW architecture from figure 4 have been overlaid with their conceptual correspondences, such as source models (e.g., S1), the enterprise model, and client models (e.g., C1). These conceptual objects are, in our approach, externally represented as extended Entity-Relationship models, and internally modeled using Description Logic formalisms from artificial intelligence to allow for subsumption reasoning. The two shadowed texts describe related support at the operational level which we do not discuss further in this paper : aggregate query optimization and view refreshment. The whole process is administered through a metadata repository which has been implemented using the ConceptBase system [JGJ+95].

In the following subsections, we illustrate the main steps by simplified examples taken from a case study for a large European telecom organization [TLN99]. As a start, the representation of the corresponding application models to the meta models shown in figures 6 and 7 is shown in the ConceptBase screendump of figure 9.



```
Individual QualityGoalType in Class with
  attribute, single
     description : String;
     direction : Purpose;
     imposedOn : ObjectType
  attribute
     forPerson : StakeholderType;
     dimension : QualityDimension;
     concreteBy : QualityQuestionType
```

```
Start graphbrowser:/home/CBase/CB_NewStruct
  Module: System Object: Class
Result: Ok
  succesfully started !

Start graphbrowser:/home/CBase/CB_NewStruct
  Module: System Object: Class
Result: Ok
  succesfully started !

Load object:
  QualityGoalType
Result: Answer
  Individual QualityGoalType in Class with
    attribute,single
      description : String;
      direction : Purpose;
```

```
name],FRAME,Now]

    0.02 sec used to find the answer


>>> CBserver141439672_0: RESPONSE IS Individual (
  attribute,single
     description : String;
     direction : Purpose;
     imposedOn : ObjectType
  attribute
     forPerson : StakeholderType;
     dimension : QualityDimension;
     concreteBy : QualityQuestionType
  attribute,graphtype
     gt : QualityMetaGT
  attribute,rule
     gtrule : $ forall g/QualityGoalType (g graph
     metagtrule : $ forall x/VAR (exists c/Quali
type QualityObjectGT) $
end
 {ok}
```

Fig. 9. Metadata model and telecom example in the ConceptBase meta database

3.1 Source Integration (Steps 1 and 2 of Figure 9)

The DWQ approach to source integration [CDL+01] is incremental. Whenever a new portion of a source is taken into account, the new information is integrated with an "Enterprise Model", and the necessary new relationships are added:

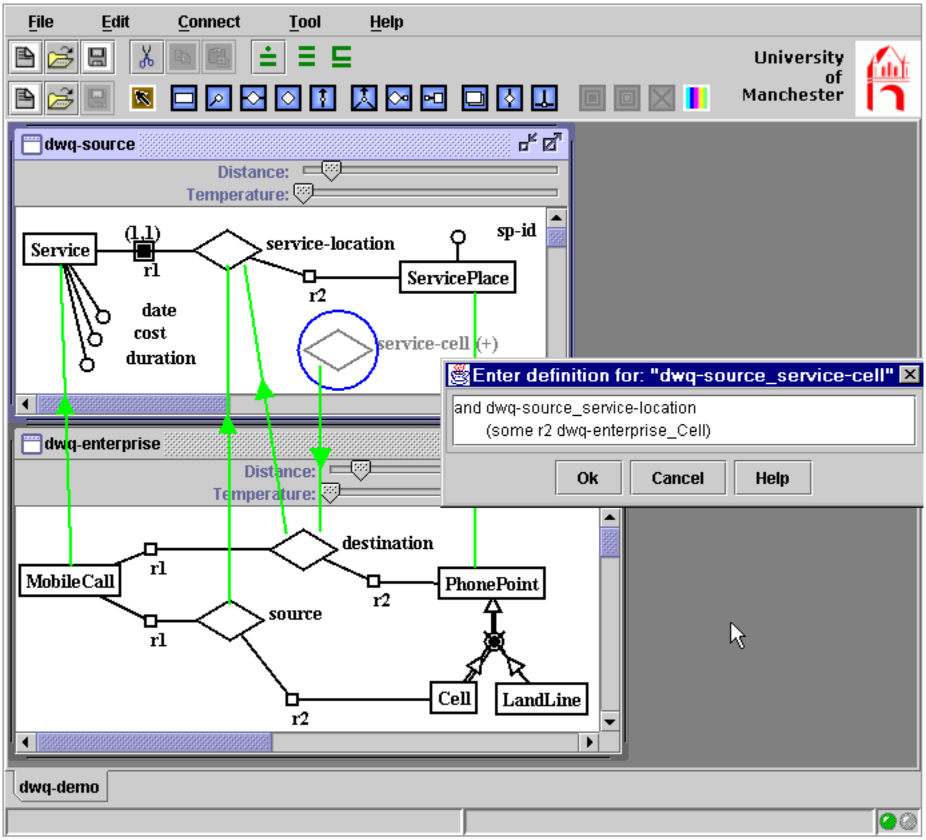
- The *Enterprise Model* is a conceptual representation of the global concepts and relationships that are of interest to the Data Warehouse application. It provides a consolidated view of the concepts and relationships that are important to the enterprise, and have so far been analysed. Such a view is subject to change as the analysis of the information sources proceeds. The Description Logic formalism we use is general enough to express the usual database models, such as the Entity-Relationship Model, and the Relational Model. Inference techniques associated with the formalism allow for carrying out several reasoning services on the representation. The formalism is hidden from the user of the DWQ tools who only uses a graphical interface.
- For a given information source S, the *Source Model* of S is a conceptual representation of the data residing in S. Again, the approach does not require a source to be fully conceptualised. Source Models are expressed by means of the same formalism used for the Enterprise Model.

- Integration does not simply mean producing the Enterprise Model, but rather being able to establish the correct relationships both between the Source Models and the Enterprise Model, and between the various Source Models. We formalize the notion of interdependency by means of *intermodel assertions* [CL93]. An intermodel assertion states that one object (i.e., class, entity, or relation) belonging to a certain Model (either the Enterprise or a Source Model) is always a subset of an object belonging to another Model. This simple declarative mechanism has been shown to be extremely effective in establishing relationships among different database schemas [Hull97]. We use a description logic-based formalism (using the iFACT reasoner [Horr99]) to express intermodel assertions, and the associated inference techniques provide a means to reason about interdependencies among models.
- The logical content of each source S, called the *Source Schema*, is provided in terms of a set of definitions of relations, each one expressed as a query over the Source Model of S. The logical content of a source represents the structure of data expressed in terms of a logical data model, such as the Relational Model. The logical content of a source S, or of a portion thereof, is described in terms of a view over the Source Model associated with S (and, therefore, of the Conceptual Data Warehouse Model). Wrappers map physical structures to logical structures.
- The logical content of the materialized views constituting the Data Warehouse, called the *Data Warehouse Schema*, is provided in terms of a set of definitions of relations, each one expressed in terms of a query over the Conceptual Data Warehouse Model. How a view is actually materialized starting from the data in the sources is specified by means of *Mediators*.

The following tasks work on this structure within steps 1 and 2 of figure 1:

- *Enterprise and Source Model construction.* The Source Model corresponding to the new source is produced, if not available. Analogously, the conceptual model of the enterprise is produced, if not available.
- *Source Model integration.* The Source Model is integrated into the Conceptual Data Warehouse Model. This can lead to changes both to the Source Models, and to the Enterprise Model. Moreover, intermodel assertions between the Enterprise Model and the Source Models and between the new source and the existing sources are added. The designer can specify such intermodel assertions graphically, and can invoke various automated analyses.
- *Source and Data Warehouse Schema specification.* The Source Schema corresponding to a new portion of the source is produced. On this basis, an analysis is carried out if the Data Warehouse Schema should be restructured and/or modified.

In all of these tasks, the metadata repository stores the values of the quality factors involved in source and data integration, and helps analyze the quality of the design choices. The Quality Factors of the Conceptual Data Warehouse Model and the various schemas are evaluated and a restructuring of the models and the schemas is accomplished to match the required criteria. Figure 10 shows an example where a new correspondence relationship between the (upper) enterprise model (upper ER model) and a particular source model (lower ER model) is entered via a simple description logic expression (popup window in the middle). The system will, in this case, return a graphical error message identifying an inconsistency with previous definitions.



Enter definition for: "dwq-source_service-cell"

```
and dwq-source_service-location
    (some r2 dwq-enterprise_Cell)
```

Fig. 10. Specification and DL-based quality control of source integration relationships

3.2 Multidimensional Aggregation and OLAP Query Generation

The next two steps consider the client side, symmetric to the source integration.

Step 3: Conceptual Client Modeling. The conceptual modeling language underlying the enterprise and source models, and the corresponding modeling and reasoning tools, have been extended to the case where concepts are organised into aggregates along multiple dimensions with multi-hierarchy structure [FS99]. Data warehouse designers can thus define multi-dimensional and hierarchical views over the enterprise conceptual model in order to express the interests of certain DW client profiles, without losing the advantages of consistency and completeness checking as well as semantic optimisation provided by the conceptual modeling approach. Examples for such multi-hierarchies include time (modeled in days-weeks or in days-months-years). Figure 11 gives an impression of this extension in which a new aggregate relationship (AggregateMobileCalls2) is inferred from existing ones and can then be renamed adequately.

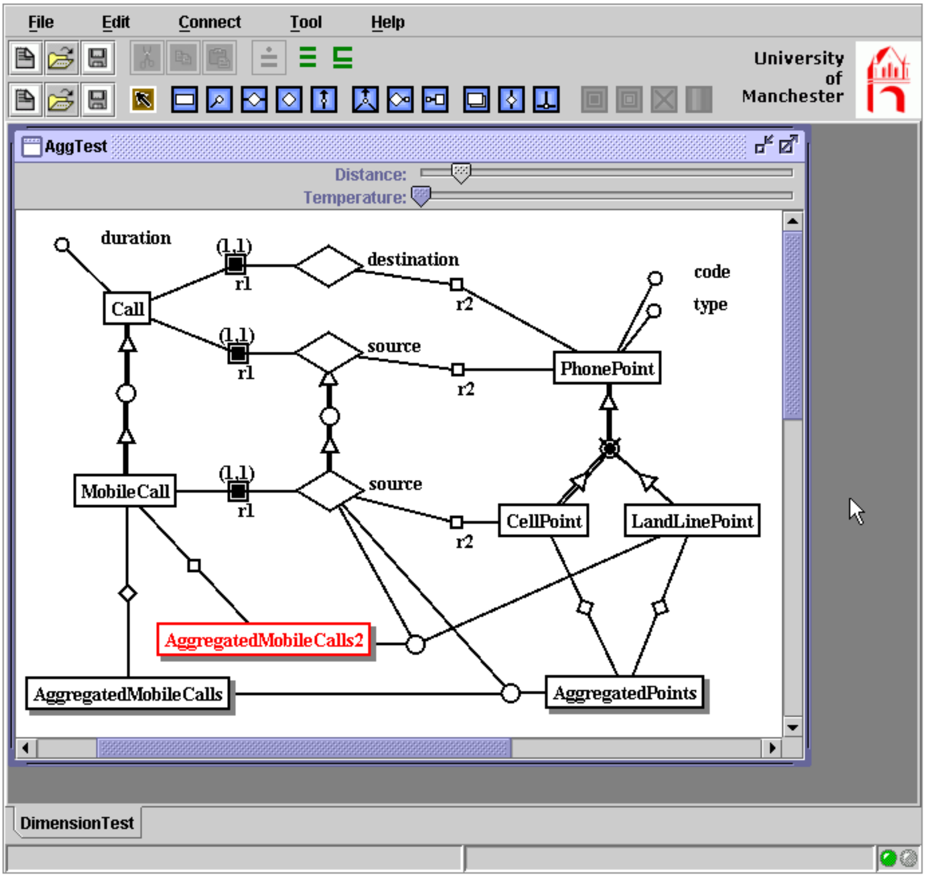


Fig. 11. Extended ER model and DL reasoning facility enhanced with modelling of aggregates

Step 4: **Translate Aggregates into OLAP queries.** The thus defined "conceptual data cubes" can either be implemented directly by multi-dimensional (MOLAP) data models, or supported by a mapping to relational database systems (ROLAP). A star schema is often used in OLAP systems with very large sparse data cubes because a mapping to just one data cube would be extremely inefficient and therefore the non-empty pieces of the cube are parcelled out into many smaller sub-cubes (recall that there were about 40.000 of these in our banking example). Faithful representation of client views requires a careful design of an OLAP relational algebra, together with the corresponding rewritings to underlying star schemas [Vass98].

3.3 Design Optimization and Data Reconciliation

Step 5: Quantitative Design Optimization. Viewed from the logical perspective, our conceptually controlled approach to source integration has created a schema of relations implementing the enterprise model. This schema could be implemented directly as an operational data store (ODS). In a data warehouse with lots of updates and completely unpredictable queries, this would be the appropriate view materialisation.

On the other extreme, the mapping of multi-dimensional aggregates to ROLAP queries creates a set of view definitions (queries). The materialization of these queries would be the optimal solution (storage space permitting!) in a query-only data warehouse with hardly any updates.

Typical data warehouses have less extreme usage patterns and therefore require a compromise between the two view materialization strategies. The DWQ project has investigated solutions to this combinatorial optimization problem [TS97].

Step 6: Data Reconciliation. The physical-level optimization is fully integrated with the conceptual modeling approaches because it works on their outcomes. Conversely, the resulting optimal design is now implemented by data integration and reconciliation algorithms, semi-automatically derived from the conceptual specifications. The views to be materialized are initially defined over the ODS relations. There can be several qualitatively different, possibly conflicting ways to actually materialize these ODS relations from the existing sources. These ways are generated by a further set of rewritings that can be derived from the source integration definitions [CDL+01].

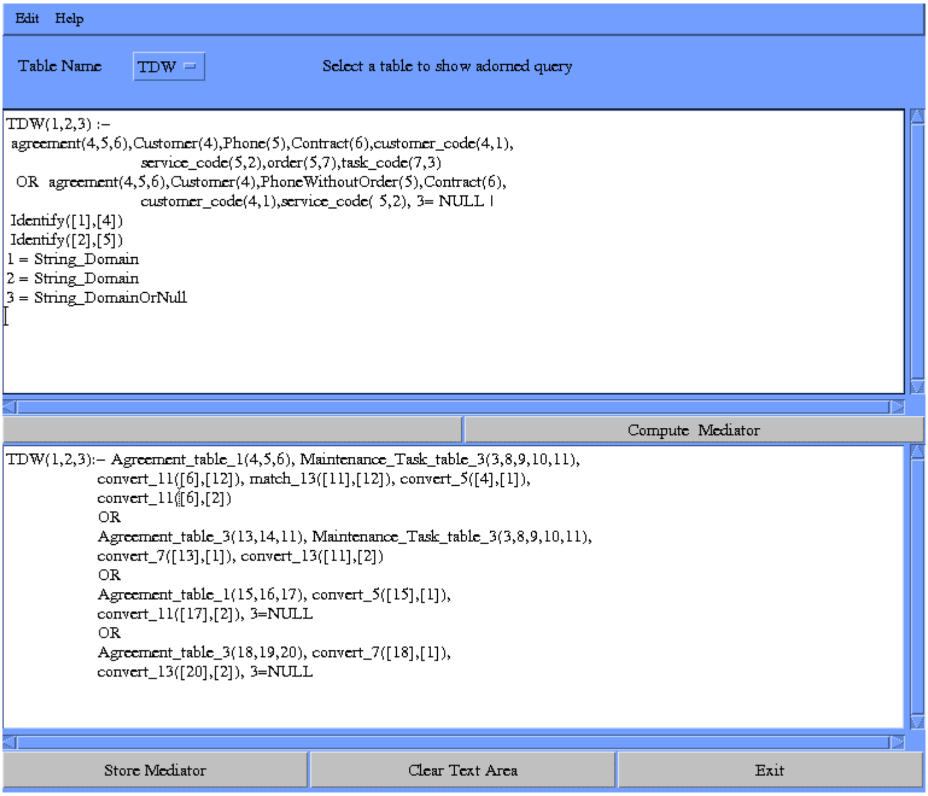
The problem of data reconciliation arises when data passes from the application-oriented environment to the Data Warehouse. During the transfer of data, possible inconsistencies and redundancies are resolved, so that the warehouse is able to provide and integrated and reconciled view of data of the organization.

In the DWQ methodology, data reconciliation is based on (1) specifying through Interschema Assertions how the relations in the Data Warehouse Schema are linked to the relations in the Source Schemas, and (2) designing suitable mediators for every relation in the Data Warehouse Schema.

In step (1), interschema correspondences are used to declaratively specify the correspondences between data in different schemas (either source schemas or data warehouse schema). Interschema correspondences are defined in terms of relational tables, similarly to the case of the relations describing the sources at the logical level. We distinguish three types of correspondences, namely Conversion, Matching, and Reconciliation Correspondences. By virtue of such correspondences, the designer can specify different forms of data conflicts holding between the source, and can anticipate methods for solving such conflicts when loading the Data Warehouse.

In step (2), the methodology aims at producing, for every relation in the Data Warehouse Schema, a specification of the corresponding mediator, which determines how the tuples of such a relation should be constructed from a suitable set of tuples extracted from the relations stored in the sources.

In effect, we have here a semi-automatic generation and embedding of mediator agents for data extraction, transformation, cleaning, and reconciliation. Figure 12 shows how the prototypical tools developed in the DWQ project solve this problem by query adornments in a deductive database style which maps easily to standard relational query languages such as SQL.



```
TDW(1,2,3) :-
 agreement(4,5,6),Customer(4),Phone(5),Contract(6),customer_code(4,1),
                    service_code(5,2),order(5,7),task_code(7,3)
  OR  agreement(4,5,6),Customer(4),PhoneWithoutOrder(5),Contract(6),
                    customer_code(4,1),service_code( 5,2), 3= NULL |
 Identify([1],[4])
 Identify([2],[5])
1 = String_Domain
2 = String_Domain
3 = String_DomainOrNull
```

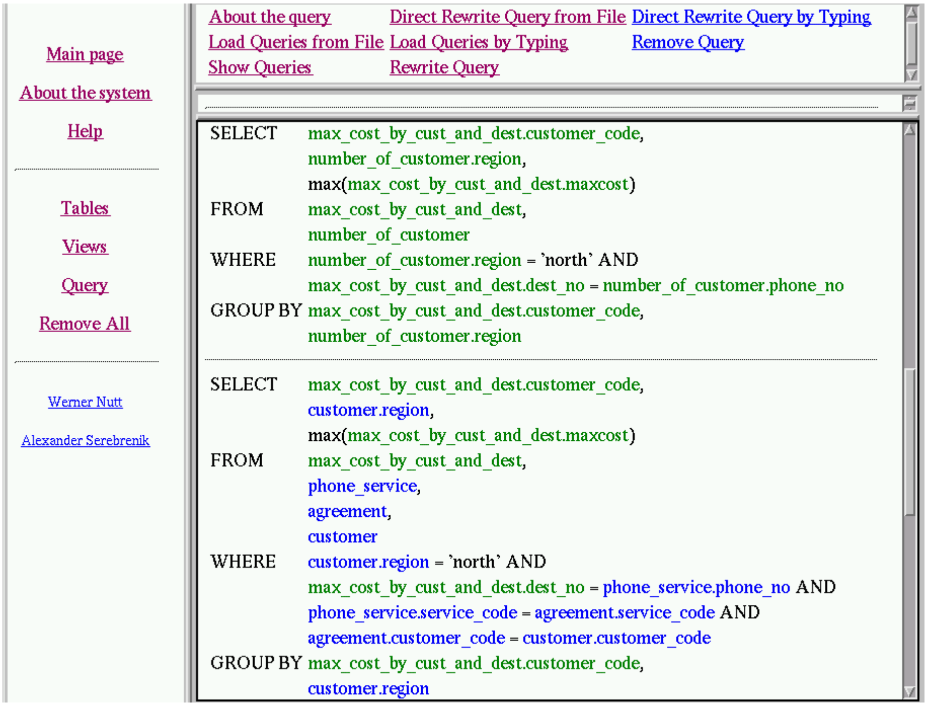
```
TDW(1,2,3):- Agreement_table_1(4,5,6), Maintenance_Task_table_3(3,8,9,10,11),
             convert_11([6],[12]), match_13([11],[12]), convert_5([4],[1]),
             convert_11([6],[2])
             OR
             Agreement_table_3(13,14,11), Maintenance_Task_table_3(3,8,9,10,11),
             convert_7([13],[1]), convert_13([11],[2])
             OR
             Agreement_table_1(15,16,17), convert_5([15],[1]),
             convert_11([17],[2]), 3=NULL
             OR
             Agreement_table_3(18,19,20), convert_7([18],[1]),
             convert_13([20],[2]), 3=NULL
```

Fig. 12. Data reconciliation using mediator frameworks with adorned logic queries

3.4 Operational Support

The runtime environment of the data warehouse which operates on the structures and mechanisms designed as described in the previous subsections, need enhancements both at the logical and at the physical level.

At the logical level, the system needs to take advantage of the design optimization in step 4 which generated a specific set of materialized views to reduce query costs. However, the client user should not have to know which views have been materialized. Indeed, ideally he should be able to formulate his queries directly on graphical or form-based representations of the conceptual client views designed in step 3. This creates the need for a further automatic rewriting – from the OLAP mapping of the client views (prepared in step 4) to the views materialized in step 5. This rewriting problem becomes quite tricky where aggregates are involved but good solutions (some of them heuristics) are known for a number of important special cases. In [CNS99], the solution underlying our approach is described. An example is given in figure 13.



```
SELECT    max_cost_by_cust_and_dest.customer_code,
          number_of_customer.region,
          max(max_cost_by_cust_and_dest.maxcost)
FROM      max_cost_by_cust_and_dest,
          number_of_customer
WHERE     number_of_customer.region = 'north' AND
          max_cost_by_cust_and_dest.dest_no = number_of_customer.phone_no
GROUP BY  max_cost_by_cust_and_dest.customer_code,
          number_of_customer.region

SELECT    max_cost_by_cust_and_dest.customer_code,
          customer.region,
          max(max_cost_by_cust_and_dest.maxcost)
FROM      max_cost_by_cust_and_dest,
          phone_service,
          agreement,
          customer
WHERE     customer.region = 'north' AND
          max_cost_by_cust_and_dest.dest_no = phone_service.phone_no AND
          phone_service.service_code = agreement.service_code AND
          agreement.customer_code = customer.customer_code
GROUP BY  max_cost_by_cust_and_dest.customer_code,
          customer.region
```

Fig. 13. Query rewriting using aggregate views

Second, source data are continuously changing – one of the main reasons why triggers and agent technologies have been proposed for monitoring such changes through active wrappers and reconciliating them incrementally into the data warehouse. Database research has produced a large number of associated algorithms at the logical perspective of incremental view maintenance.

In parallel, practice has produced so-called ETL tools (extraction-transformation-loading) to handle the physical aspects. As again evidenced by the banking example in figure 3, there is a strong need for workflow-like mechanisms that schedule the many different operations of data extraction and staging, data cleaning and integration, history management and propagation according to the identified quality needs of the enterprise. In some cases, the active database mechanisms provided by many DBMS can be used to handle this task. However, especially in the case where massive amounts of data need to be reconciled on a regular basis, performance modeling and tuning of these workflows becomes a must (see [NJ00] for a survey of modeling techniques for replicated and distributed databases). In the DWQ project, a toolkit for such a workflow system was prototypically developed and a number of different policies were evaluated [BFMS98].

A real-world application example [JQB+99] of how these logical and physical aspects are integrated in practice is shown in figure 14. The enterprise in question regularly extracts heterogenous transactional source data into a (relational) data warehouse of information relevant for its sales force. They have implemented some of our view maintenance mechanisms [SJ96] and combined them with the loading and replication mechanisms offered by the groupware system Lotus Notes at the physical level to create mobile agents which can be carried along on the laptops of the sales force and re-synchronized from time to time with the status of the data warehouse.

However, the declarative approach pursued by us for the mapping between sources, enterprise, and client models led to another surprising effect. Sales people noted that the quality of the data produced by the data warehouse was much better than that of the operational data sources due to the reconciliation mechanisms offered, and the same sales people of course also produced transactions for these sources as well – with much less quality control. As a consequence, we used the same model interrelationships backwards to create a semi-automatic view update mechanisms which propagated changes on the client views back into the transactional systems, thus significantly improving source data quality. The degree of automation could have been even greater, had not the owners of the data sources seen security risks involved via this new access path to their databases.

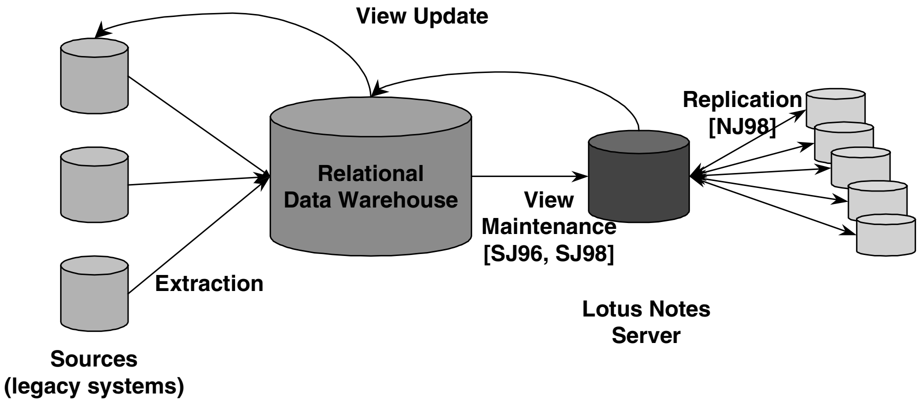


Fig. 14. A data warehouse with forward and backward data refreshment

4 Conclusion

In this paper, we have presented a comprehensive approach how to improve the quality of data warehouses through the metadata-based management of the relationships between distributed and heterogeneous data and knowledge sources. We have described the motivation for such an approach, the metadata structures for architecture and quality underlying it, and a methodology for the design integrating the three perspectives we have proposed -–the conceptual, the logical, and the physical one. Finally, we have shown with several examples that this approach does not only work in theory but that facets of it have been used with significant success in industry.

In many of the conceptual models presented throughout this paper, the concept of agent has played an important role. However, in most cases, our use of agents has been a conceptual one, mapped down to rather conventional AI and database technologies at the implementation level, largely due to the rigorous data quality demands and massive performance challenges faced in our example applications. Even now, some source integration might be solved with a higher degree of automation by adding ontology-based negotiations between intelligent software agents along the ideas of the InfoSleuth project [PTU99] to our approach. With increasing heterogeneity and volatility of the data sources and increasing demands on client-side personalization and contextualization, we expect an increase of the role of agent software technology in data warehouse applications.

Acknowledgments. This work was supported in part by the European Commission under ESPRIT Long Term Research Project DWQ and by the Deutsche Forschungsgemeinschaft under SFB 476 (IMPROVE). Thanks are due to numerous collaborators in these projects, notably Maurizio Lenzerini, Enrico Franconi, Yannis Vassiliou, Panos Vassiliadis, Werner Nutt und Christoph Quix.

References

Bouzeghoub, M., Fabret, F., Matulovic, M., Simon, E.: Developing efficient and customizable active systems using a toolkit. DWQ Technical Report, October 1998.

Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Data integration in data warehousing. *International Journal of Cooperative Information Systems*, spring 2001.

Catarci, T., Lenzerini, M.: Representing and using interschema knowledge in cooperative information systems. *Intelligent and Cooperative Information Systems*, 2(4), 375–398, 1993

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstaninou, Y., Ullman, J.D., Widom, J.: The TSIMMIS project: integration of heterogeneous information sources. Proc. IPSI Conference. Tokyo 1994.

Cohen, S., Nutt, W., Serebrenik, A.: Rewriting aggregate queries using views. *Proc. 18th Symp. Principles of Database Systems (PODS),* Philadelphia 1999.

Franconi, E., Sattler, U.: A data warehouse conceptual data model for multidimensional aggregation. In *Proc. Int. Workshop Design and Management of Data Warehouses (DMDW'99),* Heidelberg, Germany, 1999.

Horrocks, I.: FaCT and iFaCT. In P. Lambrix, A. Borgida, M. Lenzerini, R. Müller, and P. Patel-Schneider, eds, *Proc. International Workshop on Description Logics (DL'99)*, number 22 in CEUR-WS, pages 133-135, Linkoeping, Sweden, July 30 - August 1 1999.

Hull, R.: Managing semantic heterogeneity in databases: a theoretical perspective. *Proc. 16th ACM Symp. Principles of Database Systems (PODS)*, Tucson, AZ, pp. 51–61, 1997.

Jarke, M., Gallersdörfer, R., Jeusfeld, M.A., Staudt, M., Eherer, S.: ConceptBase – a deductive object base for meta data management. *Intelligent Information Systems*, 4(2), 167–192, 1995.

Jarke, M., Jeusfeld, M.A., Quix, C., Vassiliadis, P.: Architecture and quality in data warehouses. Proc. CAiSE 98, Pisa, Italy, LNCS 1413, Springer 1999,93-115.

Jarke, M, Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: *Fundamentals of Data Warehouses*. Springer 1999.

Jarke, M., Klemke, R., Nick, A. Broker's Lounge – an environment for multi-dimensional user-adaptive knowledge management. *Proc. HICSS 34*, Maui, Hw, 2001

Jarke, M., Quix, C., Blees, G., Lehmann, D., Michalk, G., Stierl, S.: Improving OLTP data quality using data warehouse mechanisms. *Proc. ACM-SIGMOD Conf. Management of Data*, Philadelphia 1999.

Jeusfeld, M.A., Papazoglou, M.P.: Information brokering. In B.Krämer, M. Papazoglou, H.-W. Schmidt (eds.): *Information Systems Interoperability*, John Wiley Research Studies Press 1998, 265-301.

Kashyap, V., Sheth, A.: Semantic heterogeneity in global information systems: the role of metadata, context, and ontologies. In Papazoglou, M.P./ Schlageter, G. (eds.): *Cooperative Information Systems. Trends and Directions*. Academic Press 1998, 149-178.

Klusch, M., Benn, W.: Intelligente Informationsagenten im Internet. *KI* 3/1998, 8-16.

Levy, A., Rajmaran, A., Ordille, J.: Query answering algorithms for information agents. Proc. AAAI-96, pp. 40-47, 1996.

Nicola, M., Jarke, M.: Performance modeling of distributed and replicated databases. Research Survey, *IEEE Trans. Knowledge and Data Engineering*, August 2000.

Oivo, M., Basili, V.: Representing software engineering models: the TAME goal-oriented approach. IEEE Trans. Software Eng. 18, 10 (1992), 886-898.

Papazoglou, M.P., Laufmann, S., Sellis, T.K.: An organizational framework for cooperating intelligent informaton systems. *Intl. J. Intelligent and Cooperative Information Systems* 1, 1 (1992), 169-202.

Perry, B., Taylor, M., Unruh, A.: Information aggregation and agent interaction patterns in InfoSleuth. Proc. Proc. 4th IFCIS Intl. Conf. Cooperative Information Systems, Edinburgh 1999, 314-324.

Schäfer, E., Becker, J.-D., Jarke, M. DB-Prism – integrated data warehouses and knowledge networks for bank controlling. Industrial paper, *Proc. 26th Intl. Conf. Very Large Data Bases (VLDB 2000*), Cairo, Egypt, 715-718

Staudt, M., Jarke, M.: Incremental maintenance of externally materialized views. *Proc. 22nd VLDB Conference*, Mumbai, India, 1996.

Theodoratos, D., Sellis, T.: Data warehouse configuration. *Proc. 23rd International Conference on Very Large Databases (VLDB)*, 126–135, Athens, Greece, 1997.

Trisolini, S., Lenzerini, M., Nardi, D.: Data integration and warehousing in Telecom Italia. *Proc. ACM SIGMOD Intl. Conf. Management of Data*, 538-539, 1999

Wang, R. (ed.): *Data Quality*. Kluwer Publ. 2000.

Wiederhold, G., Genesereth, M.: The basis for mediation. *Proc. 3rd Intl. Conf. Cooperative Information Systems*, Vienna, Austria, 1995, 140-155.

Decision Trees for Multiple Abstraction Levels of Data*

Doheon Lee, Mina Jeong, and Yong-Kwan Won

Department of Computer Science, Chonnam National University
300 Yongbong-Dong Buk-Gu Kwangju, REPUBLIC OF KOREA
dhlee@dbcore.chonnam.ac.kr

Abstract. Since the data is collected from disparate sources in many actual data mining environments, it is common to have data values in different abstraction levels. This paper shows that such multiple abstraction levels of data can cause undesirable effects in decision tree classification. After explaining that equalizing abstraction levels by force cannot provide satisfactory solutions of this problem, it presents a method to utilize the data as it is. The proposed method accommodates the generalization/specialization relationship between data values in both of the construction and the class assignment phases of decision tree classification. The experimental results show that the proposed method reduces classification error rates significantly when multiple abstraction levels of data are involved.

1 Introduction

Classification is one of the widely used data mining tasks. It consists of the construction phase and the assignment phase. The construction phase builds a classification model based on a given training set. The assignment phase utilizes the classification model to assign class labels to target records. Among many classification models proposed in the literature, decision trees are especially attractive in data mining environments due to their intuitive representation easy to understand by human users [1][2][3][4].

The training set is commonly collected from disparate sources in actual data mining environments. Information gathering agents access local sources, extract relevant data items, and deliver them to the central repository. Since the information gathering agents do not have any prior knowledge about global data distribution, the collected data in the training set is apt to be expressed in different abstraction levels. For example, a sales item description is expressed as ‘Coke’ in a local table, while it is expressed as ‘Diet Coke 1.5 P.E.T’ in other local table. In network alarm management systems, a local network management agent may report alarms in a higher abstraction level such as ‘Router Malfunction’ while other local network management agent reports them in a lower abstraction level such as ‘Router Type-32 Message Overflow’

* This work has been supported by Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 76–87, 2001.

[5]. We call this the *multiple abstraction level* problem. Uncontrolled data entry by human users can also cause the same problem. Though sophisticated user interfaces of data entry software and employee training programs can help standardizing data entry, the problem cannot be eliminated due to mistakes and/or unavailable information [6].

The multiple abstraction levels can cause a severe problem in decision tree classification. Suppose that we have built a decision tree from multiple abstraction levels of training data as shown in Fig. 1. Notice that the abstraction level of 'Far-East' and 'Mid-East'; and that of 'East' and 'West' are different. Actually, 'Far-East' and 'Mid-East' are specializations of 'East'. Also suppose that we are to assign the class label to a target record such that (Mid-East, 85k, Male). One choice is to follow the third branch of the root node, and assigns 'E_2' to the target record. However, since 'Mid-East' belongs to 'East', the other choice is to follow the first branch of the root node; and in turn, the second branch of the 'Income' node; and assign 'E_1' to the target record. As a result, the decision tree yields two contradictory assignments.

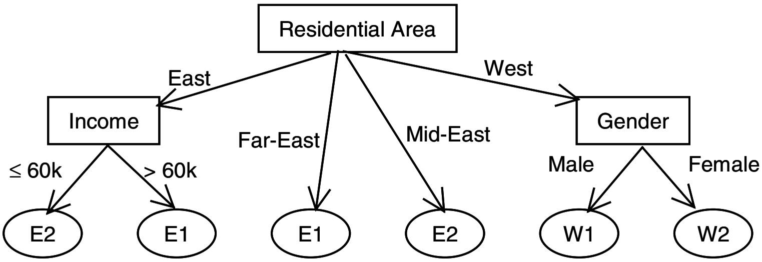


Fig. 1. An example decision tree with multiple abstraction levels of data

As this sort of data quality problem has been a subject of longstanding discussions [6][7], there have been developed data cleansing tools to improve the data quality on the market [8][9][10]. At first glance, there seems to be two options to remedy the problem by using such tools. One option is to equalize abstraction levels by generalizing too specific data values. It is obvious that more specific data values yield more informative decision trees [4]. Thus, this option results in the loss of useful information. The other option might be equalizing abstraction levels by specializing too general data values. Since it requires extracting additional information from the sources, it is hard or even impossible in actual data mining environments. Thus, the second option is also inapplicable due to the lack of information. Furthermore, it is also hard to determine how many levels we should generalize or specialize data values in both options. Consequently, the existing data cleansing tools cannot provide satisfactory solutions of the problem.

In this paper, we propose more practical methods for the multiple abstraction level problem in decision tree classification. Rather than generalizing or specializing data values by force, we utilize the information as it is, but accommodate the generalization/specialization relationship between data values in both of the construction and assignment phases. The paper is organized as follows. Section 2 reviews the skeleton algorithm of decision tree construction, and proposes modified versions of the best-split attribute selection and partitioning of training sets. Section 3 provides how to assign the class labels to target records when multiple abstraction levels of data exist. Section 4 presents experimental results to analyze classification accuracy improvement. Section 5 concludes.

2 Constructing Decision Trees

To construct a decision tree, we need a training set. A training set is a data relation whose scheme consists of one or more predictor attributes and a class label. Briefly, a decision tree is a tree-structured representation of the distribution of the class label in terms of the predictor attributes [1][2][4]. This section addresses how to construct a decision tree when a training set is given in multiple abstraction levels.

A decision tree is constructed through recursive partitioning of a training set. The first step is the selection of a predictor attribute that yields the best split of the training set. Once such a predictor attribute is selected, the training set is partitioned into subsets according to the attribute values. This selection-and-partitioning procedure is applied to each subset recursively. Fig. 2 depicts a skeleton algorithm for constructing decision trees.

```
ConstructTree(Node ThisNode, Relation R) {
(1)     Attr = SelectBestSplit(R);        // Select the best-split attribute of R
(2)     ThisNode.label = Attr;            // Mark the label of ThisNode as Attr
(3)     For each attribute value x of Attr {
                // Select records whose Attr attributes are 'x'
(4)             NewR = SelectRecords(R, Attr, x);
                // Make a new decision node
(5)             NewNode = MakeNewNode( );
                // Make NewNode a child node of ThisNode
(6)             NewNode.parent = ThisNode;
                // Invoke this procedure itself for the child node
(7)             ConstructTree(NewNode, NewR);  } }
```

Fig. 2. A skeleton algorithm for constructing decision trees

In Line (1), one of the predictor attributes of the relation *R* is selected as the best-split attribute. Though there are a variety of measures to select the best-split attributes, their common goal is to evaluate which attribute leads the most homogeneous subsets with respect to the class label after partitioning. Suppose that we have a training set of bank customer profiles, and the class label is ‘frequently visited branch’. Also suppose that there are two predictor attributes ‘residential area’ and ‘job type’. Since there is strong dependency between ‘frequently visited branch’ and ‘residential area’, each subset after partitioning according to ‘residential area’ values will contain almost the same branches. But if we partition it according to ‘job type’ values, each subset will contain a mixture of several branches. In other words, ‘residential area’-based partitioning will result in more homogeneous subsets than ‘job type’-based partitioning. As a result, we select ‘residential area’ as the best-split attribute for the class label ‘frequently visited branch’.

Once the attribute ‘residential area’ is selected, we partition the training set in Line (4). If *k* different residential areas appear in the training set, *k* child nodes are allocated. Those records in the training set whose ‘residential area’ is the *i*th residential area are assigned to the *i*th child node. This selection-and-partitioning procedure is invoked recursively for each subset in Line (7). Some variations of decision tree construction algorithms such as CART produce only two branches rather than *k* branches [3]. However, they should eventually deal with *k* residential areas in the subsequent levels of the decision tree.

When the training set contains records expressed in multiple levels of abstraction, partitioning the training set according to the best-split attribute (SelectRecords( ) in Line (5)) has to be revised as well as the selection of the best-split attribute (SelectBestSplit( ) in Line (1)).

2.1 Partitioning Training Sets with Multiple Abstraction Levels

Let us motivate why partitioning the training set has to be modified through an example. Suppose that we have a training set of customer profiles as Table 1. It consists of three predictor attributes, ‘Residential Area(RA)’, ‘Gender’, and ‘Income’; and a class label ‘Frequently Visited Branch(FVB)’. Though common training sets in actual data mining environment might contain at least tens of thousands records, our example contains only ten records for simplicity.

Table 1. A training set of customer profiles

RID	RA	Gender	Income	FVB
t1	East	Male	90k	E2
t2	East	Male	70k	E1
t3	Far-East	Female	80k	E1
t4	Mid-East	Male	50k	E2
t5	Mid-East	Female	30k	E2
t6	West	Male	90k	W1
t7	West	Male	50k	W1
t8	West	Female	100k	W2
t9	West	Female	40k	W2
t10	West	Female	50k	W2

RA: Residential Area
FVB: Frequently Visited Branch (class attribute)
RID: Record ID (only for explanation)

For the moment, let us suppose that the predictor attribute ‘Residential Area’ is selected as the best-split attribute with respect to the class label ‘FVB’. The method of selecting the best-split attribute will be discussed precisely in Section 2.2. Now, we have to partition the training set according to ‘RA’ values. Conventional partitioning results in $\{t_1, t_2\}$, $\{t_3\}$, $\{t_4, t_5\}$, and $\{t_6, t_7, t_8, t_9, t_{10}\}$. However, such partitioning misses the fact that ‘Far-East’ and ‘Mid-East’ belong to ‘East’. Thus, we can include those records whose ‘RA’ values are either ‘Far-East’ or ‘Mid-East’ in the first subset for ‘East’. Then, the result becomes $\{t_1, t_2, t_3, t_4, t_5\}$, $\{t_3\}$, $\{t_4, t_5\}$, and $\{t_6, t_7, t_8, t_9, t_{10}\}$.

Unfortunately, it still misses the fact that though the residential area of the first and second customers is registered as ‘East’, the actual residential area of them may be either ‘Far-East’ or ‘Mid-East’. Since a proper training set reflects the value distribution of the entire domain, we can obtain the probability of the actual residential areas from the distribution of the training set. Three records, t_3, t_4, and t_5 have specialized ‘RA’ values of ‘East’. Among them one record, t_3, has the value of ‘Far-East’, while the other two records, t_4 and t_5, have the value of ‘Mid-East’. From this distribution, we can say that the probability that the ‘RA’ value of t_1 (or t_2) would be actually ‘Far-East’ is 1/3 = 33%. Similarly, the probability that the ‘RA’ value of t_1 (or t_2) would be actually ‘Mid-East’ is 2/3 = 67%. As a result, we can partition the training set as $\{t_1, t_2, t_3, t_4, t_5\}$, $\{t_1/0.33, t_2/0.33, t_3\}$, $\{t_1/0.67, t_2/0.67, t_4, t_5\}$, and $\{t_6, t_7, t_8, t_9, t_{10}\}$, where t_i/μ denotes that t_i belongs to the set with the membership degree μ. Definition 1 and 2 formalize this observation.

As seen in the last example, it is necessary to deal with the partial membership of a record in a set. Rather than inventing an ad-hoc representation for membership degrees, let us adopt the notion of fuzzy relations.

Definition 1 (Fuzzy relation) [11].
A fuzzy relation T is defined as follows.
$T = \{(t, \mu_T(t)) \mid t \text{ is an ordinary record}, \mu_T(t) \text{ is the membership degree of } t \text{ in } T\}$

A membership degree $\mu_T(t)$ is attached to each record to represent how completely the record belongs to the set. If $\mu_T(t) =1$, it implies the complete membership, while $\mu_T(t) < 1$ implies the partial membership. When the corresponding set T is obvious in the context, $\mu_T(t)$ is written just as $\mu(t)$. In fact, an ordinary relation is a special case of a fuzzy relation where all the records have membership degrees of 1.0. Thus, a training set of records is regarded as a fuzzy relation from now on.

Definition 2 (Partitioning a training set).
When a fuzzy relation *T* (a training set) and an ISA hierarchy [13] *H* on the domain of an attribute *X* are given, the selection from *T* with the condition such that ‘*X* is x’, *SR(T, H, X, x)* is defined as follows.

$SR(T, H, X, x) = SR_{direct}(T, X, x) \cup SR_{descendent}(T, H, X, x) \cup SR_{antecedent}(T, H, X, x),$

where $SR_{direct}(T, X, x) = \{ (t, \mu(t)) \mid t \in T, t.X = x, \mu(t) = \mu_T(t)\},$

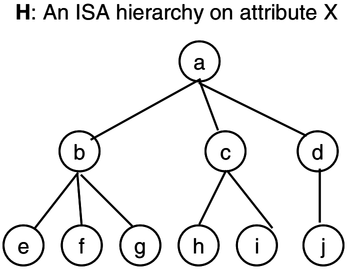
$SR_{descendent}(T, H, X, x) = \{ (t, \mu(t)) \mid t \in T, t.X \in DESC(x, H), \mu(t) = \mu_T(t)\},$ and

$SR_{antecedent}(T, H, X, x) = \{ (t, \mu(t)) \mid t \in T, t.X \in ANTE(x, H),$

$\mu(t) = \mu_T(t) \times (Card(\{(s,\mu_T(s)) \mid s \in T, s.X = x \text{ or } s.X \in DESC(x, H)\}) /$

$Card(\{(s,\mu_T(s)) \mid s \in T, s.X \in DESC(t.X, H)\}))\}.$

ANTE(x) and *DESC(x)* denote sets of values appearing as antecedents and descendents of x, respectively, in the ISA hierarchy *H*. $Card(T) = \Sigma\mu_T(t)$.



T: A training set

RID	X	μ(ti)	*RID*	X	μ(ti)
t1	a	1.0	*t9*	b	1.0
t2	c	1.0	*t10*	c	1.0
t3	d	1.0	*t11*	f	1.0
t4	h	0.7	*t12*	j	0.4
t5	a	0.9	*t13*	b	0.6
t6	c	1.0	*t14*	d	1.0
t7	e	0.9	*t15*	f	1.0
t8	i	1.0	*t16*	g	0.9

Fig. 3. An ISA hierarchy and a fuzzy relation as a training set

Though the formulae in Definition 2 seem slightly complex, the idea is simple. The result set after selecting records whose *X* values are x from *T*, i.e. *SR(T, H, T, x),* consists of three parts. The first subset $SR_{direct}(T, X, x)$ is a set of records whose *X* values are literally identical to ‘x’. Obviously, the μ(t) value is the membership degree of t in *T*, i.e., $\mu_T(t)$. The second subset $SR_{descendent}(T, H, X, x)$ is a set of records whose ‘*X*’ values are specializations of ‘x’. In the previous example, ‘Far-East’ represents a specialization of ‘East’. In this case, the μ(t) values is still $\mu_T(t)$, since a specialization completely belongs to its generalized one. The third subset $SR_{antecedent}(T, H, X, x)$ is a set of records whose ‘*X*’ values are generalizations of ‘x’. In this case, μ(t) values become smaller than 1.0, i.e., partial membership, since a generalized concept does not necessarily belong to its specialized one completely.

Fig. 3 helps understanding the membership degree assignment. Though actual fuzzy relations for decision tree construction would have several predictor attributes, the fuzzy relation in Fig. 3 shows only one predictor attribute for simplicity. An ISA hierarchy on attribute *X* represents generalization/specialization relationships between values in the domain of X. Let us compute *SR(T, H, X, b)* for example.

By Definition 2,
$SR(T,H,X,b) = SR_{direct}(T,X,b) \cup SR_{descendent}(T,H,X,b) \cup SR_{antecedent}(T,H,X,b)$..............(1)
Since *X* values of t_9 and t_{13} are literally identical to ‘b’,
$SR_{direct}(T,X,b) = \{t_9/1.0, t_{13}/0.6\}$ ....(2)
Since ‘e’, ‘f’, and ‘g’ are descendents of ‘b’, and t_7, t_{11}, t_{15}, and t_{16} have one of them.
$SR_{descendent}(T,H,X,b) = \{t_7/0.9, t_{11}/1.0, t_{15}/1.0, t_{16}/0.9\}$..(3)
Since ‘a’ is the only antecedent of ‘b’, and t_1 and t_5 have ‘a’ on *X*,
$SR_{antecedent}(T,H,X,b) = \{t_1/\mu(t_1), t_5/\mu(t_5)\}$ ....(4)
Since DESC(a) = {b, c, d, e, f, g, h, i, j} and DESC(b) = {e, f, g},
$\mu(t_1) = \mu_T(t_1) \times Card(\{t_7/0.9, t_9/1.0, t_{11}/1.0, t_{13}/0.6, t_{15}/1.0, t_{16}/0.9\}) / Card(\{t_2/1.0, t_3/1.0, t_4/0.7, t_6/1.0, t_7/0.9, t_8/1.0, t_9/1.0, t_{10}/1.0, t_{11}/1.0, t_{12}/0.4, t_{13}/0.6, t_{14}/1.0, t_{15}/1.0, t_{16}/0.9\}) = 1.0 \times 5.4/12.5 = 0.43$ ....(5)

$\mu(t_5) = \mu_T(t_5) \times \text{Card}(\{t_7/0.9, t_9/1.0, t_{11}/1.0, t_{13}/0.6, t_{15}/1.0, t_{16}/0.9\}) / \text{Card}(\{t_2/1.0, t_3/1.0, t_4/0.7, t_6/1.0, t_7/0.9, t_8/1.0, t_9/1.0, t_{10}/1.0, t_{11}/1.0, t_{12}/0.4, t_{13}/0.6, t_{14}/1.0, t_{15}/1.0, t_{16}/0.9\}) = 0.9 \times 5.4/12.5 = 0.39$........ ....(6)

From (1), (2), (3), (4), (5), and (6),

$\text{SR}(T,H,X,b) = \{t_9/1.0, t_{13}/0.6\} \cup \{t_7/0.9, t_{11}/1.0, t_{15}/1.0, t_{16}/0.9\} \cup \{t_1/0.43, t_5/0.39\}\}$

$= \{t_1/0.43, t_5/0.39, t_7/0.9, t_9/1.0, t_{11}/1.0, t_{13}/0.6, t_{15}/1.0, t_{16}/0.9\}$

Recall that only t_9 and t_{13} are selected in conventional decision tree algorithms. But, our method also includes t_1, t_5, t_7, t_{11}, t_{15}, and t_{16} with proper membership assignment.

2.2 Extended Measures for Selecting the Best-Split Attribute

Now, let us return to the problem of selecting the best-split attribute. Recall that the best-split attribute is one that partitions the training set into the most homogeneous subsets. Though there are many measures proposed to evaluate the heterogeneity of a set, we adopt the information theoretic measure here, since it is one of the most commonly used ones in actual data mining systems. As we have regarded a fuzzy relation as a convenient representation of a training set, let us define how to measure the heterogeneity of a fuzzy relation by extending the measure of entropy [4][12].

Definition 3. (Entropy of a Fuzzy Relation).

Suppose that the domain of an attribute C is $\{c_1, \ldots, c_m\}$, and a given a fuzzy relation T is partitioned into $T^{C1}, \ldots, T^{Cm}$ according to the attribute values of C, i.e., $T^{Cj} = \{(t, \mu(t)) \mid t \in T, t.C = c_j, \mu(t) = \mu_T(t)\}$. We call C the *discriminating attribute* of the partitioning. Then the entropy of T with respect to C is denoted as *info^C(T),* and defined as follows.

$$info^C(T) = -\Sigma_{j=1,\ldots,m} [\, Card(T^{Cj})/Card(T) \times \log_2(Card(T^{Cj})/Card(T)) \,],$$

where $Card(T^{Cj}) = \Sigma_{t \in T^{Cj}} \mu_T^{Cj}(t)$, and $Card(T) = \Sigma_{t \in T} \mu_T(t)$.

Let us compute the entropy of the training set in Table 1 with respect to 'FVB'. Since the table is an ordinary relation, the membership degree of each record is 1.0. Since the domain of the 'FVB' attribute is $\{E_1, E_2, W_1, W_2\}$, the table is partitioned into four subsets as follows;

$T^{C1} = \{t_2/1.0, t_3/1.0\}$ (where 'FVB' = 'E_1'),
$T^{C2} = \{t_1/1.0, t_4/1.0, t_5/1.0\}$ (where 'FVB' = 'E_2'),
$T^{C3} = \{t_6/1.0, t_7/1.0\}$ (where 'FVB' = 'W_1'), and
$T^{C4} = \{t8/1.0, t9/1.0, t10/1.0\}$ (where 'FVB' = 'W_2').

Thus, $\text{Card}(T^{C1}) = 2.0$, $\text{Card}(T^{C2}) = 3.0$, $\text{Card}(T^{C3}) = 2.0$, and $\text{Card}(T^{C4}) = 3.0$. Since the training set itself has ten records whose membership degrees are 1.0, Card(T) = 10.0. As a result, $\text{info}^C(T) = -[\, 2.0/10.0 \times \log_2(2.0/10.0) + 3.0/10.0 \times \log_2(3.0/10.0) + 2.0/10.0 \times \log_2(2.0/10.0) + 3.0/10.0 \times \log_2(3.0/10.0) \,] = 1.97$.

We have to select the best-split attribute among three predictor attributes, 'RA', 'Gender', and 'Income'. Firstly, let us consider 'RA'. By applying the partitioning method defined in Definition 2, we become to have four subsets if we partition the training set according to 'RA' values as follows;

$T_1 = \{t_1/1.0, t_2/1.0, t_3/1.0, t_4/1.0, t_5/1.0\}$ (where 'RA' is 'East'),
$T_2 = \{t_1/0.33, t_2/0.33, t_3/1.0\}$ (where 'RA' is 'Far-East'),
$T_3 = \{t_1/0.67, t_2/0.67, t_4/1.0, t_5/1.0\}$ (where 'RA' is 'Mid-East'),
$T_4 = \{t_6/1.0, t_7/1.0, t_8/1.0, t_9/1.0, t_{10}/1.0\}$ (where 'RA' is 'West').

By applying the formula in Definition 3, we compute the entropy value of each subset with respect to the class label 'FVB'.

$\text{info}^{\text{FVB}}(T_1) = -[\,2.0/5.0 \times \log_2(2.0/5.0) + 3.0/5.0 \times \log_2(3.0/5.0)\,] = 1.37$,
$\text{info}^{\text{FVB}}(T_2) = -[\,1.33/1.66 \times \log_2(1.33/1.66) + 0.33/1.66 \times \log_2(0.33/1.66)\,] = 0.72$,
$\text{info}^{\text{FVB}}(T_3) = -[\,0.67/3.34 \times \log_2(0.67/3.34) + 2.67/3.34 \times \log_2(2.67/3.34)\,] = 0.72$,
$\text{info}^{\text{FVB}}(T_4) = -[\,2.0/5.0 \times \log_2(2.0/5.0) + 3.0/5.0 \times \log_2(3.0/5.0)\,] = 0.97$.

The normalized sum of the entropy values is (1.37 + 0.72 + 0.72 + 0.97) / (5.0 + 1.66 + 3.34 + 5.0) = 0.25. Thus, we can say that the entropy value is reduced from 1.97 to 0.25, i.e. as much as 1.72. In other words, the information gain by partitioning the training set according to the attribute 'RA' is 1.72 bits. Similarly, we can compute the information gain values of the attributes 'Gender' and 'Income'. Among them, the attribute with the highest information gain is selected as the best-split attribute.

Some algorithms such as C4.5 variations adopt an additional measure such as *split info*, to eliminate a bias in favor of predictor attributes with more distinct values. In fact, the split info is also the entropy of a set, where the discriminating attribute is the predictor attribute - 'RA' in the previous example - rather than the class label. By dividing the information gain value with split info, we can eliminate unfair advantage that attributes with more distinct values may have. Obviously, the measure of split info for a fuzzy relation can be computed with the formula in Definition 2.

3 Assigning Class Labels with Decision Trees

Suppose that we obtain a decision tree as shown in Fig. 4 after applying the proposed method. The number attached to each terminal node represents the decision confidence [4]. Also suppose that we have a target record such that (East, Male, 85k, Unknown). Notice that the last attribute 'FVB', i.e. the class label, is unknown. Our goal is to determine the 'FVB' attribute of the record. Conventional methods simply follow the first branch from the root node and the second branch of the 'Income' node to arrive at the assignment of 'E_1' with the confidence of 67%.

However, we have to take account of the fact that 'East' may be 'Far-East' in fact. Suppose that 85% of records in the training set whose 'Residential Area' attributes are specializations of 'East' have 'Far-East' values. Then, we can follow the third branch of the root node, and assign 'E_2' with the confidence 85% × 98% = 83%. The assignment of 'E_2' has the confidence of 83% while the assignment of 'E_1' has the confidence of 67%. According to the principle of majority voting, we finally assign 'E_2'

with the confidence of 83%. The algorithm in Fig. 5 presents the assignment process precisely.

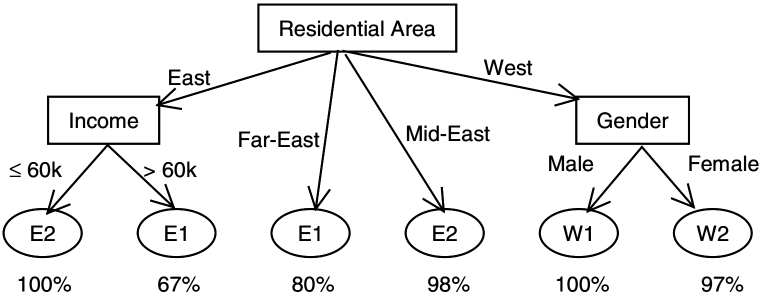


Fig. 4. An example decision tree with confidence levels

```
AssignClass(DecisionNode Attr, Record R) {
(1)    If Attr is a terminal node, return (Attr.Decision);
(2)    Child = the node followed by the branch with a label identical to R.Attr.
(3)    Answer = AssignClass(Child, R);
(4)    For each branch with a label
                that is a generalization of R.Attr {
(5)             Child  = the node followed by this branch;
(6)             Temp = AssignClass(Child, R);
(7)             If Temp.Confidence > Answer.Confidence,
                       then Answer  = Temp;      }
(8)    For each branch with a label
              that is a specialization of R.Attr  {
(9)             Child = the node followed by this branch;
(10)            Weight = the ratio of this specialization;
(11)            Temp = Weight × AssignClass (Child, R);
(12)            If Temp.Confidence > Answer.Confidence,
                       then Answer = Temp;    }
(13)  return(Answer); }
```

Fig. 5. Class assignment with multiple abstraction levels of data

The conventional assignment process consists from Line (1) to Line (3). To accommodate generalizations and specializations, the loop from Line (4) to Line (7) and the loop from Line (8) to Line (12) are added, respectively. Among all the assignments, that of the maximum confidence is chosen for the final answer.

4 Performance Evaluation

We have conducted experiments with a benchmark data set to analyze classification accuracy of the proposed method. The experimental results show that the proposed method reduces classification error rates significantly when a data set includes multiple abstraction levels of data. The experiment is performed on an Axil-Ultima 167 workstation with Solaris 7, 128 MB RAM, and 10G Hard Disk.

To show that the proposed method outperforms the existing methods such as C4.5, it is necessary to have benchmark data sets containing multiple abstraction levels of data. However, handling of multiple abstraction levels of data in decision tree classification has been seldom studied so far, it is hard to obtain such data sets from public sources. Instead, we randomly distribute multiple abstraction levels of data across a public benchmark data set obtained from UCI Machine Learning Repository. The original benchmark data set contains 48,842 records with six continuous and eight categorical attributes. It has demographic information to classify high-income and low-income residents with the threshold income of $50,000. Since all the values in each attribute are in the same abstraction level, we select values in a random fashion, and substitute them with more general or specific abstraction levels. For example, an attribute value ‘federal government’ in ‘workclass’ attribute is substituted to a more general value ‘government’. By controlling the ratio of data to substitute, we can generate a collection of data sets with various heterogeneity degrees in their abstraction levels.

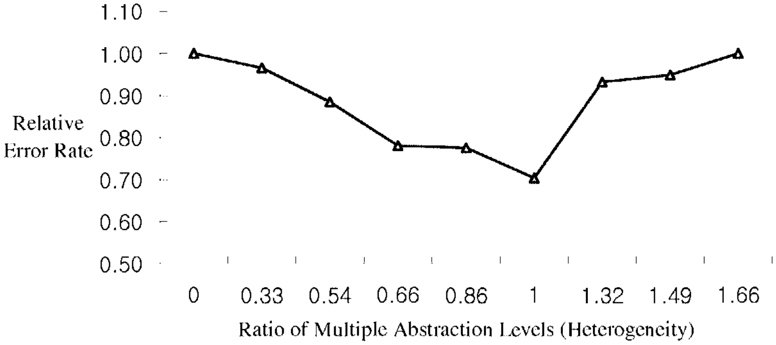


Fig. 6. Relative error rates with respect to the heterogeneity in abstraction levels

Fig. 6 shows how classification error accuracy is affected by the heterogeneity degrees of data sets. The vertical axis represents relative error rates of the proposed method with respect to the C4.5 algorithm, i.e., the error rate of the proposed method / the error rate of C4.5. When the heterogeneity of the data set is zero, in other words, all the data is in the same abstraction level, the classification error rate of the proposed method is the same as C4.5. As the heterogeneity increases, the accuracy improvement of the proposed method becomes higher up to 30%. However, notice that when the heterogeneity of the data set becomes larger than 1.0, the accuracy improvement decreases. This unexpected situation occurs because attributes with too diverse abstraction levels of data cannot be selected as best-split attributes. In other words, the resulting decision trees do not include such attributes in their decision nodes. Consequently, handling multiple abstraction levels cannot take effect when the heterogeneity of the data set is too high.

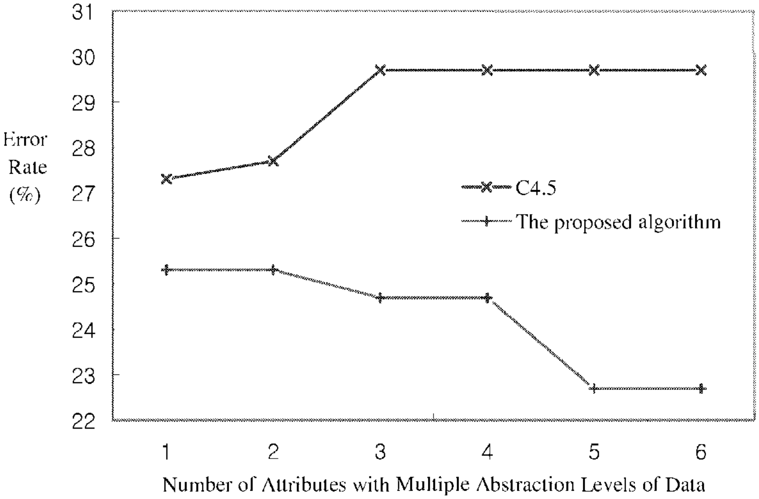


Fig. 7. Error rates with respect to the number of attributes with multiple abstraction levels of data

To analyze the effect of the number of attributes with multiple abstraction levels, we also generate another collection of data sets by controlling data substitution to the original UCI data set. Fig. 7 shows how classification error accuracy is affected as the number of attributes with multiple abstraction levels increases. As expected, the accuracy improvement becomes larger as the number attributes with multiple abstraction levels increases.

5 Concluding Remarks

This paper has introduced the multiple abstraction level problem in decision tree classification, and proposed a method to deal with it. It has explained that equalizing abstraction levels by force using data cleansing tools cannot solve the problem. We utilize the information as it is, but accommodate the generalization/specialization relationship between data values in both of the construction and assignment phases. While a specialized value is compatible with its generalization completely, a generalized value is compatible with each of its specializations as much as the portion the specialization occupies among all existing specializations. In order to represent this partial compatibility, it adopts the notion of fuzzy relations. The experimental results conducted with a modified benchmark data set originated from UCI Machine Learning Repository have shown that the proposed method reduces classification error rates significantly when a data set includes multiple abstraction levels of data.

We have been applying the proposed method to an integrated network alarm management system that suffers from different abstraction levels delivered from local network management agents.

References

1. J. Gehrke, R. Ramakrishinan, and V. Ganti, “RainForest - A Framework for Fast Decision Tree Construction of Large Datasets,” *Data Mining and Knowledge Discovery*, 4, pp. 127-162, 2000
2. J. Gehrke, V. Ganti, R. Ramakrishnan, and W.-Y Loh, “BOAT – Optimistic Decision Tree Construction,” *In Proc. of ACM SIGMOD Conf.*, Philadelphia, Pennsylvania, June 1999, pp. 169-180
3. M. Berry and G. Linoff, *Data Mining Techniques – For Marketing, Sales, and Customer Support*, Wiley and Sons, 1997
4. J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Pub., 1993.
5. K. Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen, “Knowledge Discovery from Telecommunication Network Alarm Databases,” *In Proc. of the 12th International Conference on Data Engineering*, New Orleans, Louisiana, February 1996, pp. 115-122,
6. L. English, *Improving Data Warehouse and Business Information Quality-Method for Reducing Costs and Increasing Profits*, Wiley & Sons, 1999
7. R. Wang, V. Storey and C. Firth, “A Framework for Analysis of Data Quality Research,” *IEEE Transactions on Knowledge and Engineering*, 7(4), pp. 623-640, 1995
8. Trillium Software System, “A Practical Guide to Achieving Enterprise Data Quality,” White Paper, Trillium Software, 1998.
9. J. Williams, “Tools for Traveling Data,” *DBMS*, Miller Freeman Inc., June 1997
10. Vality Technology Inc., “The Five Legacy Data Contaminants You Will Encounter in Your Warehouse Migration,” White Paper, Vality Technology Inc., 1998
11. G. Klir and T. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall Int’l Inc., 1988
12. C. Shannon, “The Mathematical Theory of Communication,” *The Bell System Tech. Jour.*, 27, 1948
13. C. Batini, S. Ceri, and S. Navathe, *Conceptual Database Design*, Benjamin Cummings, Inc., 1992

Supporting Information Integration with Autonomous Agents

S. Bergamaschi[1][2], G. Cabri[1], F. Guerra[1],
L. Leonardi[1], M. Vincini[1], and F. Zambonelli[1]

[1] Università di Modena e Reggio Emilia
DSI - Via Vignolese 905, 41100 Modena
[2] CSITE-CNR Bologna
V.le Risorgimento 2, 40136 Bologna
{sonia.bergamaschi,giacomo.cabri,francesco.guerra,letizia.leonardi,
maurizio.vincini,franco.zambonelli}@unimo.it

Abstract. The large amount of information that is spread over the Internet is an important resource for all people but also introduces some issues that must be faced. The dynamism and the uncertainty of the Internet, along with the heterogeneity of the sources of information are the two main challenges for the today's technologies. This paper proposes an approach based on mobile agents integrated in an information integration infrastructure. Mobile agents can significantly improve the design and the development of Internet applications thanks to their characteristics of autonomy and adaptability to open and distributed environments, such as the Internet. MOMIS (Mediator envirOnment for Multiple Information Sources) is an infrastructure for semi-automatic information integration that deals with the integration and query of multiple, heterogeneous information sources (relational, object, XML and semi-structured sources). The aim of this paper is to show the advantage of the introduction in the MOMIS infrastructure of intelligent and mobile software agents for the autonomous management and coordination of the integration and query processes over heterogeneous sources.

1 Introduction

Providing an integrated access to multiple heterogeneous sources is a challenging issue in global information systems for cooperation and interoperability. The problems that have to be faced in this field are mainly due to both structural and application heterogeneity, as well as to the lack of a common ontology, causing semantic differences between information sources. Complicating factors with respect to conventional view integration techniques [3] are related to the fact that semantic heterogeneity occurs on the large-scale. This heterogeneity involves terminology, structure, and domain of the sources, with respect to geographical, organizational, and functional aspects of the information use [25]. Moreover, to meet the requirements of global, Internet-based information systems, it is important that the tools developed for supporting these activities are semi-automatic and scalable as much as possible.

To face the issues related to scalability in the large-scale, in this paper we propose the exploitation of *mobile agents* in the information integration area, and, in particular,

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 88–99, 2001.

their integration in the MOMIS infrastructure, which focuses on capturing and reasoning about semantic aspects of schema descriptions of heterogeneous information sources for supporting integration and query optimization.

Mobile agents are a quite recent technology. They can significantly improve the design and the development of Internet applications thanks to their characteristics [23]. The agency feature permits them to exhibit a high degree of autonomy with regard to the users: they try to carry out their tasks in a *proactive* way, *reacting* to the changes of the environment they are hosted [31]. The mobility feature takes several advantages in a wide and unreliable environment such as the Internet. First, mobile agents can significantly save bandwidth, by moving locally to the resources they need and by carrying the code to manage them. Moreover, mobile agents can deal with non-continuous network connection and, as a consequence, they intrinsically suit mobile computing systems. All these features are particularly suitable in the information retrieval area [8].

MOMIS [5] (Mediator envirOnment for Multiple Information Sources) is an infrastructure for information integration that deals with the integration and query of multiple, heterogeneous information sources, containing structured and semistructured data. MOMIS is a support system for semi-automatic integration of heterogeneous sources schema (relational, object, XML and semi-structured sources); it carries out integration following a semantic approach which uses Description logics-based techniques, clustering techniques and an ODM-ODMG [13] extended model to represent extracted and integrated information, ODM$_{I^3}$. Using the ODL$_{I^3}$ language, referred to the ODM$_{I^3}$ model, it is possible to describe the sources (local schema) and MOMIS supports the designer in the generation of an integrated view of all the sources (Global Virtual View), which is expressed using XML standard. The use of XML in the definition of the Global Virtual View lets to use MOMIS infrastructure with other open integration information systems by the interchange of XML data files.

In particular, we show the advantage of the introduction in the MOMIS architecture of intelligent and mobile software agents for the autonomous management and coordination of the integration and query processes over heterogeneous data sources.

2 System Architecture References

Like other integration projects [1,28], MOMIS follows a "semantic approach" to information integration based on the conceptual schema, or metadata, of the information sources, and on the the I^3 architecture [22] (see Figure 1). The system is composed by the following components:

1. a common data model, ODM$_{I^3}$, which is defined according to the ODL$_{I^3}$ language, to describe source schemas for integration purposes. ODM$_{I^3}$ and ODL$_{I^3}$ have been defined in MOMIS as subset of the corresponding ones in ODMG, following the proposal for a standard mediator language developed by the I^3/POB working group [7]. In addition, ODL$_{I^3}$ introduces new constructors to support the semantic integration process;
2. *Wrapper agents*, placed over each sources, translate metadata descriptions of the sources into the common ODL$_{I^3}$ representation, translate (reformulate) a global

query expressed in the OQL_{I^3}[1] query language into queries expressed in the sources languages and export query result data set;

3. a *Mediator*, which is composed of two components in its turn: the *SI-Designer* and the *Query Manager* (QM). The SI-Designer component processes and integrates ODL_{I^3} descriptions received from wrapper agents to derive the integrated representation of the information sources. The QM component performs query processing and optimization. Starting from each query posed by the user on the Global Schema, the QM generates OQL_{I^3} queries that are sent to wrapper agents by exploiting *query agents*, which are mobile agents. QM automatically generates the translation of the query into a corresponding set of sub-queries for the sources and synthesizes a unified global answer for the user.

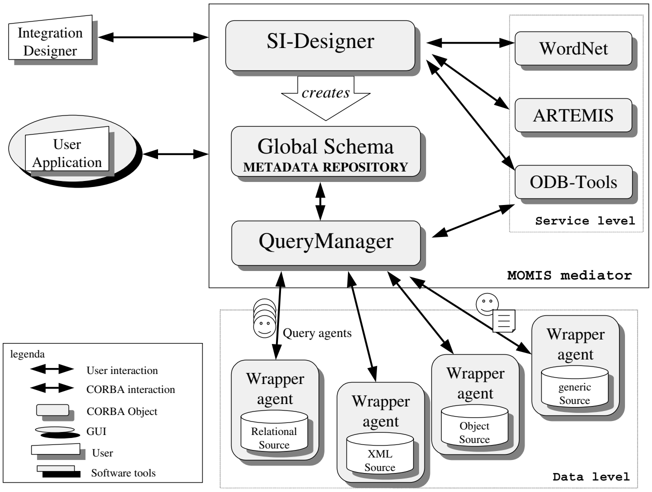


Fig. 1. The MOMIS system architecture

The original contribution of MOMIS is related to the availability of a set of techniques for the designer to face common problems that arise when integrating pre-existing information sources, containing both semistructured and structured data.
MOMIS provides the capability of explicitly introducing many kinds of knowledge for integration, such as integrity constraints, intra- and inter-source intensional and extensional relationships, and designer supplied domain knowledge. A *Common Thesaurus*, which has the role of a shared ontology of the source is built in a semi-automatic way. The *Common Thesaurus* is a set of intra and inter-schema intensional and extensional relationships, describing the knowledge about classes and attributes of sources schemas; it provides a reference on which to base the identification of classes candidate to integration and subsequent derivation of their global representation.

MOMIS supports information integration in the creation of an integrated view of all sources (Global Virtual View) in a way automated as much as possible and performs revision and validation of the various kinds of knowledge used for the integration. To this

[1] OQL_{I^3} is a subset of OQL-ODMG.

end, MOMIS combines reasoning capabilities of Description Logics with affinity-based clustering techniques, by exploiting a common ontology for the sources constructed using lexical knowledge from WordNet [21,26] and validated integration knowledge.

The Global Virtual View is expressed by using XML standard, to guarantee the interoperability with other open integration system prototype.

2.1 The Roles of the Agent

In our architecture, agents have two main roles. On the one hand, the source wrappers are agents that converts the source information and react to source changes. On the other hand, the QM exploits mobile agents to carry out the retrieval of information.

When a new source has to be integrated in MOMIS, a mobile agent moves to the site where the source resides. Such agent checks the source type and autonomously installs the needed driver to convert the source information. Moreover, a fixed agent, called *wrapper agent*, is left at this site to preside the source. Besides interacting with query agents as described later, the wrapper agents monitor the changes that may occur in the data structure of sources; when a change occurs in a source, the corresponding wrapper agent creates an appropriate mobile agent that moves to the MOMIS site to inform about the new structure, so as to update the Global Schema.

The QM works as follows. When a user queries the global schema, it generates a set of sub-queries to be made to the single sources. To this purpose, it can exploit one or more *query agents*, which move to the source sites where they interact with the wrapper agents. The choice of the number of query agents to use can be determined by analyzing each query. In some cases, it is better to delegate the search to a single query agent, which performs a "trip" visiting each source site: it can start from the source that is supposed to reduce the further searches in the most significant way, then continue to visit source sites, performing queries on the basis of the already-found information. In other cases, sub-queries are likely to be quite independent, so it is better to delegate several query agents, one for each source site: in this way the searches are performed concurrently with a high degree of parallelism. In any case, the peculiar features of mobile agents are exploited and make the MOMIS searches suitable to the Internet environment. First, by moving locally to the source site, a query agent permits to significantly save bandwidth, because it is not necessary to transfer a large amount of data, but the search computation is performed locally where the data resides. Second, MOMIS can queries also sources that do not have continuous connections: the query agent moves to the source site when the connection is available, performs locally the search even if the connection is unstable or unavailable, and then returns to the QM site as soon as the connection is available again. Finally, this fits well mobile computing, where mobile devices (which can host users, MOMIS, or sources) do not have permanent connections with the fixed network infrastructure.

The interaction between agents can occur by using several protocols. See [10] for a comparison among different kinds of coordination for Internet applications based on mobile agents; an approach that is gaining ground more and more is the one based on *programmable tuple spaces* [9].

```
<!ELEMENT fiat(car*)>
<!ELEMENT car(name,engine,dimensions,tires,
      performance,price)>
<!ELEMENT engine(name,cylinders?,layout?,
      capacity_cc?,compression_ratio?,
      power_kw, fuel_system)>
<!ELEMENT dimensions(length,width,heigth,
      luggage_capacity)>
<!ELEMENT performance (urban_consumption,
      combined_consumption,speed)>
<!ELEMENT name (#pcdata)>
...
```

Fig. 2. Fiat database (FIAT)

```
Vehicle(name, length, width, height)
Motor(cod_m, type, compression_ratio,
       KW, lubrification, emission)
Fuel_Consumption(name, cod_m, drive_trains,
       city_km_l, highway_km_l )
Model(name, cod_m, tires, steering, price)
```

Fig. 3. Volkswagen database (VW)

2.2 Running Example

In order to illustrate how the MOMIS approach works, we will use the following example of integration in the Car manufacturing catalogs, involving two different data-sources that collect information about vehicle. The first data-source is the FIAT catalog, containing semistructured XML informations about cars of the Italian car factory (see Figure 2).

The second data-source is the Volkswagen database (VW) reported in Figure 3, a relational database containing information about this kind of car. Both database schemata are built by analyzing the web site of this factory.

3 Integration Process

The MOMIS approach to perform Global Virtual View is articulated in the following phases:

1. *Generation of a Common Thesaurus.*
   The *Common Thesaurus* is a set of terminological intensional and extensional relationships, describing intra and inter-schema knowledge about classes and attributes of sources schemas. We express intra and inter-schema knowledge in form of terminological and extensional relationships (em synonymy, *hypernymy* and *relationship*) between classes and/or attribute names. In this phase, to extract lexicon derived relationships the WordNet database is used.
2. *Affinity analysis of classes.*
   Relationships in the *Common Thesaurus* are used to evaluate the level of *affinity*

between classes intra and inter sources. The concept of affinity is introduced to formalize the kind of relationships that can occur between classes from the integration point of view. The affinity of two classes is established by means of affinity coefficients based on class names, class structures and relationships in *Common Thesaurus*.

3. *Clustering classes.*
   Classes with affinity in different sources are grouped together in clusters using hierarchical clustering techniques. The goal is to identify the classes that have to be integrated since describing the same or semantically related information.
4. *Generation of the mediated schema.*
   Unification of affinity clusters leads to the construction of the predicted schema. A class is defined for each cluster, which is representative of all cluster classes and is characterized by the union of their attributes. The global schema for the analyzed sources is composed of all classes derived from clusters, and is the basis for posing queries against the sources.

In the following we introduce the generation of the *Common Thesaurus* associated with the example domain, starting from the lexicon relationships definition by using Wordnet.

3.1 Generation of a *Common Thesaurus*

The *Common Thesaurus* is a set of terminological intensional and extensional relationships, describing intra and inter-schema knowledge about classes and attributes of sources schemas; it provides a reference to define the identification of classes candidate to integration and subsequent derivation of their global representation. In the Common Thesaurus, we express knowledge in form of intensional relationships (SYN, BT, NT, and RT) and extensional relationships (SYN$_{ext}$, BT$_{ext}$, and NT$_{ext}$ between classes and/or attribute names.

The Common Thesaurus is constructed through an incremental process during which relationships are added in the following order:

1. schema-derived relationships
2. lexical-derived relationships
3. designer-supplied relationships
4. inferred relationships

All these relationships are added to the Common Thesaurus and thus considered in the subsequent phase of construction of Global Schema. For a more detailed description of the above described process see [6].

Terminological relationships defined in each step hold at the intensional level by definition. Furthermore, in each of the above step the designer may "strengthen" a terminological relationships SYN, BT and NT between two classes C_1 and C_2 by establishing that they hold also at the extensional level, thus defining also an extensional relationship. The specification of an extensional relationship, on one hand, implies the insertion of a corresponding intensional relationship in the Common Thesaurus and, on the other hand, enable subsumption computation (i.e., inferred relationships) and consistency checking between two classes the C_1 and C_2.

Global Class and Mapping Tables

Starting from the output of the cluster generation, we define, for each cluster, a *Global Class* that represents the mediated view of all the classes of the cluster. For each global class a set of *global attributes* and, for each of them, the intensional mappings with the *local attributes* (i.e. the attributes of the local classes belonging to the cluster) are given [2].

Shortly, we can say that the global attributes are obtained in two steps: (1) Union of the attributes of all the classes belonging to the cluster; (2) Fusion of the "similar" attributes; in this step redundancies are eliminated in a semi–automatic way taking into account the relationships stored in the *Common Thesaurus*. For each global class a persistent *mapping-table* storing all the intensional mappings is generated; it is a table whose columns represent the set of the local classes which belong to the cluster and whose rows represent the global attributes.

The final step of the integration process provides the export of the Global Virtual View into a XML DTD, by adding the appropriate XML TAGs to represent the mapping table relationships. The use of XML in the definition of the Global Virtual View lets to use MOMIS infrastructure with other open integration information system by the interchange of XML data files. In addition, the Common Thesaurus is translated into XML file, so that MOMIS may provides a shared ontology that can be used by different semantic ontology languages [17,16].

In the referring example the following *Global Class* are defined:

- Vehicle: contains the `Vehicle`, `Model`, `car` source classes and a global attributes indicates the source name;
- Engine: contains the `engine`, `Motor` source classes;
- Performance: contains the `performance`, `Fuel_Consumption` source classes;
- Dimensions: contains the `dimensions` source class.

Dynamic Local Schema Modification

Now, let suppose that a modification occurs in a local source, for example in the FIAT DTD a new element `truck` is added:

```
<!ELEMENT truck(name, engine, dimensions, price, capacity)>
```

In this case the *local wrapper agent* that resides at the FIAT source creates a mobile agent that goes to the MOMIS site and there checks the permission to modify the Global Schema: if the agent is enabled it directly performs the integration phases (i.e. *Common Thesaurus*, *Clustering*, *Mapping Generation*) caused by the schema modification and notifies to the *Integration Designer* its actions. Otherwise, if the agent has not enough rights, it delegates the whole integration re-process to the *Integration Designer*.

In our example the new `truck` element is inserted by the em local wrapper agent in the Vehicle *Global Class* and the *mapping* is performed.

[2] For a detailed description of the mappings selection and of the tool SI-Designer which assist the designer in this integration phase see [4].

3.2 The Query Manager Agents

The user application interacts with MOMIS to query the Global Virtual View by using the OQL$_{I^3}$ language. This phase is performed by the QM component that generates the OQL$_{I^3}$ queries for wrapper agents, starting from a global OQL$_{I^3}$ query formulated by the user on the global schema. Using Description Logics techniques, the QM component can generate in an automatic way the translation of the generic OQL$_{I^3}$ query into different sub-query, one for each involved local source. The *query agents* are mobile agents in charge of bringing such sub-queries to the data source sites and there they interact with the local wrapper agents to carry out the queries; then they report the data result to the QM. To achieve the global answer, the QM has to merge each local sub-queries result into a unified data set. This process involves the solution of redundancy and reconciliation problems, due to the incomplete and overlapping information available on the local sources, i.e. *Object Fusion* [27].

For example, over the Global Virtual View build in the previous section we should “retrieve the car name and price for power levels present in every sources”, that is obtained by the following query:

```
Q: select V1.name, V1.power, V1.price,
   from Vehicle V1
   where 1 <= (select count(*)
               from  where Vehicle V2
               where V2.power = V1.power
               and   V2.source <> V1.source)
```

Processing the above global query would individuate all the local classes involved: VW.Vehicle, VW.Model, VW.Motor, FIAT.car, FIAT.engine.

The QM, by the query reformulation process [6], defines the following local queries (QL1 expressed by SQL and QL2 by XQuery [14]):

```
QL1: select M1.name, Motor.KW, M1.price
     from VW.Model M1, VW.Motor
     where M1.cod_m = Motor.cod_m

QL2: FOR $C in document("fiat.xml")//car
     RETURN
     <car>
           <name>$C/name</name>
           <price>$C/price</price>
           <kw>$C/engine/power_kw</kw>
     </car>
```

This process may be executed in parallel by two query agents that move to the two local sources. After performs the query, each query agents returns the data result to the MOMIS QM that fuses the two data-set obtaining the final result (in the example the fusion is obtained by join the data-set on the power attribute).

Following a more autonomous approach, for this class of queries (i.e., where a fusion of data derived by different local sources is necessary) the data process should be moved to the local systems to minimize the data transfer. This is performed by providing query agents with appropriate local service queries that obtain an intermediate result, which can be sent to other query agents to perform fusion at the local sites.

For example the following service queries using local query language are added:

```
QS1: select DISTINCT Motor.KW
   from VW.Motor

QS2: FOR $KW in distinct(document("fiat.xml")//car)
     RETURN
        <kw>$KW</kw>
```

These service queries are executed by the query agents to the local sources and the data results are exchanged with the other agents. In each single source the intermediate data sets are joined to the local queries to obtain the fused result (the join attributes are extracted from the mapping tables of which each query agent has a copy):

```
Q1: select DISTINCT QL1.name, QL1.KW,  QL1.price
    from QL1, QS2
    where QL1.KW = QS2.power_kw

Q2: select DISTINCT QL2.name, QL2.power_kw,  QL2.price
    from QL2, QS1
    where QL2.power_kw = QS1.KW
```

In this way, the union of Q1 and Q2 results are the same obtained by the first shown approach, while the data transfer amount with the MOMIS central site is drastically reduced, since only the request data are moved by the agent.

This example has shown the effective use and the related advantages of the exploitation of agents in our infrastructure for the coordination of data retrieval and management.

4 Related Work

As far as we know, there are few agent-based approaches in the area of information integration.

A relevant one is the MCC InfoSleuth(tm) [19] Project 1. It is an agent-based system for information gathering and analysis tasks performed over networks of autonomous information sources. A key motivation of the InfoSleuth system is that real information gathering applications require long-running monitoring and integration of information at various levels of abstraction. To this end, InfoSleuth agents enable a loose integration of technologies allowing: (1) extraction of semantic concepts from autonomous information sources; (2) registration and integration of semantically annotated information from diverse sources; and (3) temporal monitoring, information routing, and identification of trends appearing across sources in the information network. Another experience is the RETSINA multi-agent infrastructure for in-context information retrieval [29]. In particular the LARKS description language [30] is defined to realize the agent matchmaking process (both at syntactic and semantic level) by using several different filters: Context, Profile, Similarity, Signature and Constraint matching. Differently from our approach, both InfoSleuth and RETSINA does not take advantage from the mobility feature of the agents, which we consider fundamental in a dynamic and uncertain environment such as the Internet.

In the area of heterogeneous information integration, many projects based on a mediator architecture have been developed. The mediator-based TSIMMIS project [15] follows a 'structural' approach and uses a self-describing model (OEM) to represent heterogeneous data sources, the MSL (Mediator Specification Language)rule to enforce source integration and pattern matching techniques to perform a predefined set of queries based on a query template. Differently from MOMIS proposal, in TSIMMIS only the predefined queries may be executed and for each source modification a manually mediator rules rewriting must be performed.

The GARLIC project [12] builds up on a complex wrapper architecture to describe the local sources with an OO language (GDL), and on the definition of Garlic Complex Objects to manually unify the local sources to define a global schema.
The SIMS project [2] proposes to create a global schema definition by exploiting the use of Description Logics (i.e., the LOOM language) for describing information sources. The use of a global schema allows both GARLIC and SIMS projects to support every possible user queries on the schema instead of a predefined subset of them.
The Information Manifold system [24] provides a source independent and query independent mediator. The input schema of Information Manifold is a set of descriptions of the sources. Given a query, the system will create a plan for answering the query using the underlying source descriptions. Algorithms to decide the useful information sources and to generate the query plan have been implemented. The integrated schema is defined mainly manually by the designer, while in our approach it is tool-supported.
Infomaster [20] provides integrated access to multiple distributed heterogeneous information sources giving the illusion of a centralized, homogeneous information system. It is based on a global schema, completely modeled by the user, and a core system that dynamically determines an efficient plan to answer the user's queries by using translation rules to harmonize possible heterogeneities across the sources. The main difference of these project w.r.t. MOMIS is the lack of a tool aid-support for the designer in the integration process.

5 Conclusions and Future Work

This paper has presented how a system for information integration can be improved by the exploitation of mobile agents. In particular, agents are useful in the management of the sources, which, in an open and dynamic scenario like the Internet, can be spread and can change in a uncontrolled way. The mobility feature permits to overcome the limitations of the traditional approaches of distributed systems.

With regard to future work, we sketch some research directions.

The first one relates to the dynamic integration of information sources. Currently, when a new source is added (or when is deleted), the MOMIS system has to be restarted. Our aim is to allow source integration at runtime, possibly by exploiting mobile agents that search for interesting new sources or check the request of an administrator for integrate her/his new source. This will permit to face the high degree of dynamism of the Internet.

Then, we are evaluating which further components of the system can be modeled as agents, and, in particular, which ones can take advantage from the mobility feature. The fact that some components may be mobile permits to deploy the whole system in a more flexible and adaptable way.

Finally, an area that is worth to be explored is the interaction between agents, which can occur in different ways and with different languages. On the one hand, complex languages for the knowledge exchange have been proposed [18]; on the other hand, mobility of agents promotes the adoption of simple and uncoupled coordination protocols [11].

Acknowledgements. Work supported by Italian Ministry for University and Science and Technology Research (MURST) within the project frameworks of "MUSIQUE: Infrastructure for QoS in Web Multimedia Services with Heterogeneous Access" and "D2I: Integration, Warehousing, and Mining of Heterogeneous Data Sources", and by the University of Modena and Reggio Emilia with a fund for young researchers.

References

1. Y. Arens, C.Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.
2. Y. Arens, C. A. Knoblock, and C. Hsu. Query processing in the sims information mediator. *Advanced Planning Technology*, 1996.
3. C. Batini, M. Lenzerini, and S. B. Navathe. A comprehensive analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):322–364, 1986.
4. I. Benetti, D.Beneventano, S.Bergamaschi, A. Corni, F. Guerra, and G. Malvezzi. Si-designer: a tool for intelligent integration of information. *International Conference on System Sciences (HICSS2001)*, January 2001.
5. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: The momis project demonstration. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September, 2000, Cairo, Egypt*, pages 611–614. Morgan Kaufmann, 2000.
6. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogenous information sources. *Journal of Data and Knowledge Engineering*, 36(3):215–249, 2001.
7. P. Buneman, L. Raschid, and J. Ullman. Mediator languages - a proposal for a standard, April 1996. Available at ftp://ftp.umiacs.umd.edu/pub/ONRrept/medmodel96.ps.
8. G. Cabri, L. Leonardi, and F. Zambonelli. Agents for Information Retrieval: Issues of Mobility and Coordination. *Journal of Systems Architecture*, 46(15):1419–1433, December 2000.
9. G. Cabri, L. Leonardi, and F. Zambonelli. MARS: A Programmable Coordination Architecture for Mobile Agents. *IEEE Internet Computing*, 4(4):26–35, July/August 2000.

10. G. Cabri, L. Leonardi, and F. Zambonelli. Mobile-agent coordination models for internet applications. *IEEE Computer*, 33(2):82–89, February 2000.
11. G. Cabri, L. Leonardi, and F. Zambonelli. XML Dataspaces for the Coordination of Internet Agents. *Applied Artificial Intelligence*, 15(1):35–58, January 2001.
12. M.J. Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J.H. Williams, and E.L. Wimmers. Object exchange across heterogeneous information sources. Technical report, Stanford Univ., 1994.
13. R. G. G. Cattell, editor. *The Object Database Standard: ODMG 3.0.* Morgan Kaufmann Publishers, San Mateo, CA, 2000.
14. D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. Xquery: A query language for xml. In *W3C Working Draft 15 February 2001*, Feb 2001.
15. S. Chawathe, Garcia Molina, H., J. Hammer, K.Ireland, Y. Papakostantinou, J.Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *IPSJ Conference, Tokyo, Japan*, 1994.
16. DAML Joint Committee. Daml Project. Available at http://www.daml.org.
17. D. Fensel, I. Horrocks, F. van Harmelen, S. Decker, M. Erdmann, and M. Klein. OIL in a nutshell. In *Proceedings of the European Knowledge Acquisition Conference (EKAW-2000)*, Lecture Notes In Artificial Intelligence. Springer-Verlag, 2000. To appear.
18. Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In Jeffrey M. Bradshaw, editor, *Software Agents*, chapter 14, pages 291–316. AAAI Press / The MIT Press, 1997.
19. J. Fowler, B. Perry, M. H. Nodine, and B. Bargmeyer. Agent-based semantic interoperability in infosleuth. *SIGMOD Record*, 28(1):60–67, 1999.
20. M. R. Genesereth, A. M. Keller, and O. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD Conference*, 1997.
21. J. Gilarranz, J. Gonzalo, and F. Verdejo. Using the eurowordnet multilingual semantic database. In *Proc. of AAAI-96 Spring Symposium Cross-Language Text and Speech Retrieval*, 1996.
22. R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.
23. Neeran M. Karnik and Anand R. Tripathi. Design issues in mobile-agent programming systems. *IEEE Concurrency*, 6(3):52–61, July/September 1998.
24. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB 1996*, pages 251–262, 1996.
25. S. E. Madnick. From vldb to vmldb (very many large data bases): Dealing with large-scale semantic heterogeneity. In *VLDB Int. Conf.*, pages 11–16, 1995.
26. A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
27. Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *VLDB Int. Conf.*, Bombay, India, September 1996.
28. M.T. Roth and P. Scharz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. of the 23rd Int. Conf. on Very Large Databases*, Athens, Greece, 1997.
29. K. Sykara. In-context information management truough adaptative collaboration of intelligent agents. In *Intelligent Information Agents*. M. Klusch Ed. - Springer, 1999.
30. K. Sykara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Records*, 28(1), March 1999.
31. F. Zambonelli, N. R. Jennings, A. Omicini, and M. J. Wooldridge. Agent-Oriented Software Engineering for Internet Applications. In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents*. Springer-Verlag, March 2001.

Using Agents in Performing Multi-site Queries

Elisabetta Di Nitto[1], Carlo Ghezzi[1], Maurizio Sabba[2], and Paolo Selvini[2]

[1] Politecnico di Milano, Dipartimento di Elettronica e Informazione,
Piazza L. Da Vinci, 32 20133 Milano
[2] CEFRIEL, Via Fucini, 2 20133 Milano

Abstract. In this paper we present a runtime platform and a language supporting infomediaries (information intermediaries) in tailoring information extracted from the web to satisfy end user requirements. The language is based on a XML query language and adapts it to the case where information is obtained by aggregating parts distributed over multiple sites. The platform is based on mobile agents and acts as an intermediary between users and information providers.

1 Introduction

The role of information intermediaries – *infomediaries* [3] – on the web is emerging. They allow users, otherwise overwelmed by any kind of information, to get in contact with proper providers. Sometimes they also support the execution of business transactions.

In this paper we present a platform for infomediaries that supports a powerful approach to extract and assemble information according to users' needs. The platform offers to the user *searching agents* that are instructed with the objective of the search and are able to produce a result by executing multiple queries on different sites, each of which provides part of the information required. Searching agents can either perform the queries on a remote server or they can move from a data source to another to collect results, if this is more appropriate for performance reasons. We assume that data are described in XML and that queries are executed by an XML query engine [1].

The work we present is part of the EU-IST OPELIX project (http://www.opelix.org), which aims at supporting *i-commerce.* By i-commerce we mean electronic commerce of information, i.e., goods that can be handled and delivered electronically (e.g., documents, tickets, and reservations). OPELIX aims at defining a number of composable services (search, negotiation, user profiling, information pushing, advertising, security, and copyright protection) that enable the development of intermediaries.

The rest of the paper is structured as follows. Section 2 presents an example of information search that is used in the rest of the paper. Section 3 provides a general overview of the searching infrastructure we have developed. Sections 4 and 5 present the language to instruct searching agent and the architecture of our prototype, respectively. Finally, Section 6 draws the conclusions.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 100–105, 2001.

2 A Running Example of Information Search

Let us consider a network of web sites providing information about shows, flight tickets, and hotel reservation. Such information can have a non-homogeneous structure. Each of the sites supports transactions on only one kind of data. Tables 1 and 2 show a partial snapshot of the state of two of these sites.

Table 1. Theatre performances database hosted by site site1.domain1.com.

name	city	date	price
Hamlet	London	05/03/2001	200 EUR
La Traviata	Roma	07/07/2001	100 EUR
La Traviata	Milano	07/31/2001	400 EUR

Let us suppose a user wants to attend an opera performance, e.g. "La Traviata", no matter where it is staged, provided that a flight directly connecting his/her home city (New York) to the performance city exists (with a schedule compatible with the performance) and that a five stars hotel in that city has a suite available.

It is clear that our user will have to browse the network of sellers, looking for each piece of information individually, taking care of all the compatibilities and matchings among such data.

Table 2. Flights database hosted by site site2.domain2.com.

depart/date	depart/place	arrival/date	arrival/place	code	price
07/05/2001	New York	07/08/2001	Roma	AZ1301	700 EUR
04/05/2001	New York	04/06/2001	Tokyo	NW675	900 EUR

While the matching is trivial if the amount of data being analyzed is small, when the size of data and number of sites to visit grow, searching can become difficult, time consuming, and boring. Our agent-based infrastructure aims at mechanising exactly this search and match activity.

3 The Request and MatchMaking Component

The Request and MatchMaking component (RMM henceforth) is in charge of matching the requests of users with offers available in the OPELIX system at some vendor site. The RMM component is actually a distributed system that can be installed on all the sites that store the offers managed by a certain intermediary. In general, a request formulated on a site can require the execution of searching activities on the other sites. The actual execution of the search activity is delegated to a *searching agent*, which is also able to move through different sites when needed, with some degree of autonomy.

Referring to the example of Section 2, an agent, having a list of hotels and theaters, can first look in the data about theaters for the dates of the customer's preferred performances and then look for an accommodation in some nearby hotel. We call *combined searches* those that are executed on different sites on inter-dependent data, and provide an integrated result. A searching agent (either mobile or fixed) can make decisions on the number of results to collect before terminating the visit: it can either decide to search for all data or stop when, for instance, at least one result has been found. Moreover a searching mobile agent can take decisions on the sites to visit and the opportunity to perform parallel searches on more than one site. All these decisions can be taken on the basis of information that are either provided to the agent as part of a request or part of its own semantics. In our prototype, agents are instructed by formulating requests in a simple language we have called *Request Definition Language* (RDL).

4 The Request Definition Language

Request Definition Language (RDL) is the language we have developed to define a search request. A search request is defined in terms of a *main objective* and, optionally, some *subordinate objectives.* Referring to the example of Section 2, the main objective is to find a "La Traviata" performance. Subordinate objectives concern finding flights landing in the city where the theater is located and hotels in the same city having a suite available in the performance period.

Any main or subordinate objective is defined in RDL according to the following syntax:

```
APPLY ‘‘query’’ AS ‘‘queryname’’
AT ‘‘site1’’, ‘‘site2’’, ... ,‘‘siteN’’ USING_POLICY ‘‘policyname’’
```

The `query` defines the data that have to be extracted to fulfill the corresponding objective. Later in this section we present its syntax. `queryname` is a label that identifies the query. `site1`, `site2`, etc. form the set of information providers where the query has to be executed (the *itinerary*). Seller addresses are currently represented in the form `tcp://<IP address>:PORT`. The term `policyname` defines the strategy to enforce termination of the current search by stating the level of completeness required for results. The *best_first* policy determines that the search is stopped when a first set of results is retrieved. The *complete* policy allows to retrieve all available results. Finally, the *medium* policy allows the search to be stopped as soon as the specified number of results is retrieved.

In principle the `query` might be expressed in any language that is understood at the sites where the query is actually executed. We use a XML query language, XQL [6], since we assume that data being queried are expressed in XML. However, RDL has been designed in such a way that it will be easy to replace XQL with any language will result from the standardization work of the W3C query working group.

An example of query in XQL that extracts the author's firstname and lastname and title of books that cost less then 50, with a discount of more than 10 is the following:

```
/Order/Item[discount > 10]/Book[price < 50]?/
(Author/Firstname | Author/Lastname | Title)
```

In RDL, queries referring to main objectives are XQL well-formed formulas. The clause below represents the main objective of the example presented in Section 2:

```
APPLY ``//show[name = ``La Traviata'' $and$ status = ``Available'']?/
(city|date|price)'' AS ``performances''
AT ``tcp://site1.domain1.com:10000'', ``tcp://site3.domain3.com:10000''
USING_POLICY ``complete''
```

Subordinate objectives are preceded by the THEN keyword. Queries referring to them present some additional non-XQL terms that are used to express relationships between main and subordinate objectives. An example of subordinate clause is reported below:

```
THEN APPLY ``//flight[arrival/date < %performances@show.date% $and$
           arrival/place = %performances@show.city% $and$
           departure/place = ``New York'']?/
           (code|price|departure/date)'' AS ``flights''
     AT ``tcp://site2.domain2.com:10000'', ...
     USING_POLICY ``complete''
```

In the constraint part references to results of some other queries are present. Such references are resolved during the execution of a request. Syntactically, we use the notation %queryname@tagname% to refer to the values of tagname in the result set of query queryname.

5 The System Architecture

The RMM component is in charge of managing search requests expressed in the RDL language. The core components of the architecture are searching agents that interpret and execute the requests. In Figure 1 agents are represented as small faces. The picture shows that they can move from site to site to perform a search. In addition, the picture shows that several sources of information (sellers, in the OPELIX context) can be involved in the execution of a search.

An agent is instantiated as a result of a customer's request. Such a request is usually received by an Intermediary who interacts with the customer offering searching and other i-commerce services. The RMM component installed at the Intermediary site is aware of the structure of sellers' databases given in terms of DTD and other meta-information in XML. It provides users with guidance in the definition of a search request and then translates it into a RDL script to be passed to an agent. The agent interprets an RDL statement by executing the following steps: a) identify all objectives to be fulfilled; b) for each objective,

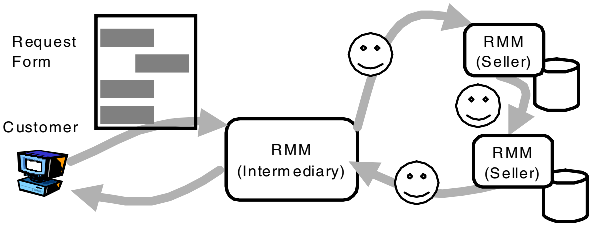


Fig. 1. High level architecture of the RMM component. Agents are visualized by small faces.

query proper sources (in our example, the web sites providing information on theatres, airlines, and hotels); c) assemble the results.

The management of each objective requires the execution of the following operations:

- *Query preparation*: if the query contains references to other objectives then these references are solved and a well-formed XQL query is generated.
- *Query execution*: the query is executed at the sites belonging to the site list.
- *Assembling of results*: results obtained at each site are merged.

Agents live and operate within specific execution contexts that isolate them from the details of the machine where they are running and provide all services needed for moving and communicating with other agents. As a mobile agent infrastructure, we have selected Voyager [4] which was found to be easy to use and powerful enough for our purposes. In principle, however, any infrastructure could replace it.

From the protected context provided by the agent infrastructure, the agent has to execute queries that require access to the information on products stored at each site. Such access is mediated by the XQL interpreter which, for the sake of flexibility, we assume to be installed locally at each site. Such a component could have been moved and installed on each site together with the agent at the cost of increasing the network load.

6 Conclusion

Our search subsystem, being dynamic in its behavior, distinguishes itself from the traditional search engines. It allows users to perform sophisticated searches, which can yield results that are tightly adherent to their expectations. The searching process depends on the single request. Requests with some degree of mutual inter-dependency can be formulated, as we have shown in our running example.

Our approach is based on autonomous agents. Each agent interprets a searching task and executes it producing a result that is either communicated to the customer or packaged with the results obtained along its trip. By exploiting mobility, the agent, which packages both the knowledge needed for searching and the partial results obtained in a searching activity, performs the search at the site where the resources are stored. This approach is not always convenient wrt traditional remote method invocation [2]. Our preliminary experience gained with OPELIX shows that moving agents offers advantages in the case of network faults and when users may become disconnected from the network. In these cases, in fact, the agent can perfom at least part of its task even if it is temporarily isolated from the rest of the system.

One of the problems we faced with in the development of our prototype is the lack of standard approaches for mobile agents development. Each support plaform offers its own linguistic and runtime mechanisms based on a model of mobility that in general vary from case to case. MASIF [5] tries to unify these differencies by standardizing various aspects of agent management.

A notable issue that is currently missing in our approach concerns security. This problem has several facets that need to be considered separately. A crucial issue is to ensure that agents do not try to attack the system hosting them and, in turn, are not attacked by it. We plan to integrate in our prototype results achieved by other researchers. Other future activities concern the refinement of RDL in order to make it more expressive. The RMM prototype has been delivered to the OPELIX partners and is going to be used in real case studies. We plan to collect useful hints, feedbacks, and new requirements from these experiences.

Acknowledgements. We thank all members of the OPELIX team. We also thank Angela Bonifati who helped us with XML query languages.

References

1. A. Bonifati and S. Ceri. Comparative Analysis of Five XML Query Languages. *ACM Sigmod Record*, 29(1):68–77, March 2000.
2. A. Carzaniga, G. P. Picco, and G. Vigna. Designing Distributed Applications with Mobile Code Paradigms. In R. Taylor, editor, *Proceedings of the 19th International Conference on Software Engineering (ICSE'97)*, pages 22–32. ACM Press, 1997.
3. V. Grover and J. T. C. Teng. E-Commerce and the Information Market. *Communication of the ACM*, 44(4), April 2001.
4. O. Inc. ObjectSpace Voyager 4.0 Documentation, 2000. http://support.objectspace.com/doc/index.html.
5. OMG. CORBA Facilities: Mobile Angent System Interoperability Facilities Submission. OMG Technical Report, 1997.
6. J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL), 1998. http://www.w3.org/TandS/QL/QL98/pp/xql.html.

Extending a Multi-agent System for Genomic Annotation*

Keith Decker, Salim Khan, Carl Schmidt, and Dennis Michaud

Computer and Information Sciences Department
University of Delaware, Newark, DE 19716
{decker}@cis.udel.edu

Abstract. The explosive growth in genomic (and soon, expression and proteomic) data, exemplified by the Human Genome Project, is a fertile domain for the application of multi-agent information gathering technologies. Furthermore, hundreds of smaller-profile, yet still economically important organisms are being studied that require the efficient and inexpensive automated analysis tools that multi-agent approaches can provide. In this paper we give a progress report on the use of the DECAF multi-agent toolkit to build reusable information gathering systems for bioinformatics. We will briefly summarize why bioinformatics is a classic application for information gathering, how DECAF supports it, and recent extensions underway to support new analysis paths for genomic information.

1 Introduction

Massive amounts of raw data are currently being generated by biologists while sequencing organisms. Most of this raw data must be analyzed through the piecemeal application of various computer programs and hand-searches of various public web databases. Typically both the raw data and any valuable derived knowledge will remain generally unavailable except in published natural language texts such as journal articles. However, it is important to note that a tremendous amount of genetic material is *similar* from organism to organism, even when they are as outwardly different as a yeast, fruit fly, mouse, or human being. This means that if a biologist studying the yeast can figure out what a certain gene does—its *function*—that other biologists can at least guess that similar genes in other organisms play similar roles. Thus huge databases are being populated with sequence data and functional annotations [2]. All new sequences are routinely compared to known sequences for clues as to their functions.

A large amount of work in bioinformatics over the past ten years has gone into developing algorithms (pattern matching, statistical, and/or heuristic/knowledge-based) to support the work of hypothesizing gene function. Many of these are available to biologists in various implementations, and now many are available over the web. Metasites combine many published algorithms, and sites specialize in information about particular topics such as protein motifs.

* This work was supported by the National Science Foundation under grants IIS-9812764, IIS-9733004 and BDI-0092336.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 106–117, 2001.

From a computer science perspective, several problems have arisen, as we have described elsewhere [6]. To summarize, what we have is a large set of heterogeneous and dynamically changing databases, all of which have information to bring to bear on the biological problem of determining genomic function. We have biologists producing thousands of possible genes, for which functions must be hypothesized. For the case of all but the largest and well-funded sequencing projects, this must be done by hand by a single researcher and their students.

Multi-agent information gathering systems have a lot to contribute to these efforts. Several features make a multi-agent approach to this problem particularly attractive: information is available from many distinct locations; information content is heterogeneous; information content is constantly changing; much of the annotation work for each gene can be done independently; biologists wish to both make their findings widely available, yet retain control over the data; new types of analysis and sources of data are appearing constantly.

We have used DECAF, a multi-agent system toolkit based on RETSINA [21,10,7]: and TAEMS [9,23], to construct a prototype multi-agent system for automated annotation and database storage of sequencing data for herpesviruses [6]. The resulting system eliminates tedious and always out-of-date hand analyses, makes the data and annotations available for other researchers (or agent systems), and provides a level of query processing beyond even some high-profile web sites.

Since that initial system, we have used the distributed, open nature of our multi-agent solution to expand the system in several ways that will make it useful for biologists studying more organisms, and in different ways. This paper will briefly describe our approach to information gathering, based on our work on RETSINA; the DECAF toolkit; our initial annotation system; and our new extensions for functional annotation, EST processing, and metabolic pathway reasoning.

2 DECAF

DECAF (Distributed, Environment-Centered Agent Framework) is a Java-based toolkit for creating multi-agent systems [13]. In particular, several tools have been developed specifically for prototyping information gathering systems. Also, the internal architecture of each DECAF agent has been designed much like an operating system—as a set of services for the "intelligent" (resource-efficient, adaptively-scheduled, soft real-time, objective-persistent) execution of agent actions. DECAF consists of a set of well defined control modules (initialization, dispatching, planning, scheduling, and execution, each in a separate, concurrent thread) that work in concert to control an agent's life cycle. There is one core task structure representation that is shared between all of the control modules. This has meant that even non-reusable domain-dependent agents can be developed more quickly than by the API approach where the programmer has to, in effect, create and orchestrate the agent's architecture as well as its domain-oriented agent actions. This section will first discuss the internal architecture of a generic DECAF agent, and then discuss the tools (such as middle agents, system debugging aids, and the information extraction agent shell) we have built to implement multi-agent information gathering systems. The overall internal architecture of DECAF is shown in Figure 1. These modules run **concurrently**, each in their own thread. Details of the DECAF implementation can be found elsewhere [13].

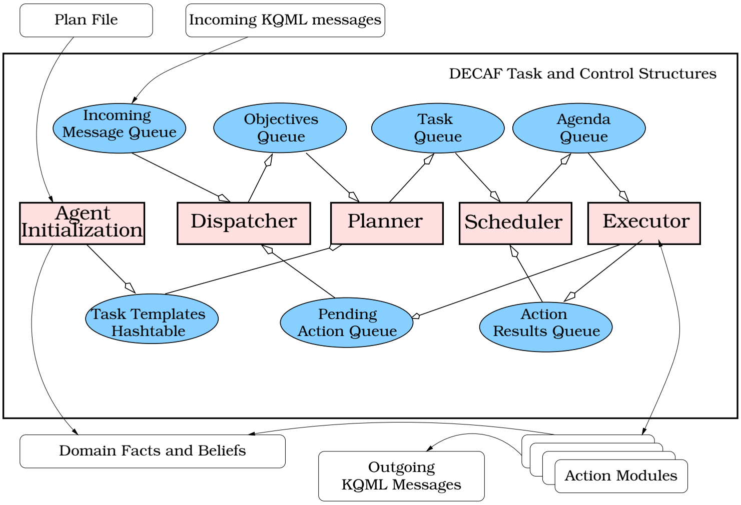


Fig. 1. DECAF Architecture Overview

2.1 DECAF Support for Info Gathering

DECAF provides core internal architectural support for secondary user utility. Thus DECAF plans can include alternatives, and these alternatives can be chosen dynamically at runtime depending on user constraints on answer timeliness or other resource constraints. DECAF also supports building information gathering systems by providing useful middle agents and a shell for quickly building information extraction agents for wrapping web sites. The **Agent Name Server (ANS)** ("white pages") is an essential component for agent communication. It works in a fashion similar to DNS (Domain Name Service) by resolving agent names to host and port addresses. The **Matchmaker** serves as a "yellow pages" to assist agents in finding services needed for task completion. The **Broker** agent acts as a kind of "middle manager" to assist an agent with collections of services. The broker can now provide a larger service than any single provider can, and often manage a large group of agents more effectively [8]. A **Proxy** agent allows web page Java applets to communicate with DECAF agents that are not located on the same server as the applet. The **Agent Management Agent (AMA)** allows MAS designers a look at the entire running set of agents spread out across the Internet that share a single agent name server. This allows designers to query the status of individual agents and watch or record message passing traffic.

Information Extraction Agent Shell. The main functions of an information extraction agent (IEA) are [7]: Fulfilling requests from external sources in response to a *one shot* *query* (e.g. “What is the price of IBM?”). Monitoring external sources for *periodic* information (e.g. “Give me the price of IBM every 30 minutes.”). Monitoring sources for patterns, called *information monitoring* requests (e.g. “Notify me if the price of IBM goes below $50.”).” These functions can be written in a general way so that the code can be shared for agents in any domain.

Since our IEA operates on the Web, the information gathered is from external information sources. The agent uses a set of *wrappers* and the wrapper induction algorithm STALKER [18], to extract relevant information from the web pages after being shown several marked-up examples. When the information is gathered it is stored in the local IEA “infobase” using Java wrappers on a PARKA [15] knowledgebase. This makes new IEA’s fairly easy to create, and forces the difficult parts of this problem back on to KB ontology creation, rather than the production of tools to wrap web pages and dynamically answer queries. Currently, there are some proposals for XML-based page annotations which, if adopted, will make site wrapping easier syntactically (but still, does not solve the ontology problem—but see projects such as OIL).

3 A DECAF Multi-agent System for Genomic Analysis

These tools can be put to use to create a prototype multi-agent system for various types of genomic analysis. In the prototype, we have chosen to simplify the query subsystem by materializing all annotations locally, thus removing the need for sophisticated query planning (e.g. [16]). This is a reasonable simplification since most of our work is with viruses that have fairly small genomes (around 100 genes for a herpesvirus and around 30 herpesviruses) or with larger organisms (e.g. chickens) for which we are constructing a consensus database explicitly.

Figure 2 shows an overview of the system as four overlapping multi-agent organizations. The first, *Basic Sequence Annotation*, is charged with integrating remote gene sequence annotations from various sources with the gene sequences at the Local KnowledgeBase Management Agent (LKBMA). The second, *Query*, allows complex queries on the LKBMAs via a web interface. The third, *Functional Annotation* is responsible for collecting information needed to make an informed guess as to the function of a gene, specifically using the three-part Gene Ontology [22]. The fourth organization, *EST Processing* enables the analysis of expressed sequence tags (ESTs) to produce gene sequences that can be annotated by the other organizations.

An important feature to note is that we are focusing on annotation and analysis services that are not organism specific. In this way, the resulting system can be used to build and query knowledgebases from several different organisms. The original subsystems (basic annotation and the simple query system) were built to annotate the newly sequenced Herpesvirus of Turkey (the bird), and then to compare it to the other known sequenced herpesviruses. Work is just beginning to build a new knowledgebase from chicken ESTs, and to extend the depth of the herpesvirus KB for Epstein-Barr Virus (human herpesvirus 4) which has clinical significance for pediatric organ transplant patients.

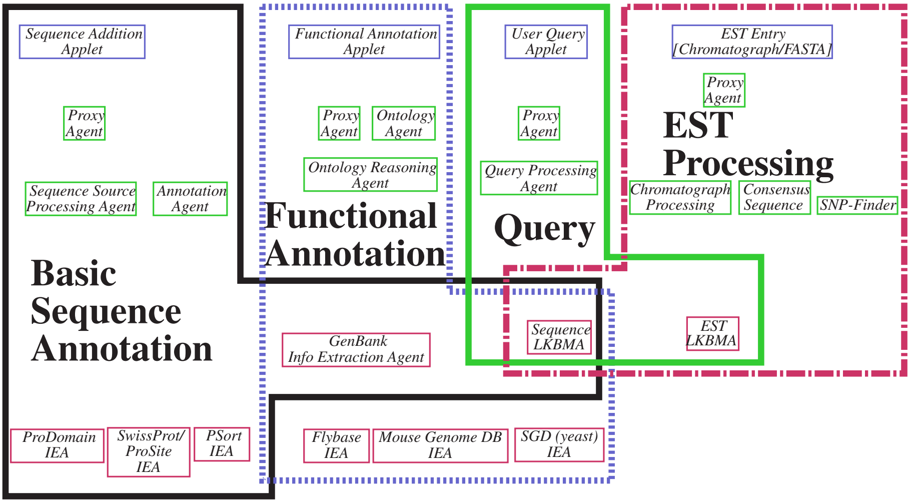


Fig. 2. Overview of DECAF Multi-Agent System for Genomic Analysis

3.1 Basic Sequence Annotation and Query Processing

Figure 3 shows the interaction details for the basic sequence annotation and query subsystems. We will describe the agents by their RETSINA classification.

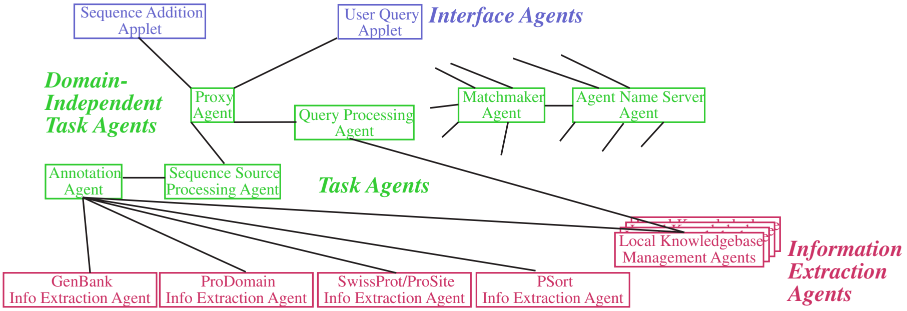

Sequence Addition Applet
User Query Applet
Interface Agents
Domain-Independent Task Agents
Proxy Agent
Query Processing Agent
Matchmaker Agent
Agent Name Server Agent
Annotation Agent
Sequence Source Processing Agent
Task Agents
Local Knowledgebase Management Agents
Information Extraction Agents
GenBank Info Extraction Agent
ProDomain Info Extraction Agent
SwissProt/ProSite Info Extraction Agent
PSort Info Extraction Agent

Fig. 3. Basic Annotation and Query Agent Organizations

Information Extraction Agents. Currently 4 agents based on the IEA shell wrap public web sites. The Genbank wrapper primarily supplies "BLAST" services: given the sequence of a herpesvirus gene, what are the most similar genes known in the world (called "homologs")? The answer here can give the biologist a clue as to the possible function of a gene, and for any gene that the biologist does not know the function of, a change in the answer to this query might be significant. The SwissProt wrapper primary provides protein motif pattern searches. If we view a protein as a one-dimensional string of amino acids, then a motif is a regular expression matching part of the string that may indicate a particular kind of function for the protein (i.e. a prenylation motif indicates a place where the protein may be modified after translation by the addition of another group of molecules) The PSort wrapper accesses a knowledge-based system for estimating the likely sub-cellular location that a sequence's encoded protein will be used. The ProDomain wrapper allows access to other information about the encoded protein; a protein domain is similar to a motif but larger. As we move to new organisms, many more resources could be wrapped at this level (almost all biologists have a "favorite" here).

The local knowledgebase management agent (KBMA) is a slightly different member of this class because unlike most IEAs it actually stores data via agent messages rather than only querying external data sources. It is here that the annotations of the genetic information are materialized, and from which most queries are answered. Each KBMA is updated with raw sequencing data indirectly from a user sequence addition interface that is then automatically annotated under the control of an annotation task agent. KBMAs can be "owned" by different parties, and queried separately or together. In this way, researchers with limited computer knowledge can create sharable annotated sequence databases using the existing wrappers and other analysis tools as they are developed, without having to necessarily download and install them themselves. Using a PARKA-DB knowledgebase allows efficient, modern relational data storage on the back end and query as well as limited KB inferencing [15].

Task Agents. There are two domain task agents; the rest are generic middle agents described earlier. The Annotation Agent directs exactly what information should be annotated for each sequence. It is responsible for storing the raw sequence data, making queries to the various wrapped web sites, storing those annotations, and also indicating the provenance of the data (meta-information regarding where an annotation came from). The Sequence Source Processing Agent takes almost raw sequence data in ASN.1 format as output by typical sequence estimation programs or stored in Genbank. The main function of this agent is to test this input for internal consistency.

Interface Agents. There are two interface applets that communicate via the proxy agent with other agents in the system. One is oriented towards adding new sequences to a local knowledgebase (secured by a password) and the other allows anyone to query the complete annotated KB (or even multiple KBs). The interface hardly scratches the surface of the queries that are actually possible, but a big problem is that most biologists are not comfortable with complex query languages. Indeed, the simple interface that allows simple conjunctive and disjunctive queries over dynamic menus of annotations (constructed by the applet at runtime from the actual local KB) is quite advanced as compared to most of the existing public sites that allow textual keyword searches only.

3.2 Functional Annotation

This subsystem is responsible for assisting the biologist in the difficult problem of making functional annotations of each gene. Unfortunately, many of the millions of genes sequenced so far have fairly haphazard (from a computer scientist's perspective) functional annotation: simply free natural language descriptions. Recently, a fairly large group representing at least some of the primary organism databases have created a consortium dedicated to creating a gene ontology for annotating gene function in three basic areas: the biological process in which a gene plays a part, the molecular function of the gene product, and the cellular localization [22]. The subsystem described here supports the use of this ontology by biologists as sequences are added to the system, eventually leading to even more powerful analysis of the resulting KBs.

Information Extraction Agents. Besides the gene sequence LKBMA and the GenBank IEA, we are wrapping three new organism-specific gene sequence DBs, for Drosophila (fruit fly), Mus (Mouse), and Saccrynomaeces cervasie (yeast). Each of these organisms is part of the Gene Ontology (GO) consortium, and has spent considerable time in making the proper functional annotation. Each of these agents, then, finds GO-annotated, close homologs of the unannotated gene and proposes the annotation of the homologs for the annotation of the new gene.

Task Agents. There are two new task agents, one is a domain-independent ontology agent using the FIPA ontology agent specification as a starting point. The ontology agent contains both the GO ontologies and several mappings from other symbologies (i.e. SwissProt terms) to GO terms. In fact, the Mouse IEA uses the Ontology agent to map some non-GO terms for certain records to GO terms. Although not indicated on the figure, some of the other organism DB IEA agents must map from GO ontology descriptive strings to the actual unique GO ID. The other service provided by the ontology agent (and not explicitly mentioned in the experimental FIPA Ontology Agent specification) is for the ontology reasoning agent to ask how to terms are related in an ontology. The Ontology Reasoning Agent uses this query to build a minimum spanning tree (in each of the three GO ontologies) between all the terms returned in all the homologies from all of the GO organism databases. This information can then be used to propose a likely annotation, and to display all of the information graphically for the biologist via the interface agent.

Interface Agents. The functional interface agent/applet consists of two columnar panes: on the left, the top pane displays the gene being annotated, and the bottom displays the general homologies from GenBank with their natural language annotations. On the right, three panes display the subtrees from the three GO ontologies (biological process, molecular function, cellular location) marked in color with the homologs from the three organism databases.

3.3 EST Processing

One way to broaden the applicability of the system is to accept more kinds of basic input data to the annotation process. For example, we could broaden the reach of the system by starting with ESTs (Expressed Sequence Tags) instead of complete sequences. Agents could wrap the standard software for creating sequences from this data, at which point the existing system can be used. The use of ESTs is part of a relatively inexpensive approach to sequencing where instead of directly sequencing genomic DNA, we instead use a method that produces many short sequences that partially overlap. By finding the overlaps in the short sequences, we can eventually reconstruct the entire sequence of each expressed gene. Essentially, this is a “shotgun” approach that relies on statistics and the sheer number of experiments to eventually produce complete sequences.

As a side effect of this processing, information is produced that can be used to find Single Nucleotide Polymorphisms (SNPs). SNPs indicate a change of one nucleotide (A,T,C,G) in a single gene between different individuals (often, conserved across strains or subspecies). These markers are very important for identification even if they do not have functional effects.

Information Extraction Agents. The process of consensus sequence building and SNP identification does not require any external information, so the only IEAs are the LKBMAs. Up until now, there has only been one LKBMA, responsible for the gene sequences and annotations. EST processing adds a second LKBMA responsible for storing the ESTS themselves and the associated information discussed below. Primarily, this is because (especially early on in a sequencing project) there will be thousands of ESTs that do not overlap to form contiguous sequences, and that ESTs may be added and processed almost daily.

Task Agents. There are three new domain-level task agents. The first deals with processing chromatographs. Essentially the chromatograph is a set of signals that indicate the relative strengths of the wavelengths associated with each luminous nucleotide tag. Several standard Unix analysis programs exist to process this data, essentially "calling" the best nucleotide for each position. The chromatograph processing agent wraps three analysis programs: Phred, which "calls" the chromatograph and also separately produces an uncertainty score for each nucleotide in the sequence; phd2fasta which converts this output into a standard (FASTA) format; and x-match which removes a part of the sequence that is a byproduct of the sequencing method, and not actually part of the organism sequence. The consensus sequence assembly agent uses two more programs (Phrap and consed) on all the ESTs found so far to produce a set of candidate genes by appropriately splicing together the short EST sequences. This produces a set of candidate genes that can then be added to the gene sequence LKBMA and from which the various annotation processes described earlier may commence. Finally, a SNP-finder agent operates the PolyBayes program which uses the EST and Sequence KBs and the uncertainty scores produced by Phred to nominate possible single nucleotide polymorphisms.

Interface Agents. There is only one simple interface agent, to allow participants to enter data in the system. Preferably, this is chromatograph data from the sequencers, because the original chromatograph allows Phred to calculate the uncertainty associated with each nucleotide call. However, FASTA-format (simple "ATCG..." named strings) ESTs called from the original chromatographs can be accommodated. These can be used to build consensus sequences, but not for finding SNPs.

4 Gene Expression Processing

A new kind of genomic data is now being produced, that may swamp even the amount of sequencing data. This is so-called *gene expression* data, and indicates quantitatively how much a gene product is expressed in some location, under some conditions, at some point in time. We are developing an multi-agent system that uses available online genomic and metabolic pathway knowledge to extend gene expression analysis. By incorporating known relationships between genes, knowledge-based analysis of experimental expression data is significantly improved over purely statistical methods.

Although this system has not yet been integrated into the existing agent community, eventually relevant genomic information will be made available to the system through the existing GenBank and SwissProt IEAs. Metabolic pathways of interest to the investigator are identified through a KEGG (Kyoto Encyclopedia of Genes and Genomes) database wrapper. Analysis of the gene expression data is performed through an agent that executes SAS, a statistical package that includes clustering and PCA analysis methods. Results are to be presented to the user through web pages hyperlinked to relevant database entries.

Current techniques for gene expression analysis have primarily focused on the use of clustering algorithms, which group genes of similar expression patterns together [12]. However, experimental gene expression data can be very noisy and the complicated pathways within organisms can generate coincidental expression patterns, which can significantly limit the benefits of standard cluster analysis. In order to separate gene co-regulation patterns from co-expression, the gene expression processing organization was developed to gather available pathway-level information in order to presort the expression data into functional categories. Thus, clustering of the reduced data set is much more likely to find genes that are actually regulated together. The system also promises to be useful in discovering regulatory connections between different pathways. One advantage of using the KEGG database is that its gene/enzyme entries are organized by the EC (Enzyme Commission) ontology, and so are easily mapped to gene names specific to the organism of interest.

5 Related Work

There has been significant work on general algorithms for query planning, selective materialization, and the optimization of these from the AI perspective, for example TSIMMIS [4], Infosleuth [19], SIMS [1], etc., and of course on applying agents as the way to embody these algorithms [16,21,10].

In Biology, compared to the work being done to create the raw data, all the work on how to organize and retrieve it is relatively small. Most of the work in computer science directed to biological data has been in the area of heterogeneous databases, focusing on the semi-structured nature of much of the data that makes it very difficult to store usefully in commercial relational databases [5]. Some work has begun in applying the work on wrappers and mediators to biological databases, for example TAMBIS [20]. These systems differ from ours in that they are pure implementations of wrapper/mediator technology that are centralized, do not allow for dynamic changes in sources, support persistent queries, or consider secondary user utility in the form of time or other resource limitations.

Agent technology has been making some inroads in the area. The word “agent” with the popular connotation of a single computer program to do a user’s bidding is found in the promotional material for Doubletwist (`www.doubletwist.com`). Here, an “agent” stands for a persistent query (e.g. “tell me if a new homolog is found in your database for the following sequence”). There is no collaboration or communication between agents.

We know of a few truly multi-agent projects in this domain. First, InfoSleuth has been used to annotate livestock genetic samples [11]. The flow of information is very similar to our system. However, the system is not set up for noticing changes in the public databases, for integrating new data sources on the fly, or for consideration of secondary user utility. Second, the EDITtoTrEMBL system [17] is another automated annotation system, based on the wrapper and mediator concept, for annotating proteins awaiting manual annotation and entry to SwissProt. Dispatcher agents control the application of potentially complex sequences of wrappers. Most importantly, this system supports the detection and possible revision of inconsistencies revealed between different annotations. Third, the GeneWeaver project [3] is another true multi-agent system for annotation of genomes. GeneWeaver has as a primary design criterion the observation that the source data is always changing, and so annotations need to be constantly updated. They also express the idea that new sources or analysis tools should be easy to integrate into the system, which plays to the open systems requirement, although they do not describe details. The primary differences are the way in which an open system is achieved (it is not clear that they use agent-level matchmaking, but rather possibly CORBA specifications) and that GeneWeaver is not based on a shared architecture that supports reasoning about secondary user utility. In comparison to the DECAF implementation, GeneWeaver uses CORBA/RMI rather than TCP/IP communication, and a simplified KQML-like language called BAL.

6 Discussion

The system described here is operational and normally available on the web at http://udgenome.ags.udel.edu/herpes/. This is a working prototype, and so the interface is strongly oriented to biologists only. In general, computational support for the *processes* that biologists use in analyzing data is primitive (Perl scripts) or non-existent. In less than 10 min, we were able to annotate the HVT-1 sequence, as well as store it in a queryable and web-publishable form. This impressed the biologists we work with, compared to manual annotation and flat ASCII files. Furthermore, we have recently added approximately 15 other publicly available herpesvirus sequences (e.g. several strains of Human herpesvirus, African swine fever virus, etc.). The resulting knowledgebase almost immediately resulted in queries by our local biologists that indicated possible interesting relationships that may result in future biological work. This summer we will begin testing with viral biologists from other universities.

Other things about the system which have excited our biologist co-workers are the relative ease by which we can add new types of annotation or analysis information, and the fact that the system can be used to build similar systems for other organisms, such as the chicken. For example, the use of open system concepts such as a matchmaker allow the annotation agent to access and use new annotation services that were not available when it was initially written. Secondary user utility will become useful for the biologist when faced with making a simple office query vs. checking results before publication.

The underlying DECAF system has been evaluated in several ways, especially with respect to the use of parallel computational resources by a single agent (all of the DECAF components and all of the executable actions are run in parallel threads), and the efficacy of the DRU scheduler which efficiently solves a restricted subset of the design-to-criteria scheduling problem [14]. Running the gene annotation system as a truly multi-agent system results in true speedups, although most of the time is currently spent in remote database access Parallel hardware for each agent will be useful for some of the more locally computationally intensive tasks involving EST processing.

7 Conclusions and Future Work

In this paper we have discussed the very real problem of making some use of the tremendous amounts of genetic sequence information that are being produced. While there is much information publicly available over the web, accessing such information is different for each source and the results can only be used by a single researcher. Furthermore, the contents of these primary sources are changing all the time, and new sources and techniques for analysis are constantly being developed.

We cast this sequence annotation problem as a general information gathering problem, and proposed the use of multi-agent systems for implementation. Beyond the basic heterogeneous database problem that this problem represents, an MAS solution gives us mechanisms for dealing with changing data, the dynamic appearance of new sources, minding secondary utility characteristics for users, and of course the obvious distributed processing achievements of parallel development, concurrent processing, and the possibility for handling certain security or other organizational concerns (where part of the agent organization can mirror the human organization).

We currently are offering the system publicly on the web, with the known herpesvirus sequences. A second system based on chicken ESTs should be available by the end of 2001. We intend to broaden the annotation coverage and add more complex analyses. An example would be the estimation of the physical location of the gene as well as its function. Because biologists have long recorded certain QTLs (Quantitative Trait Loci) that indicate that a certain *physical region* is responsible for a trait (such as chickens with resistance to a certain disease), being able to see what genes are physically located in the QTL region is a strong indicator as to their high-level genetic function.

In general, we have not yet designed an interface that allows biologists to take full advantage of the materialized data —they are uncomfortable with complex query languages. We believe that it may be possible to build a graphical interface to allow a biologist, after some training, to create a commonly needed analysis query and to then save this for use in the future by that scientist, or others sharing the agent namespace.

Finally, the next major subsystem will be agents to link and analyze gene expression data (which will in turn interoperate with the metabolic pathway analysis systems described above). This data needs to be linked with sequence and function data, to allow more powerful analysis. For example, linked to QTL data, this allows us to ask questions such as “what chemicals might prevent club root disease in cabbage?”.

References

1. Y. Arens and C.A. Knoblock. Intelligent caching: Selecting, representing, and reusing data in an information server. In *Proc. 3rd Intl. Conf. on Info. and Know. Mgmt.*, 1994.
2. D.A. Benson and et al. Genbank. *Nucleic Acids Res.*, 28:15–18, 2000.

3. K. Bryson, M. Luck, M. Joy, and D.T. Jones. Applying agents to bioinformatics in geneweaver. In *Proc. 4th Int. Wksp. Collab. Info. Agents*, 2000.
4. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: integration of heterogeneous information sources. In *Proc. 10th Mtg. Info. Proc. Soc. Japan*, Dec. 1994.
5. S. B. Davidson and et al. Biokleisli:a digital library for biomedical researchers. *Intnl. J. on Digital Libraries*, 1(1):36–53, 1997.
6. K. Decker, X. Zheng, and C. Schmidt. A multi-agent system for automated genomic annotation. In *Proceedings of the 5th Intl. Conf. on Autonomous Agents*, Montreal, 2001.
7. K. S. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proc. 1st Intl. Conf. on Autonomous Agents*, pages 404–413, 1997.
8. K. S. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proc. 15th IJCAI*, pages 578–583, 1997.
9. K. S. Decker and V. R. Lesser. Quantitative modeling of complex computational task environments. In *Proc. 11th AAAI*, pages 217–224, 1993.
10. K. S. Decker and K. Sycara. Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9(3):239–260, 1997.
11. L. Deschaine, R. Brice, and M. Nodine. Use of infosleuth to coordinate information acquisition, tracking, and analysis in complex applications. MCC-INSL–008-00, 2000.
12. M.B. Eisen, P.T. Spellman, P.O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Nat. Acad. Sci.*
13. J. Graham and K.S. Decker. Towards a distributed, environment-centered agent framework. In *Intelligent Agents VI*, LNAI-1757, pages 290–304. Springer Verlag, 2000.
14. J. Graham. *Real-time Scheduling in Multi-agent Systems*. PhD thesis, University of Delaware, 2001.
15. J. Hendler and M. Taylor K. Stoffel. Advances in high performance knowledge representation. Technical Report CS-TR-3672, University of Maryland Institute for Advanced Computer Studies, 1996. Also cross-referenced as UMIACS-TR-96-56.
16. C.A. Knoblock, Y. Arens, and C. Hsu. Cooperating agents for information retrieval. In *Proc. 2nd Intl. Conf. on Cooperative Information Systems*. Univ. of Toronto Press, 1994.
17. S. Möller and M. Schroeder. Consistent integration of non-reliable heterogeneous information applied to the annotation of transmembrane proteins. *Journal of Computing and Chemistry*, to appear, 2001.
18. I. Muslea, S. Minton, and C. Knoblock. Stalker: Learning expectation rules for simistructured web-based information sources. In *Papers from the 1998 Workshop on AI and Information Gathering*, 1998. also Technical Report ws-98-14, University of Southern California.
19. M. Nodine and A. Unruh. Facilitating open communication in agent systems: the infosleuth infrastructure. In *Intelligent Agents IV*, pages 281–295. Springer-Verlag, 1998.
20. R. Stevens and et al. Tambis: Transparent access to multiple bioinformatics information sources. *Bioinformatics*, 16(2):184–185, 2000.
21. K. Sycara, K. S. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–46, December 1996.
22. The Gene Ontology Consortium. Gene ontolgy: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.
23. T. Wagner, A. Garvey, and V. Lesser. Complex goal criteria and its application in design-to-criteria scheduling. In *Proc. 14th AAAI*, 1997.

Domain-Independent Ontologies for Cooperative Information Agents

Mario Gomez, Chema Abasolo, and Enric Plaza

IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain.
{mario,abasolo,enric}@iiia.csic.es
http://www.iiia.csic.es

Abstract. Cooperative Information Agents and modern information systems in general have to access large amount of information distributed across multiple heterogeneous sources. A great challenge of such systems is to evolve by adding new information sources or adapting the existing components for different domain knowledge. We propose the UPML as a framework to build Information Agents by reusing a library of problem solving components that are defined in a domain-independent manner. Moreover, the UPML language is used as an Agent Capability Description Language (ACDL) suitable to configure and build an application. From this approach, a new application can be build by linking the components of the library with a particular domain and a collection of heterogeneous information sources. Adaptability and dynamic configuration of such a system is achieved by reasoning about the UPML specifications of agent capabilities. Independence of the domain and semantic interoperability are achieved by using ontologies and bridges (mappings between ontologies), while independence from the information sources is based on the use of ontologies to overcome semantic heterogeneity and wrappers to achieve syntactic interoperability.

1 Introduction

Modern information systems shall manage or have access to large amounts of information and computing services. The different system components can conduct computation concurrently, communicating and cooperating to achieve a common goal. These systems have been called *cooperative information systems*[2]. One major goal of this field is to develop and build information systems from reusable software components. This goal can be achieved by assembling information services on demand from a montage of networked legacy applications and information sources [3]. A promising approach to this problem is provided by Cooperative Information Agents, computational software entities that accesses one or multiple, heterogeneous and distributed information sources [1]. A critical requirement for these systems is the independence of the reasoning components from the domain knowledge. We propose the UPML framework as

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 118–129, 2001.

a methodology to build Information Agents by reusing a library of problem solving components that are defined in a domain-independent manner. Moreover, the UPML language is used as an Agent Capability Description Language. From this approach, a new application can be built by linking the components of the library with a particular domain and a collection of heterogeneous information sources. Independence of the domain and semantic interoperability are achieved using ontologies and bridges (mappings between ontologies), while independence from the information sources is based again in the use of ontologies to overcome semantic heterogeneity and wrappers to achieve syntactic interoperability.

We have built an application that shows how to build Cooperative Information Agents by using the UPML framework: the Web Information Mediator (WIM). The overall goal of WIM is to provide a mediation service for information tasks of a professional user. A mediator is an agent that offers an added value to the information sources it accesses[4]. Typical services offered by a mediator include selection of information sources, information retrieval, and fusion of information from different sources. We have built a library of components to solve this kind of tasks belonging to the field of Intelligent Information Integration (I3) [5]. WIM is a multiagent information system dealing with the problem of looking for medical literature, thus it has been built by connecting the components in the I3 Library with a medicine domain and some web-based information sources that serve bibliographic references in medicine. We want to emphasize the independence between the library (I3), the domain knowledge (medicine) and the external information sources.

The overall description of UPML is presented in §2, the I3 Library is described in §3. The WIM application is described at the conceptual level in §4. The "reuse" question is addressed in §5. The WIM multiagent architecture is described in §6, including a brief discussion on the use of UPML as an ACDL in §6.1. Finally, some conclusions are summarized in S7.

2 An Overview of UPML.

The goal of *software architectures* is learning from system developing experience in order to provide the abstract recurring patterns for improving further system development. As such, software architectures contribution is mainly methodological in providing a way to specify systems. A software architecture has the following elements: (i) components, (ii) connectors, and (iii) a configuration of how the components should be connected [6]. UPML [7] is a software architecture for knowledge systems where its components are tasks, problem-solving methods and domain models. The connectors in UPML are called bridges and the configuration is shown in Fig.1.

First we will briefly explain the basic elements of the UPML architecture, and second how this UPML architecture addresses the problem of the domain-independence.

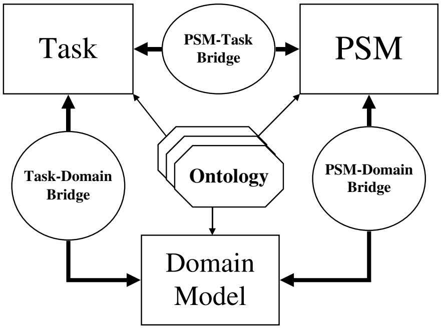


Fig. 1. The UPML software architecture. Note that the relation between an ontology and a component is that the component use this ontology.

2.1 UPML Components

There are three classes of components in the UPML framework: tasks, problem-solving methods and domain models.

Tasks define the goals we have in solving problems, that is, a task specifies what we want to do. A task is characterized by preconditions and goals. A Task specifies constrains and properties to be satisfied for a Problem Solving Method (PSM) that can solve a particular task.

Problem-Solving Methods (PSM) describe which reasoning steps and which types of knowledge are needed to perform a task. A PSM specifies a particular way to solve a task. The main attributes of a PSM are the input/output roles, plus the preconditions and postconditions to be fulfilled by the input and output roles. There are two subclasses of PSM: *Problem Decomposers* and *Reasoning Resources*. Problem Decomposers specify decomposition of tasks into subtasks. Reasoning Resources specifies how to solve a task, it does not describe its internal structure, which is regarded as an implementation aspect.

Domain models model the domain knowledge that can be used by tasks and PSMs. A domain model is characterized by domain knowledge and its properties.

The UPML framework has the notion of *Ontology*. An ontology defines a terminology and its properties. All the components described above use an ontology in their descriptions, this ontology may be different in each component or be shared with other components. The fact of describing tasks, PSMs and domain models with different ontologies makes task and PSM independent of the domain. This independence enable the reuse of task descriptions in different domains, the reuse of PSMs across different tasks and domain, and the reuse of domain knowledge for different tasks or PSMs.

UPML, as a software architecture, has a specific kind of connectors called *Bridges*. A bridge models the relationship between two different components. The function of the bridges is to connect components with different ontologies,translating the concepts of the components ontologies. A bridge provides mapping axioms and assumptions about the components that it relates. There are three kinds of bridges: Task-PSM, Task-Domain Model and PSM-Domain Model (see Fig.1).

2.2 UPML and Domain-Independence

Once we have described the UPML components, let's see how the UPML framework addresses the issue of domain-independence [8]. First of all we have to differentiate two concepts: *Library* and *Application*.

A *Library* is a collection of UPML descriptions of tasks and PSMs. A Library is totally independent of the domain because tasks and PSMs are described in terms of their own ontologies, and not in terms of the domain ontology.

An *Application* is made of one or more libraries, a set of domain models, and the bridges linking those library components with the domain models employed. The mapping axioms of the bridge allow to translate the concepts of the domain model ontology into concepts of the PSM ontology. This translation enable the PSM to work with the domain knowledge of the domain model.

This approach makes the library independent of the domain. This independence allows the library to be reusable, in the sense that the same library can be used for different applications, with different domain models or even with different domains.

In our architecture, the agents define their capabilities in terms of the UPML descriptions of tasks and PSMs. The agents register their capabilities in a Library agent. We will see this link between agents and UPML in §6.

3 The I3 Library

The I3 library offers a collection of methods to solve some of the usual tasks carried on by Information Agents (see for instance [14] [15] [16] [17][18], which are typically known as Intelligent Information Integration[5]), a concept that originally means the integration of multiple databases and nowadays is being focused to the integration of multiple web sources with the use of ontologies [9].

Instead of adopting the information retrieval approach (IR), we adopt a vision more close to meta-search. IR focuses on improving retrieval methods, while meta-search focuses on exploiting existing "information sources", where each resource posses a specific content accessible by a "local" retrieval engine. For this reason, the library and the WIM application focus on this process — and do not include components that can be found inside retrieval engines.

A second consideration is the paradigmatic distinction between the concept of the "relevance" in classical IR and the more rich conceptualizations currently in use for intelligent information agents [13]. The canonical concept of relevance in IR is a test where the results for a query by a retrieval engine are compared to a gold standard provided by a human expert that assesses false positives and false negatives. The problem of that approach is that "relevance" is independent of the purpose of the specific user in posing a query. The I3 Library includes some methods to elaborate and rank results according to a specific utility measure.

I3 can be seen as an adaptation process with four subtasks: transformation of the user consultation , selection of information sources, information retrieval and integration of information from multiple sources.

Adaption refers to the process of elaborating the user consultation to better fulfill his interest, as well as adapting the queries for the idiosyncratic syntax of each information source. Once the retrieval is performed, the results from different queries should be aggregated (§3.2) for different sources, and finally the results for each source are also aggregated to obtain a unique result for the user consultation.

We have adopted a general approach for the overall process of information integration that is based on using query weighting and numerical aggregation operators. Query weighting refers to the process of assigning weights to queries generated according to some domain knowledge, while numerical aggregation operators are the mechanism used to merge items coming from different queries (and sources), and combining the different weights to obtain an unique weight for each item summarizing the relative importance of that item for the user interest. This mechanism allows to score documents retrieved from engines that originally do not give any score, and defining user-oriented utility measures simply by defining the appropriate knowledge categories (see §4.1).

3.1 Adaptation of Queries

We adopt a very well known approach to queries as vectors of keywords instead of complex database query languages. This decision is justified because nowadays professional databases could often be accessed through the use of a web-based search-engine, where queries are made of keywords belonging to a particular domain. We also include search filters as optional constrains allowing to restrict a search procedure. A bibliographic ontology have been used to model the kind of filters allowed by professional bibliography search-engines like *publication date*, *language* and so on. Let's see the both types of query adaptation, adaptation with respect to the domain, and customization for particular information sources.

Query elaboration: refers to the adaptation of queries for a particular user interest, within a particular domain. This task can be achieved by using domain knowledge, like synonyms or predefined knowledge categories.

Query customization: a query is customized to a particular information source by translating keywords and filters into the *search modes* and *filters* used by each selected source. This task is different from the task achieved by wrappers, where keyword-based queries are transformed in the particular syntax of the source, following the rules and particular features of each source at the lowest level.

Selection of sources: it isn't a query elaboration method, but is needed when more than one source is available, so it is very related with the query adaptation process, and in particular, with the query customization task. The selection of sources could be done by asking the user or using a particular method, like Case-Based Reasoning (CBR) . In the current version of WIM three information sources are available (see section §4.1).

In our approach to query adaptation, not only new queries are enriched with domain knowledge, but these queries are weighted according to the strength of the relation between the original and the elaborated query. For example, in a query elaboration with synonyms, new queries are weighted according to the correlation coefficients between the original keywords and the new ones.

3.2 Aggregation of Results

The aggregation of the results is needed in two situations: a) query aggregation and b) fusion. Query aggregation refers to the aggregation of results coming from different *queries*, these *queries* are transformations of an original *query*. Fusion is the aggregation of the results coming from different sources.

Remember that elaboration and customization of queries produce new queries with a weight assigned by these PSMs (Elaboration and Customization). Given that an item can be a result in more than one query, and that each query have a weight, the aggregation functions take the different weights of the queries, where the item is a result, and unify them in a unique value per item.

The aggregation functions are domain-independent, and are well known and classified [20]. WIM has a library of aggregation functions. This library is formed by the Weighted-Mean, the Ordered Weighting Mean (OWA) and the Weighted OWA (WOWA).

Let's see how the weighted-mean aggregates the results of four queries, coming from a source that does not score results. When a source does not return scores, we assign 1 when an item appears (the maximum) in the result of a query and zero when it does not (the minimum). Having an item that appear only in the first and second query we obtain the following rankings

Query	*Weight*	*Normalized-Weight*	*Score*
query1	1.0	0.36	1
query2	0.8	0.29	1
query3	0.6	0.21	0
query4	0.4	0.14	0

The *Weighted-Mean* operator is $WM(a_1, \ldots, a_n) = \sum_{i=1}^{n} w_i a_i$ and weights are normalized, i.e. $w_i \in [0,1]$ and $\sum_{i=1}^{n} w_i = 1$. The resulting aggregation is $WM(1,1,0,0) = 0.36 + 0.29 = 0.65$

4 The WIM Application

The core of WIM is the I3 Library, but there are other components need to build a complete application (See §2.2). In our approach, an application is built by linking the reasoning components in the library with a specific domain knowledge and a collection of information sources.

4.1 Linking the Library with the Domain Knowledge

Domain knowledge do not belongs to the library; this is one of the most important features of the UPML approach, because the independence from the domain is considered a basic requirement to achieve one of the most challenging goals of the knowledge modelling and the software engineering communities: reuse of existing components[10].

The domain chosen to build the WIM application is medicine, and in particular, *Evidence-Based Medicine* (EBM). EBM proposes a medicine practice which calls for careful clinical judgment in evaluating the "best available evidence"[23]. The main task for the WIM application is looking for medical literature, and the utility criteria used to rank documents are those given by EBM to asses the quality of medical bibliography. Hence, we need also some bibliographic knowledge to describe queries and results for the queries, and some knowledge about the information sources selected for the application: PubMed and HealthStar. Let's see the different domain models and how are they used by PSMs to solve a task. The domain knowledge is organized into four domain models.

- A general *medical thesaurus* is used to select the keywords to pose a query and during the elaboration of the queries. We have chosen the MeSH, a thesaurus that can be accessed through a web-based retrieval engine called MeSH Browser. This domain model is used by the PSM *query-elaboration-with-synonyms*.
- An ontology about *bibliographic data*, used to describe the filters typical in bibliographic search engines. After considering standard bibliographic ontologies like the Stanford Bibliographic Data, we decide to use only a limited set of bibliographic concepts, those ones used to build queries and process the results. This domain model is used to pose the queries and by the query-customization PSM.
- A collection of *source descriptions*, where the search modes and allowed filters of each source are described, including the mapping rules between the common bibliographic ontology and the particular terms used by that source. This is used by the *query-customization* method.
- A collection of predefined categories describing the keywords and filters that are useful to rank documents according to the EBM criteria. Used by the PSM *query-elaboration-with-categories*.

Example 1:The query weighting approach adopted for the query adaptation task has great advantages to rank documents by different criteria, not only classical IR's relevance. To introduce new utility criteria we have built a method to elaborate queries by using predefined knowledge categories. A category is a collection of terms and filters associated to one topic, which are weighted according to the strength of that association. For example *Good-Evidence* is Category that defines some filters to get only papers based on a good evidence quality. Two of the filters are the following:

```
(Publication Type = "Meta-Analysis", Weight = 1)
(Publication Type = "Randomized Controlled Trial", Weight = 0.9)
```

Given the query Q = (Levofloxacin, Pneumonia) and applying the PSM *Query-expansion-with-categories* with this category, we get the following set of queries:

```
Q1 = (Levofloxacin, Pneumonia, Publication Type = Meta-Analysis),
     Weight = 1.
Q2 = (Levofloxacin, Pneumonia, Publication Type = Randomized Controlled Trial),
     Weight = 0.9
```

4.2 Linking the Library with the Information Sources

Information sources are not domain models, they are external PSMs that solve the *retrieval* task. But there is a domain model consisting of source descriptions, as explained in previous section (§4.1).

Example 2: The *query-customization* PSM expands a query expressed in a source independent way in a collection of queries in terms of a particular information source, by using the search modes and filters allowed by that source. This knowledge is described in the sources domain model, for example, this is our description of the HealthStar information source — when accessed through the retrieval engine Internet Grateful Med (IGM):

```
Search Modes: (Subject, weight = 1), (Title, weight = 0.5)
Filter Translations: (Begin Year = begyear), (Publication
                      Type = publication)
```

Given the query Q = (AIDS, Diagnosis, Begin Year = 1980), the resulting set of queries, after applying the *Query-Customization* method is given below:

```
   Q1 = (Subject = AIDS, Subject = Diagnosis, (begyear =
   1980), Weight = 1)
   Q2 = (Title = AIDS, Subject= Diagnosis, begyear =
   1980), Weight = 0.5)
   Q3 = (Subject =AIDS, Title = Diagnosis, begyear =
   1980), Weight = 0.5)
```

New sources may be added to the application by including their descriptions according to the source domain model, and building the appropriate wrappers (task-psm bridges) between the *retrieve* task and the retrieval engines.

5 Reusing Knowledge Components in WIM

UPML has been defined as "a framework for knowledge system reuse"[8]. Components reuse is achieved in UPML by separating the specification of the different components of a software architecture. The separation between domain and PSM has been extended to the separation of task and PSM specifications to maximize reusability. Ontologies play a crucial role in modern knowledge systems to separate the reasoning components from the domain knowledge. In fact, ontologies are often defined as "shared conceptualizations" or "shared terminologies," putting the emphasis in the function of an ontology as a vehicle to achieve interoperability between different entities. Separation of components is needed to achieve reuse, while ontologies are needed to link the separated elements.

There is a class of components in UPML defined specifically to achieve reuse: bridges. A bridge can be seen basically as a mapping schema between two ontologies [11]. There are three kinds of bridges: Task-Domain, Task-PSM and PSM-Domain bridges. The WIM application is configured using a set of tasks, PSMs plus the PSM-Domain and Task-Domain bridges that link them to the Domain Models. Essentially, WIM needs to define bridges to the external information sources, these bridges connect the retrieve task from the library and the search engines of each source. These bridges, in our architecture, are called *wrappers*. These *wrappers* play two roles: a) they have the UPML specification of the source, and b) allow the interoperatibility with the source.

The other kind of bridges, needed in WIM, are the PSM-Domain Bridge. An example of this bridge is given by the PSM *query-elaboration-with-synonyms*, that uses an external domain knowledge, the MeSH thesaurus. MeSH do not speak explicitly of synonyms, instead of synonyms it uses the terms "See also" and "Related terms" to indicate semantically related terms. We use a bridge to map between the concept of "correlated term" (synonyms are terms with a correlation near to 1) used in the PSM *query-elaboration-with-synonyms*, and the concepts used in MeSH.

6 The WIM Multiagent Architecture

This section deals with the mapping between the knowledge components of WIM — within and without the library — and multiagent systems.

Different multiagent architectures could be used to build an application upon a library of knowledge components. This section will overview the kind of agents used in the WIM system together with a brief comment about some interoperability issues and the use of UPML as an Agent Capability Description Language (ACDL). We distinguish four classes of agents in WIM:

Problem-solving agents: they are the agents that solve the tasks in the library. The tasks they are able to solve and the available PSMs are both specified in UPML and registered to the Librarian.

Librarian: The librarian holds the UPML specifications of the components in the library: tasks and PSMs. UPML is the ACDL for describing the competencies of the Problem-solving agents: the PSMs they provide and the tasks they are able to solve.

Wrappers: They are the agents that accesses the heterogeneous information sources, solving the interoperability problem at the syntactic level and technical levels, while the customization PSM deals with the semantic level. From the UPML point of view, a wrapper is a software implementation of a bridge between a task in the library and an external PSM that solves that task. From a MAS perspective, wrappers are components used to agentify components that are not agents, like external information sources.

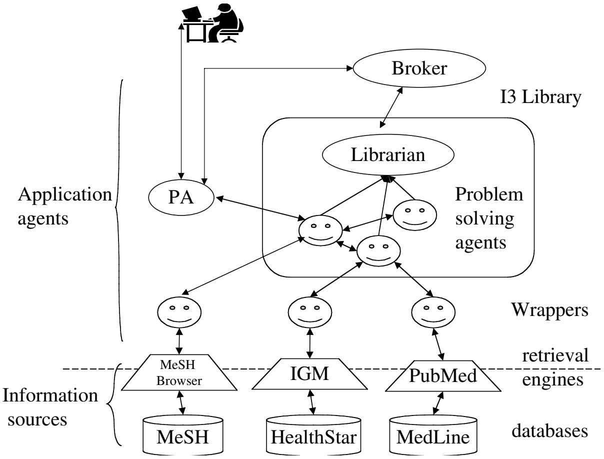


Fig. 2. WIM multiagent architecture

Application agents: These agents achieve elementary tasks needed by the application. WIM has two kinds of application agents: the *Broker Agents* configure components of the library to carry out the user needings and mediate between the user's personal agent and the problem solving agents. The *Personal Agents* are responsible for interacting the user asking for data and presenting him back the results of their previous requests. Information about the user's preferences and the history of previous cases, is used by the Personal Agent to help configuring or adapting the problem solving library for his particular interest, or for other users in a group oriented environment.

6.1 UPML as an ACDL

Describing agent capabilities is an active field of research, as this is an important issue to achieve interoperability in open agent architectures. The most widely adopted approach to solve the interoperability problem is the combined use of middle agents[21] and agent capabilities description languages (ACDL). An example of an ACDL is LARKS, and ACDL that is being included in the RETSINA multi-agent infrastructure[19] to be used within the matchmaking process (`http://www.cs.cmu.edu/ softagents/interop.html`).

The advantage of UPML is that it is able to describe the both the tasks an agent is able to solve, plus the methods it is equipped with to solve that task, and these descriptions are domain independent. UPML as and ACDL allows developing Information Agent with independence of the domain, an critical issue to achieve component reuse and scalability.

7 Conclusions

We have presented a framework (UPML) and an implemented cooperative information agents system developed using that framework (WIM). The architecture of UPML aims at maintaining a strict independence between domain (domain knowledge and ontology) and the information agents (with separate ontologies). The framework is based on using UPML as an agent capability description language. UPML defines bridges as a type of connectors that link domain-independent description of agent capabilities to concrete domain models. The advantage is that UPML allows the agents to talk about both the *tasks* they are able to solve and the *methods* by which they are able to solve them. Moreover, we have distinguished between library and application. A library is a repository of UPML descriptions of agents capabilities. An application is built selecting a collection of agents and developing the bridges required to link library-registered agents with domain resources and domain ontologies.

We have described the WIM application for medical information. Agents in WIM work with a domain independent ontology—the same by which tasks and PSMs are described. The WIM application is built by linking the "knowledge requirements" specified in UPML to specific domain resources using bridges—examples shown are the MeSH thesaurus and the EBM models. We have left out of the scope of this paper the brokering process, i.e. the process by which a collection of PSMs (and their corresponding agents) are selected to realize an application. We are currently developing several prototypes of automatic brokering that use UPML descriptions to configure on the fly an application for a given "input" task specification in the framework of the IBROW project (`http://www.swi.psy.uva.nl/projects/ibrow/home.html`).

Acknowledgements. This work has been supported by IST-1999-19005 Project IBROW "An Intelligent Brokering Service for Knowledge-Component Reuse on the World Wide Web".

References

1. M. Klusch (ed). Intelligent Information Agents. Springer, 1999.
2. Second International Conference on Cooperative Information Systems (CoopIS-94). Springer, 1994.
3. G. De Michelis, E. Dubois, M. Jarke, F. Matthes, J. Mylopoulos, K. Pohl, J. Schmidt, C. Woo, and E. Yu. Cooperative Information Systems: A Manifesto. In 4th Intl. Conf. on Cooperative Information Systems, Brussels, Belgium, 1996.
4. Wiederhold, G. Mediators in the architecture of future information systems. In *IEEE Computer* 25, 3 (March), 38-49, 1992.
5. Wiederhold, G. Intelligent integration of information. In *Proceedings of ACM SIGMOD Conference on Management of Data, Washington*, DC, pp. 434–437, 1993.
6. D. Garland and D. Perry (Eds.). Special issue on software architectures. *IEEE Transactions on Software Engineering*, 1995.

7. D. Fensel, V. R. Benjamins, M. Gaspari S. Decker, R. Groenboom, W. Grosso, M. Musen, E. Motta, E. Plaza, G. Schreiber, R. Studer, and B. Wielinga. The component model of UPML in a nutshell. In *Proceedings of the International Workshop on Knowledge Acquisition KAW'98*, 1998.
8. D. Fensel, V. R. Benjamins, E. Motta, and B. Wielinga. UPML: A framework for knowledge system reuse. In *Proceedings of the International Workshop on Knowledge Acquisition KAW'98*, 1998.
9. D. Fensel, C. Knoblock, N. Kushmerick and M. Rousset Workshop on Intelligent Information Integration III99. citeseer.nj.nec.com/316426.html
10. E. Motta Reusable components for Knowledge Modelling. Frontiers in Artificial Intelligence and Applications, 53 IOS Press, 1999
11. D. Fensel,S. Decker, E. Motta and Z. Zdrahal Using ontologies for defining task, problem-solving methods and their mappings. In *Proceedings of European Knowledge Acquisition Workshop (EKAW'97)*, Lecture Notes in Artificial Intelligence (LNAI), Springer-Verlag, 1997.
12. A. Gomez and V.R. Benjamins Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Components. Proceedings of the IJCAI'99 workshop on Ontologies and Problem-Solving Methods (KRR5), 1999.
13. J. Carbonell A machine learning perspective on information filtering. *ISMIS'00 Invited Talk*, October 11, 2000.
14. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, and 6Jennifer Widom J. D. Ullman. The Tsimmis project: integration of heterogeneous information sources. In *Proceedings of the Fourteenth International Conference on Data Engineering*, February 23-27, 1998.
15. Yigal Arens and Craig A. Knoblock. SIMS Retrieving and integrating information from multiple sources. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 562–563. ACM Press, 1993.
16. Michael R. Genesereth, Arthur M. Keller, and Oliver Duschka. Infomaster: An information integration system. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, ACM Press,1997.
17. A.Y. Levy, A.Rajaraman, and J.J. Ordille. Query-answering algorithms for information agents. In *Proceedings of the AAAI-96, Portland, Oregon, August 4-8*, 1996
18. Marian H. Nodine, William Bohrer, and Anne H. H. Ngu. Semantic brokering over dynamic heterogeneous data sources in InfoSleuth. In *ICDE*, pages 358–365. IEEE Computer Society, 1999.
19. Katia P. Sycara and Anandeep Pannu. The retsina multiagnet system: Towards integrating planning, execution and information gathering. In *Proceedings of the second international conference on Autonomous agents*, 1998, pages 350-351.
20. V. Torra Weighted OWA operators for synthesis of information In *Proc. 5th IEEE Inter. Conference on Fuzzy Systems*, New Orleans, USA, 1996, 966-971.
21. K. Decker,K. Sycara and M. Williamson. Middle-Agents for the Internet. P in *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 1997.
22. M. Gaspari, E. Motta and D. Fensel. Exploiting Automated Theorem Proving in UPML: Automatic Configuration of PSM from PSMs Libraries IBROW3 Deliverable, D4.1
23. A.R. Feinstein and R.I. Horwitz. Problems in the "evidence" of "Evidence-Based Medicine" *American Journal of Medicine*, 1997, vol 103, pages 529-535

An Autonomous Bidding Agent for Simultaneous Auctions

Nicoletta Fornara[1,2] and Luca Maria Gambardella[1]

[1] IDSIA, Lugano, Switzerland
{nicoletta,luca}@idsia.ch
[2] Università della Svizzera italiana, Lugano, Switzerland
nicoletta.fornara@lu.unisi.ch

Abstract. Market based mechanisms, such as auctions, will represent a widespread means of interaction for electronic commerce on the Internet. This paper is concerned with the problem of determining an efficient bidding policy in simultaneous auctions for goods that exhibit some complementarities and are substitutable. The First International Trading Agent Competition (TAC) held in Boston on July 8th, 2000 represents an interesting benchmark of this type of problems. The bidding policy of our trading agent Nidsia, who took part to the final phase of TAC competition, is described and some experiments testing our agent behavior are reported.

1 Introduction

Our research interest is to study a *bidding policy* for an agent whose task is to buy complementary and substitutable goods when such goods are sold in parallel auctions. The goal for the agent is to get the maximum utility for itself. This means that it buys and sells goods in order to maximize a given objective based on the value of goods bought and sold, and on the prices of the exchanges.

Choosing a bidding policy means deciding in which auction to bid, how much to bid and when to bid based on the goods already owned and on the information available on the running auctions. Some goods could be *complementary*, meaning that obtaining one good without another makes that good worthless. Some other goods could be *substitutable*, meaning that obtaining a certain bundle of goods can lower the value of obtaining another, or render it worthless [1].

Clearly this is a problem that everyone has to tackle when wanting to buy something using electronic commerce. However it is not easy to resolve optimally mainly because it becomes very complicated when multiple sellers offer various resources of interest.

The existing studies on this kind of problem are few. In fact most existing studies in auction theory have a different goal. Namely they try to devise a bidding policy for a team of competing auction agents, and then study what social outcomes follow for the multiagent system at whole. In this case the end goal is to study what happen if you adopt that policy in an auction market to resolve resource allocation and related problems in MASs [2].

In this paper we present the bidding policy used by our trading agent, Nidsia, in the First International Trading Agent Competition (TAC) [3]. The paper

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 130–141, 2001.

is organized as follows. In section 2 we describe important aspects of the TAC game. Section 3 explains the structure of Nidsia agent. Section 4 presents experimental results testing Nidsia's behavior. Section 5 presents some ideas for future improvements and concludes.

2 Description of the Trading Agent Competition Game

TAC is shopping a game in which eight software agents developed by multiple suppliers compete in a challenging market. For details on TAC visit http://tac.eecs.umich.edu. During the competition, the agents take part in a series of games. Each game is composed of various auctions that start at the same time, run in parallel and can last till the end of the game or finish in advance.

Each competitor is a travel agent with the goal of assembling a travel package for each one of its eight customers. These customers express their preferences for various aspects of the trip. The objective of the travel agent is to maximize the total satisfaction of its customers. The TAC server auctioneer runs at the University of Michigan and the various bidding agents connect over the Internet to the server to send bids and to update the market state.

A travel package consists of a round-trip flight, a hotel reservation in a good or bad hotel, and tickets to three types of entertainment events (Boston Red Sox Baseball, Boston Symphony or Boston Theater). A travel package is *feasible* if it contains rooms in the same hotel for every night between the arrival and departure dates. The soonest that any customer can leave is one day after arrival. An entertainment ticket is feasible and can be inserted in a feasible package if none of the tickets are for events on the same day, all of the tickets coincide with nights the customer is in town, and all assigned tickets are of different type.

From this brief description it is clear that flight tickets and the corresponding hotel rooms are complementary goods and that night rooms in the good or bad hotel and entertainment tickets on the same day are substitutable goods.

2.1 The Utility Function

Customers specify their preferences on a travel packages by a preferred arrival date AP; a preferred departure date DP; a reservation value for upgrading to the better hotel HU and a reservation value for each type of entertainment event BRS, SY, PH. In the travel packages pairs of arrival/departure days are equally likely, hotel reservation values are chosen for each customer uniformly in the range \$50 to \$150 and entertainment reservation values are chosen uniformly in the range \$0 to \$200. A travel package is specified by an actual arrival date AD, an actual departure date DD, a grand hotel indicator GH in {0,1}, and a ticket indicator for each event type BT, ST, PT, each in {0,1}. The utility for each customer is:

$$Utility = 1000 - travel\_penalty + hotel\_bonus + fun\_bonus; \tag{1}$$

where

$$travel\_penalty = 100 * (|AP - AD| + |DP - DD|); \tag{2}$$

$$hotel\_bonus = GH * HU; \quad (3)$$

$$fun\_bonus = BT * BRS + ST * SY + PT * PH; \quad (4)$$

The final score, for the trading agent, is the sum of the utility for each customer minus the expenses and minus the negative entertainment balances. If an agent tries to sell tickets it does not have, it is assessed a penalty of 200 for each ticket.

2.2 Auction Rules

Each game instance in TAC last 15 minutes. There are three different auctions: one for flights, one for hotel rooms and one for tickets. There is a separate auction for each good and each one of the 5 possible day of the travel package, up to a total of 28 parallel auctions. A detailed description of each auction type can be found in the appendix. Some important aspects of the three kinds of auctions are listed below.

Flights (8 auctions). Available flights are infinite. The price follows a random walk and does not depend on other bidders, but rather only on the auctioneer. If a bid is higher than the asking price, the bidder always get the flight. Further, since the price increase and decrease with equal probability, the expected change in price is 0.

Hotel Rooms (8 auctions). The auction for rooms is a standard English ascending auctions Mth price, with M=16 [4]. The final clear price depends on other buyer bids, so in a very competitive scenario prices could be very high. In this case, a bid higher than the asking price is not guaranteed to get a hotel room. Rather, depends on the behavior of other buyers. Further more, it is crucial to note that we does not know the outcome until after the auction is closed.

Entertainment Tickets (12 auctions). Entertainment tickets are sold using a continuous double auction [5]. In this case the final clear price depends on the other buyer and on the other seller bids. If a bid beats the ask-quote and comes in before other buyer bids, success is certain. If a bid is less than the bid quote and comes in before other seller bids, success is certain.

2.3 Consideration on the TAC Game

Because TAC game has some properties common to many problems, it is a good benchmark for studying the behavior and the performance of a software trading agent. Like in a real market place, goods are sold in parallel auctions, usually each auction, of a different vendor, has its own rules and characteristics and the available quantity of goods is limited. Further, like in the real life, the utility of getting some goods is related to conquering some other resources, so the trading agent's goal is to buy a bundle of goods. Still, in a real market place a buyer has to choose among various vendors where to buy substitutable goods. Keeping on with the analogies, in electronic commerce, like in the TAC game, it is not easy to know the name or important properties of the others competitors. It is not possible to repeat the same game, with the same customers' preference and the same contestants. Furthermore each auction game is too short to extract reliable statistical information about the behavior of competitors. This makes hard to introduce learning techniques in a trading agent for TAC.

On the other way, there are some unrealistic rules in the TAC game. The number of available flights is unlimited, so the corresponding auction is very similar to a shop with an expected change in price equal to zero. Another unrealistic characteristic is the absence of a penalty in the utility function, when the trading agent does not buy a feasible travel package for one of its customer.

3 Description of Nidsia Trading Agent

The are two main problems that the trading agent has to tackle in playing in the TAC game.

The allocation problem: given the set of goods it already holds, it has to decide how to optimally allocate them to its eight customers. The solution of this problem is relevant during the bidding phase and at the end of the game. It can be proved that the problem of finding the allocation that guarantees the maximum utility for the agent, can be formalized as an integer linear programming problem; specifically, as a weighted set packing problem, that is NP-complete. Although only few actual instances of the problem lead to a significantly longer solution times (in only 1% of 1866 problem instances, taken before the finals, the optimization took 6 or more seconds) [6]. Two of the competitors, RoxyBot and ATTac-2000, implemented an optimal strategy algorithm for allocation, while the other TAC participants, like us, used some form of greedy strategy [7].

The acquisition problem: given the set of goods it already holds, the valuation of a feasible travel package for each one of its customers, the current ask and bid price of the open auctions, decide where, how much and when to bid in the auctions. We call the result of this decision the *bidding action* of the agent and we call *bidding policy* the method that we used to choose the action in each possible state. The crucial point of this decision is that the trading agent does not know the independent value of the individual goods because it knows only the value of complete bundle of goods that forms a feasible package. Auction theory tells us when, where and how much an agent should bid only as a function of its separate valuation of a single resource given the auction mechanism [8]. Moreover, the agent does not know the final assignment of the auctions, that is the effect of its actions, therefore the problem can be considered as a decision problem under uncertainty. These considerations lead us to construct an agent bidding policy, in which bids for any resource are conditioned by the possible outcome of the other bids. Our approach is inspired by value iteration, a standard stochastic dynamic programming algorithm, and Boutilier, Goldszmidt and Sabata's paper on bidding in sequential auctions [1].

The allocation and acquisition problems are strictly connected; in fact it is necessary to use the allocation module to evaluate the goodness of every bidding action. It is possible to simplify the approach to these two problems facing them in a distributed manner. That is, do not try to compute one collective bidding policy that satisfies all the customers' requests at once. But try to compute a bidding policy for each customer independently, then the overall bidding policy is the sum of each isolated approach. The disadvantage of this strategy is in the loose of the centralized view of the problem. The agent looses the chance to exchange goods among its own customers, during the game. Because of its simplicity, we decided to use the distributed approach to compute the bidding policy and to allocate the goods at the end of the game in a centralized manner.

3.1 Bidding Policy

Our main goal was to investigate the more general problem of studying bidding policies in parallel auctions, so our agent is not restricted to this game. Nidsia computes its bidding policy in the same way for the entire game. What changes during an instance of the competition is the probability to really obtain the desired goods. Some other competitors, instead, decide to split their bidding strategy in two parts: one strategy for the first part of the game, with the only goal of keeping open the hotel auctions and another strategy for the final part of the game, in which all competitors start to bid seriously to get the goods.

When deciding its bidding action, the trading agent considers all customers in turn; this discussion therefore applies to an arbitrary customer. The agent's state s_t is a bit vector that describes the agent's current holdings for the current customer at time t. A bidding policy is a function from states to actions, where an action a is a vector of bids, one per auction, and each bid is a price-quantity pair.

The number of possible bids (and therefore actions) is numerable infinite, if one considers all possible discrete values of price and quantity. To reduce the space of actions to a manageable size, we only consider bids in which the quantity is 1 and the price for auction i is the ask-quote $q_{t,i}$ at time t plus a fixed increment δ. With this simplification, an action a is a bit vector, where $a_i = 1$ if a bid is submitted at price $q_{t,i} + \delta$ and $a_i = 0$ if no bid is submitted.

To further reduce the complexity of our model, we focus only on auctions for travel goods (i.e., flights and hotels), and therefore, by the nature of the TAC auction mechanisms, are primarily concerned with hotel auctions.

Another problem of this kind of model is that the time needed to reach a new state, s_{t+1}, is not known, because it depends on the time needed by the auctioneer to compute another price quote or to terminate the auction and assign goods to bidders. This asynchronism among the auctions makes impossible to forecast the behavior of the system during an entire game. Because of this, in computing the bidding policy our agent assumes, at every computation of the bidding policy, that the following state, s_{t+1}, will be the final one. Under these simplifying assumptions, our agent computes an optimal bidding policy.

For each customer and given current holdings, Nidsia computes the expected utility of each of 256 possible actions, corresponding to whether or not each of the 8 possible hotel rooms is included in the action or not. Then the trading agent bids according to the action that maximizes expected utility.

The expected utility $E[U(s_t, a)]$ of taking action a in state s_t is the sum over all possible states s_{t+1} of the probability $P(s_{t+1}|s_t, a)$ of reaching state s_{t+1} times the utility $V(s_{t+1})$ of state s_{t+1}.

The quantity $P(s_{t+1}|s_t, a)$ is computed as the product of the probability of the outcomes of the bids described by the action a, taken in state s_t, that lead to state s_{t+1}. At this level also the previous bids that are still active are taken in consideration. This formulation assumes that the probability distributions among the various auctions are independent.

The probability of obtaining item i is assumed to be near 0 at the beginning of the game and near 1 at the end of the game. Specifically, for the purposes of TAC game, these probabilities were given by the following equation for a straight line: $F(t) = mt + b$, that satisfies the condition: $F(1) = 0.1$ and $F(15) = 1$, that is $m = 0.9/14$, $b = 0.1 - m$ and 15 is the number of seconds in a TAC game. The probability of failing to obtain an item at time t is $1 - F(t)$.

The utility $V(s_t)$ to be at state s_t is taken to be the reward $r(s_t)$ for being in state s_t less the cost $c(s_t)$ of obtaining the items held in this state: i.e. $V(s_t) = r(s_t) - c(s_t)$. The cost $c(s_t) = \sum s_{t,i} c_{t,i}(h)$, where $c_{t,i}(h) = 0$ if Nidsia owns item i at time t, and $c_{t,i}(h) = q_{t,i}$ otherwise, for h representing hotel rooms. The reward $r(s_t)$ is taken to be the maximum possible value obtainable among all feasible packages that include the hotels indicated by bit vector s_t.

Formally,

$$E[U(s_t, a)] = \sum_{s_{t+1}} P(s_{t+1}|s_t, a) V(s_{t+1}) \tag{5}$$

$$P(s_{t+1}|s_t, a) = \prod_i P(s_{t+1,i}) \tag{6}$$

$$P(s_{t+1,i}) = s_{t+1,i} F(t+1, i) + (1 - s_{t+1,i})(1 - F(t+1, i)) \tag{7}$$

3.2 Allocation Strategy

Our agent allocates its goods to customers according to a fixed heuristic, rather than computing optimal allocations (using e.g. integer linear programming). One minute before the end of the game, Nidsia bids on flights that coincide with the hotel room auctions that it expects to win for each client. Also at this time, the initial endowment of entertainment tickets is greedily allocated to customers. Unused tickets are auctioned off, and useful tickets currently on sale are purchased. At the end of the game, Nidsia confirms that its customers have all the necessary goods to complete their travel, and it heuristically tries to allocate any unused goods so as to satisfy as many customers as possible.

4 Results

Nidsia was eleventh in the final TAC game competition. The resulting behavior of our trading agent during one of the games is the following: during the initial part of the game, the agent bids in every auction, because the prices and the probability of success are very low. So it bids for substitutable goods to be sure to get at least one of them. Then after some minutes, the agent start to react in accordance with the way of the prices are changing and it begin to concentrate its bids on the more convenient goods. In general Nidsia prefers short travel packages, because they have a much higher probability to be obtained entirely. Further our agent does not consider the utility of the entertainment tickets when it computes its bidding policy, so is not interested to remain longer to get the fun bonus.

During the entire game the probability distribution of success in the auctions is a crucial point. If the probability of success is underestimated, like in the first minutes of the game, the agent bids in almost all auctions, to avoid to be left without some important goods but it risks to buy much more goods than it really needs. On the opposite side, if the probability of success is overestimated the agent is quite sure to get some goods. It bids exactly on the goods to form the best feasible packages and it does not take into account the risk of being left without necessary goods.

The TAC competition showed that our agent was able to produce a satisfactory performance. We then decided to carry out some experiment to study Nidsia behaviour in different situations. In particular we wanted to manipulate some factors that where not under our control in the actual competition, like for example the composition of the group of competitors.

We have run several experiments, three of which we will now describe.

Experiment 1. In the first experiment Nidsia trading agent competes against seven dummy agents. Dummy agents are provided by the TAC server. The behavior of a dummy agent is not random, but is based on a simple and fixed strategy. This scenario is the less competitive one. The results are that Nidsia wins almost always (see Table 1). The average utility of Nidsia is $2'111$, while the average utility of the best dummy agent is 792. The difference between the mean utilities is statistically significant (2-tailed t-test, $p = 5.02\ 10^{-14}$).

Experiment 2. In the second experiment Nidsia competes against three instances of itself and against four dummy agents. This second scenario is more competitive than the previous one. The result is that the 4 Nidsia agents are placed almost always in the first places in the race (see Table 2). The average utility of the first Nidsia agents is 2656. It is possible to notice that in these games there are less negative scores because the four Nidsia agents exploit and share the available resources better. In this experiment, like in the previous one, the difference between the mean utilities, of Nidsia and dummy sample, is statistically significant (2-tailed t-test, $p = 6.71\ 10^{-15}$).

Experiment 3. In the third experiment two different instances of Nidsia compete against six dummy agents. In this scenario we want to compare the performance of Nidsia agent with the performance of an agent equal to Nidsia, but that computes a different probability of success for each auction. We call it Nidsia2. Our idea is to exploit the information, about the current game, that we can extract from the on going of prices in the market. In Nidsia2, if an auction is very competitive, meaning that its price-quote rises very quickly, the probability of success is lower than in a less competitive auction. It results that Nidsia2 is a risk adverse agent. Precisely, for each item i the probability of obtaining it, is given by the following equation for an exponential function:

Table 1. Results of the games in experiment 1

game n°	7845	7848	7856	7859	7863	7867	7868	7869	7870
1° Nidsia	3577	1673	3047	2420	2297	2226	2491	1757	2287
2° dummy	85	526.5	1993	360	1887	1910	356	589	1034
3° dummy	-1047	289.5	1957.5	-728	1526	1886	-807	-190.5	716.5
4° dummy	-2649.5	175.5	-150.5	-798	-1979	1820	-1269	-264.5	520.5
5° dummy	-3192	-3278	-2365	-1021	-2001	1394	-1979.5	-618.5	-1937
6° dummy	-3296.5	-3816	-3326.5	-1074.5	-2113	-1353.5	-3003	-2633	-2591.5
7° dummy	-3752	-5875.5	-3844.5	-2357	-4377	-3365	-4532.5	-2912.5	-3086
8° dummy	-4325	-5908	-4542	-4323.5	-4769	-3454.5	-5012	-3240	-5715.5
game n°	7884	7926	7936	7937	7941	7947		7847	**average**
1° Nidsia	1513	1140	2521	3504	2405	2252	**4° Nidsia**	-1330	2,111
2° dummy	1405	-1009	-792	1869	947	1309.5	1° dummy	207.5	792
3° dummy	618	-3909	-2079	1520	-2996	1307	2° dummy	-1042	-186
4° dummy	241	-4277	-4642	903	-3133	1002.5	3° dummy	-1174.5	-980
5° dummy	-2187	-4330	-4679	-397	-3526	-598	5° dummy	-1402.5	-2,007
6° dummy	-3233	-4819	-4956.5	-1381	-3776	-2625.5	6° dummy	-1511	-2,844
7° dummy	-3794	-5322	-5897.5	-4325	-3957	-4183	7° dummy	-4276	-4,116
8° dummy	-4084	-6089	-6317	-6339	-4907	-6960.5	8° dummy	-4442.5	-5,027

Table 2. Results of the games in experiment 2

game n°		7838		7846	7858	7864	7871	7873	7874	7880	7898	
Nidsia	**2°**	-122	**1°**	4095	2366	1986	3135	2640	2900	2802	2258	
Nidsia	**3°**	-183	**2°**	2676	2365	1316	2884	1606	2610	2247	1580	
Nidsia	**6°**	-3830	**3°**	2309	2204	-273	1843	1470	2022	1912	930	
Nidsia	**8°**	-6096	**4°**	2169	963	-1977	1445	1126	1258	1876	628	
dummy	1°	1488	5°	2010	-2567	-2545.5	555	-1248	-2542	-2288	-2976	
dummy	4°	-535	6°	664	-3521.5	-2688	-1425	-2655	-3517	-2473	-3799	
dummy	5°	-1479.5	7°	-182	-3730	-3086.5	-1791	-3255	-4231.5	-4124	-4829	
dummy	7°	-4446.5	8°	-1237	-5112.5	-3703	-2398	-3853	-4890.5	-6297	-6066	
game n°		7881	7899		7882		7883		7887		7890	**average**
Nidsia	**1°**	2453	3440	**1°**	3532	**1°**	3471	**1°**	2896	**1°**	1982	2656
Nidsia	**2°**	1739	1925	**2°**	2498	**2°**	3136	**3°**	1993	**2°**	1734	2008
Nidsia	**3°**	1541	221	**3°**	1536	**6°**	768	**5°**	1764	**4°**	1453	1058
Nidsia	**7°**	-1866	-2536	**5°**	-21	**7°**	754	**6°**	1120	**5°**	1232	5
dummy	4°	-459	104	4°	999	3°	2903	2°	2174	3°	1615	-185
dummy	5°	-603	-77	6°	-2398.5	4°	1713	4°	1822.5	6°	-420.5	-1328
dummy	6°	-1524	-501.5	7°	-2614.5	5°	1448	7°	959	7°	-2441.5	-2092
dummy	8°	-2881	-5648.5	8°	-3738	8°	-3196	8°	-2933.5	8°	-3767	-4011

$$F'(t,i) = e^{-((\Delta price/\Delta time)/\Delta rate_{max})} F(t,i) \tag{8}$$

where

$$\Delta rate_{max} = 100 \tag{9}$$

The probability of failing to obtain an item at time t is $1-F(t,i)$. In Figure 1 three different kinds of price trends are shown and in Figure 2 are shown the corresponding probability trends. In this experiment Nidsia competes against Nidsia2 and against 6 dummy agents. The two Nidsia agents are placed first and second in almost all the test games, but Nidsia2 does not perform always better than Nidsia. A possible explanation is that the information that is possible to exploit during a single game, is not enough to have a more successful agent. The difference between the mean utilities of Nidsia and Nidsia2 sample, is not significant (2-tailed t-test, $p = 0.46$).

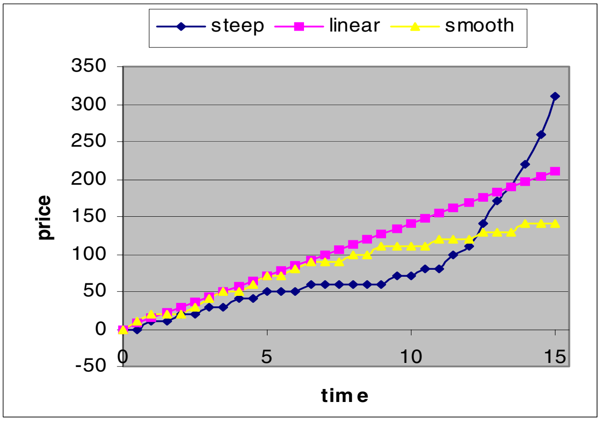


Fig. 1. Some price trends

4.1 Comments

The main problem of our trading agent is that when it decides (two minutes before the end of the game) on which flights to bid, relies on hypothesis that all its active bids will be successful. But if it overestimates the probability of success, especially in some high competitive game, it gets stuck in a situation where it has the flights, that usually are very expensive, but it does not have the necessary rooms to compose a feasible package. We notice also that the expected utility of the best action could be near the expected utility of some other actions and we decide to choose the action that bid for the little number of goods.

With regard to the allocation module, at the end of the game, Nidsia uses a heuristic strategy, although during the official competition it gets 95% of the optimal utility. As concern the distributed approach to tackle the acquisition

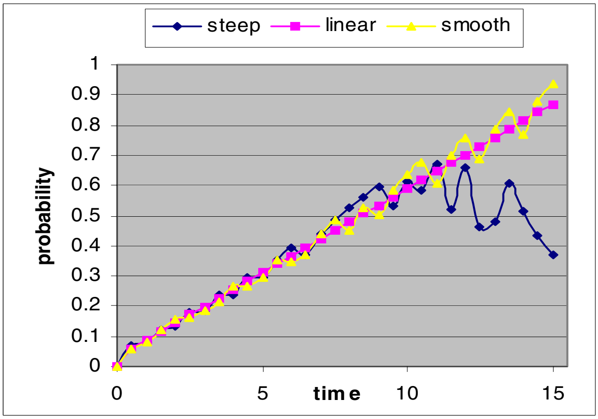


Fig. 2. Probability trends corresponding to above price trends

problem during the game. It is possible to notice that it is not very relevant, because in many games of the TAC competition, few goods are bought during the first and middle part of the game, so there are few goods to be redistributed among the customers.

TAC is not a zero-sum game, but there is high competition for hotel room resources. In some games it is possible to notice that trading agents compete strongly on a special good, so the price rises and the utility average decreases. While in some other games there is less competition and the utility average is higher.

5 Conclusion

As we already said, our trading agent's algorithm was not tailored to the particular auction mechanisms of TAC, but rather is more general in its applicability to combinatorial auctions of substitutable and complementary goods. As a result, the implementation of our algorithm required some strong simplifying assumptions. Nonetheless, Nidsia's overall performance illustrates the promise of this general method.

In a future work it would be interesting to compare a *centralized* with a *distributed* approach. The former consists in computing the bidding policy considering all the customers' requests at one. The latter approach consists in computing separately a bidding policy for each customer, as Nidsia did.

Further, it could be interesting to improve the agent using a broad the set of possible value of bids in the auctions. Some other improvements consist in trying to introduce some learning feature in the trading agent. For example it could learn some of the parameters that it uses, like the instant when it starts to bid for flights, the amount to increase the ask price when bid in auctions. Or harder, try to learn some characteristics of the other competitors, like their preferences or if they are risk neutral, risk averse or risk seeking.

Acknowledgements. We thank Marco Colombetti for advice and for helpful comments on a draft version of the paper.
We thank the TAC team for their continuous collaboration and helpfulness.
We thank also Amy Greenwald for useful suggestions.

References

1. Craig Boutilier, Moises Goldszmidt and Bikash Sabata. Sequential auctions for the allocation of resources with complementarities. Proceedings of IJCAI-99, 1999
2. T. W. Sandholm. Distributed rational decision making. In G. Weiss, editor, Multiagent Systems, pages 201–258. The MIT Press, 1999
3. Michael P.Wellman, Peter R. Wurman, Kevin O'Malley, Roshan Bangera, Shou-de Lin, Daniel Reeves, and William E. Walsh. A trading agent competition. In Press, 2001
4. Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. Decision Support Systems, 24:17-27, 1998
5. D. Friedman and J. Rust, editors. The Double Auction Market: Institutions, Theories, and Evidence. Addison-Wesley, New York, 1993
6. Peter Stone, Michael L. Littman, Satinder Singh, and Michael Kearns. ATTac-2000: An adaptive autonomous bidding agent. In Proceedings of the Fifth International Conference on Autonomous Agents, 2001. To appear, available at http://www.research.att.com/pstone/papers.html
7. Amy Greenwald and Peter Stone. Autonomous bidding agents in the trading agent competition. IEEE Computing, 2001. To appear. Available at http://www.research.att.com/pstone/papers.html
8. R. Preston McAfee and John McMillan. Auctions and bidding. Journal of Economic Literature, 25:699-738, 1987

Appendix

A schematic description of the three auctions type.

Flights

number of sellers: single seller auction;
number of auctions: one auction for each day and direction;
time of clear: the auctions will clear continuously;
clear price: if the current quoted price is less than your bid, you will get the ticket at the current price. If the current quoted price is greater, your bid will sit in the auction until the price becomes less than or equal to your bid, at which point you will get it for the price you bid. Of course, this may never happen if the price never goes that low. Matching is based on earliest submission time;
time of price quote: quotes are continuous;
price quote: the ask quote is set according to a stochastic function. The process used to update flight prices is a random walk, starting between \$250 and \$400 and perturbed by -\$10 to \$10 every 30 to 40 seconds. Prices will always be in the range \$150 to \$600. All distributions are uniform;

availability of goods: unlimited;
bid admittance conditions: all buy bids are accepted, multi-point buy bids are allowed;
bid withdrawal conditions: bids can be withdrawn at any time if they have not transacted;
reservation price: the ask quote;

Hotels

auction type: standard English ascending auctions, Mth price;
number of sellers: single seller auction;
number of auctions: there are two hotels in Boston: the Boston Grand Hotel and Le Fleabag Inn, so there is one auction for each combination of hotel and night.
time of clear: it will clear at the earliest, 1 minute after the game has started or at latest when the game ends. It might clear earlier if there has been no activity (no bids) for a random chosen period of time;
clear price: is the price of the lowest winning bid, matching is based on earliest submission time;
time of price quote: continuous quotes;
price quote: the ask quote is the 16th highest price;
availability of goods: 16 rooms;
bid admittance conditions: must beat quote, multiple buy points in bid allowed;
bid withdrawal conditions: no withdrawal, but since the hotel auctions are ascending, once a bid is "out of the money" (i.e., the ask price is above the offer) it is effectively decommitted;
reservation price: no reserve price, minimum bid is $0;

Entertainment Tickets

auction type: continuous double auction (CDA);
number of sellers: multiple seller;
number of auctions: there will be one auction for each event-night combination.
time of clear: continuous clears;
clear price: when a new bid is processed, the auction check whether the offered price would match the lowest (best) existing sell bid, and vice versa. The transaction price is the price that was put in first;
time of price quote: continuous quotes;
price quote: the ask quote is the price an agent would have to bid over in order to place a winning buy bid (the least of the sell bids). The bid quote is the price an agent would have to bid under in order to place a winning sell bid (the biggest of the buy bids);
availability of goods: depends on the seller;
bid admittance conditions: any type of bid allowed;
bid withdrawal conditions: bids in the entertainment auctions can be withdrawn at any time if they have not transacted;
reservation price: no reserve price;

Optimality and Risk in Purchase from Multiple Auctions

Onn Shehory

IBM Research Lab in Haifa, the Tel-Aviv Site
2 Weizmann St. Tel-Aviv 61336, Israel
onn@il.ibm.com

Abstract. In the face of multiple electronic sales sites, buyers can benefit from considering multiple opportunities and devising their purchase strategy to reduce risk and increase expected utility. However, human users cannot approach and rapidly monitor multiple sites. In particular, sites where prices change dynamically such as auction sites pose a major difficulty for human concurrent activity. Even without concurrency, the majority of human users do not have the ability or the resources to compute optimal purchase decisions. Such activities can be performed by computational agents. In this paper we present mechanisms that allow agents to perform purchases on behalf of users. In particular, we devised methods that allow an agent that faces multiple dynamic sales sites to compute bids that optimized the expected utility of the user, or instead reduce and manage the risk of the purchase. The proposed mechanism are currently embedded in agents we develop in our lab.

Introduction

The number and variety of web sites where one can buy goods, either as a consumer or as a business, has grown beyond the capacity of a single human buyer. Human users, even when looking for a single good, can enter both static sales sites (catalog-based) and dynamic sales sites (auctions and exchanges). Price differences among identical products introduce and opportunity for profit based on comparison. Such price comparison is already provided as a service by sites such as MySimon [1], DealTime [2] and others. Yet, the comparison problem further extends to the following: 1. How should goods traded in dynamic trade mechanisms be compared, in particular in cases where opening and closing time of e.g., auctions, are different for the different goods? 2. How should goods which are similar, however not identical, be compared as alternatives to each other (e.g., one may be willing to compromise some attributes to get a better price)? In this document we provide mechanisms that explicitly handle the first question and provide hints on how the latter should be addressed. In particular, we address the purchase from multiple auctions which may time-wise overlap, and for which we may have some statistical historical price profile. Our mechanisms allow maximization of the expected utility, as one would seek for rational purchase decisions. However, our solution departs from the common approach to electronic purchase, as it does not require utility

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 142–153, 2001.

maximization of the expected utility. Instead, it allows the user to require a minimal probability of a successful purchase. The agent computes bids that optimally respect this requirement and accordingly performs its purchase activity. An algorithm for an agent to implement these mechanisms is provided. The algorithm takes into consideration the dynamic nature of electronic purchase. In particular, it handles changes in site and product availability, in prices, in the user's valuation of the sought good and in the probabilities of winning a good. The purchase of a single item from multiple dynamic sales sites is fully addressed by our algorithm. Hints on the purchase of multiple items are provided.

The rest of the paper is organized as follows: we start by presenting definitions and notations. We proceed by presenting our algorithm, initially concentrating on actions preceding the actual bidding and purchase. We then present the two mechanisms for bid computation, the first addresses the requirement on winning probability and the second addresses the maximization of the expected utility. Then, the rest of the algorithm is described, and finally we conclude and discuss future work.

Definitions and Notations

Our algorithm refers to auctions in general, however since the majority of auctions on the web are English auctions, we explicitly handle only them. Nevertheless, due to the equivalence between different types of auctions [3] we believe that adjustments to other auction mechanisms should not be a major difficulty.

An electronic (English) auction a is a server system where k identical items are sold. A degenerate case is the one where only one item is sold. An auction has a starting time t_0 and a closing time t_c, thus it is active during a time interval $\mathbf{T}=[t_0 ,t_c]$. It also has an initial (reservation) price p_0, a minimal increment δ (in cases where the latter is not explicitly defined, δ is the smallest currency unit), and a closing price list $\mathbf{P}_c=\langle p_k, p_{k-1},\ldots, p_1\rangle$, where p_i is the price of the ith item sold. Once an auction becomes active, the auction server accepts bids sent to it by clients. We ignore here the technical details of how the bids are submitted and how the server system determines whether a bid is valid. A valid bid is a bid submitted during the time interval $\mathbf{T}$, and is greater by at least δ from the lowest bid among the currently active bids. At any moment during $\mathbf{T}$ there may be up to k active bids. To simplify representation and analysis, we assume that there are always exactly k active bids, where all bids are initially set to $p_0 - \delta$ by the server. Thus, at any time during $\mathbf{T}$ there is an ordered list $\mathbf{B}=\langle b_k, b_{k-1},\ldots, b_1\rangle$ of the active bids, where b_1 is the highest bid (note that in case of bids equal in value, the time of submission determines the order, i.e., the earliest is first). Each new valid bid that arrives at the server during $\mathbf{T}$ is inserted into its appropriate location in B, thus pushing all of the lower bids one place down the list. Note that, as a result, b_k, which is the lowest bid, is dropped off the list. At the closing time t_c, the server stops accepting bids, and the latest list of bids $\mathbf{B}$ becomes the closing price list $\mathbf{P}_c$. The winners of the items sold in the auction are the bidders whose bids entered the closing price list. We do not discuss here the processes proceeding winner determination. We assume that the closing prices are either published by the auction server or can be learned by clients in other ways (e.g., by frequent monitoring of the site).

An item *x* sold in an electronic sales site *y* has a vector of attributes A_{xy}. The values within the vectors are each in the range [0..1], where 1 refers to having the attribute, 0 refers to not having the attribute, and values in-between represent partiality, when applicable. Commonly, buyers seek an item that exactly matches a given set of attributes. Two items are considered equal when their attribute vectors are equal. In times, buyers are willing to compromise some of the attributes. In such cases, it is useful to have a measure of the similarity between the requested item and its substitute. The similarity between non-identical items is computed as the cosine of their attribute vectors.

The Algorithm

To offload the human user from its monitoring and trading burden when buying from multiple sites, we develop an automated agent that monitors sites, participates in trade and makes the appropriate bidding decisions on behalf of its human user. We assume that the user is interested in buying an item[1] from multiple sales sites, and that the user is price sensitive, however has no discrimination among the sites. We also assume that the user is willing to compromise some of the attributes of the requested item, as long as other attributes compensate for this compromise, or there is an appropriate discount due to lacking attributes. We further assume that the maximal price a user is willing to pay for the requested goods depends on time. That is, since the user has some (possibly implicit) deadline *d* for purchasing the goods, he/she may be willing to pay more as time elapses without a successful purchase. The user also has a maximal valuation $V_{\max}$ that is the upper bound to his willingness to pay for the requested good. We denote the time dependent valuation by $V(t) \leq V_{\max}$.

To simplify our discussion, we initially refer to the case of a user buying a single item from multiple sites. Although some sites may run multiple auctions, we refer to each auction as if it is held on a different site. This should have no significant effect on our solution, as each multi-auction site can be perceived as a collection of single-auction sites. The purchase algorithm and the bid computation methods are described in the following sections.

The Algorithm: Site Location and Selection

The agent's purchase activity is initiated by its user. Then, the agent needs to locate the sites where the requested item is sold and select the one relevant to the user's requirements. Below we provide the algorithm for these initial steps. Further activities (i.e., bid computation and submission) will be addressed in proceeding sections.

1. The user places with the agent a request for purchasing an item *m*. The request includes a vector of the attributes A_m, the deadline *d*, $V_{\max}$ and (optionally) $V(t)$. To each attribute, the user may attach an importance measure in the range [0..1], where 0 means not important and 1 means

[1] The purchase of multiple identical items is not addressed in the core part of the algorithm. Some details on how this should be done are provided in a later section in the paper.

mandatory (i.e., cannot be compromised for). Attribute importance not set by the user is automatically set by the agent's default values.

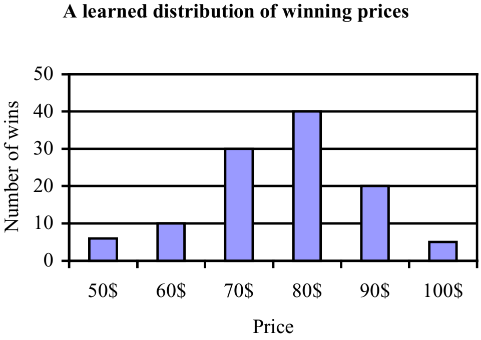
2. The agent constructs a list of candidate sites S_c (some known in advance) in which m (or sufficiently similar items) may be sold.
3. The agent approaches each site in S_c, searching for full and partial matches, placing URLs of full matches in a list S. Partial matches (i.e., similar items) are placed in S', unless some mandatory attribute does not match.
4. Each entry in S' includes a URL, an item and a similarity measure between m and the substitute. In computing the similarity measure, each attribute is multiplied by its importance measure to express similarity as viewed by the user.
5. The agent separates S into two lists, a list of fixed price sites (catalog-based) S_f and a list of dynamic price sites S_d. Similarly[2], S' is separated into S_f' and S_d'.
6. The agent locates the best (lowest) price of m among the sites in S_f, denoted p_f.
7. If $V_{\max} > p_f$, then set $V_{\max} = p_f$, to prevent cases of buying in dynamic mechanisms for more than the best fixed-price.
8. For each site i in S_d, the agent retrieves the time interval $\mathbf{T}_i$, the active bid list $\mathbf{B}_i$ and the price increment δ_i.
9. The agent selects a list of relevant sites S_r as follows. For each i in S_d,
    a. Time irrelevance: if ($t_c > d$) or ($t_c <$ current time), then omit i.
    b. Price irrelevance: if $b_k+\delta_i > V_{\max}$, then omit i (assuming no prior bids at i by the agent).
    c. If not omitted, add i to S_r.

At the end of step 9, the agent has a list S_r of sites where the requested item m is sold during the relevant time period, and where the current price is lower then the lowest fixed price available. Next, the agent needs to iteratively compute appropriate bids and select the right sites to which the bids will be submitted. It should also monitor the sites for changes and update S_r whenever new sales become available. These activities are described below.

Bid Computation, Submission, Monitoring and Update

We assume that, for each site i in S_d, a probability distribution function f_i of the winning price is given, or can be learned from historical wins. A learned winning distribution of winning prices may be of the type depicted in Fig. 1. The learned probability distribution will be a similar chart, though normalized.

[2] For clarity, the use of S' is omitted from the rest of the algorithm. In general, its use is similar to the use of S, however substitute items require a separate value calculation based on attribute similarity.



f_i may be time dependent, as differences in the day of week, the time of day and other temporal parameters can significantly affect the results of dynamic sales. Time dependence of this type has no direct effect on our analysis. Note that f_i is not necessarily an analytical function. It may be a table of pairs ⟨*price*, *probability*⟩, which is sufficient for practical use. In the case that f_i is an analytical function, we can compute the probability that auction *i* will terminate with a winning price at most *p* by integrating f_i from 0 to *p*.

It is common to assume that rational agents should attempt to maximize expected utility. Yet, this assumption holds only for risk-neutral agents.[3] Since the human on behalf which the agent participates in the purchase process is not necessarily risk-neutral, our agent must be able to handle different risk attitudes. The case of a risk-averse buyer is easy to handle. The simplest risk-free refuge for buyers that seek a guaranteed purchase is buying from a fixed-price site at the best available price (we assume here that the sites can be trusted). Yet, we need to handle the risk-neutral and risk-prone cases. Further, risk attitudes may lay in-between the three basic attitudes. For instance, one may be willing to take a *limited* risk of not winning a requested item, in exchange to some monetary compensation. To address risk attitudes, our agent supports two approaches for determining the maximal bid it should consider:

1. The user decides what the (minimal) probability q_{win} of winning the requested good he/she seeks is. This approach is better for users who would like to manage their risk. The exact probability they select depends on their risk attitude.
2. The user seeks utility maximization regardless of what the probability of winning is. It may happen that the expected utility is high, however the probability of winning is not. This approach is best for risk-neutral users.

The first approach requires, given a probability q_{win}, a deadline *d* and the valuation of the good *V*, the computation of the set of maximal candidate bids b_i to be placed in each auction *i* such that the joint probability of winning the good once, by (possibly) participating in all of the auctions, is at least q_{win}. The second approach skips the

[3] The utility function of an agent can be modified to measure risk attitude. Thus, the utility maximization assumption may hold regardless of risk attitude. Yet, the required change is non-trivial.

probability computations of the first one. It only requires the computation of the optimal bidding strategy given d and *V*. Below we provide the computational details of both of these approaches.

Minimal Probability

We seek a mechanism that will allow a user, or its agent, to decide in which auction to participate, for how long and, in particular, what bid to submit, such that:

1. The joint probability of winning the requested good (once) is at least q_{win}.
2. The expected value of the bid is minimized (given q_{win}).

We start from examining a single auction. To guarantee a probability of at least x for winning a single auction we need to find a bid b_x such that

$$\int_0^{b_x} f(b)\,db = x$$

where $f(b)$ is the probability distribution of the winning values of the auction. An analytical solution of the above is not always possible, however a numerical solution is rather simple. The ability to participate in multiple auctions improves the buyer's situation. We examine this case below. Assume, for simplicity, no correlation between the auctions. We require that the probability of winning the requested good, when (possibly) approaching all of the auctions, will be at least X. Denote the probability of winning auction i by x_i. The probability of not winning auction i is $(1-x_i)$ and the probability of not winning any auction is the product of these. Thus, the probability of winning (at least[4]) one auction is given by:

$$X = 1 - \prod_i (1 - x_i)$$

Since we would not like to pay different prices at different auctions, we require that the candidate bids for the different auctions be all the same. That is,

$$\forall i, j \quad b_x^i = b_x^j = b_x^*$$

where b_x^i is the candidate bid for auction i given a probability of winning x, and b_x^* is the sought, equal for all auctions, bid. Recall that the probabilities x_i are not given, and must be computed in the same way as computed in the single auction case. That is,

$$x_i = \int_0^{b_x^*} f_i(b)\,db$$

This implies that we need to find b_x^* such that

[4] Although the probability takes into account winning more than one good, such winning can be prevented by technical means of the purchase mechanism (e.g., quit once purchase is made).

$$X = 1 - \prod_i \left(1 - \int_0^{b_x^*} f_i(b)db \right)$$

Since we assume that the probability distributions f_i are known, and since X is provided by the user, it is possible to numerically compute b_x^*. The requirement that the expected payment for the good be minimized is fulfilled by the setting of all bids equal to b_x^*. The proof of this minimization is straightforward.

In case that the user would like to have a chance of winning by placing a bid lower than b_x^*, it will be necessary to place at least one lower bid b_k in some auction k. To keep the joint probability of purchase at its value X, at least one of the other bids must be raised. Yet, this will result in a non-optimal expected payment. Since, in the general case, the following holds

$$1 - X = \prod_i \left(1 - \int_0^{b_x^i} f_i(b)db \right)$$

setting one bid b_k lower than b_x^* will pose a new requirement on the other bids, as follows:

$$\frac{1 - X}{1 - \int_0^{b_k} f_k(b)db} = \prod_{i \neq k} \left(1 - \int_0^{b_x^i} f_i(b)db \right)$$

To optimize this expression, we require that

$$\forall i, j \neq k, \quad b_x^i = b_x^j = b_x^{\bar{k}}$$

This requirement is similar to the one of the previous case. $b_x^{\bar{k}}$ can be computed numerically.

To summarize, we have so far provided a mechanism for computing the lowest candidate bids the agent should place such that a requested probability of winning the good is guaranteed.

Utility Maximization

The second mechanism we seek should allow our agent to compute the bids it should submit in order to maximize the expected utility, with no guarantees regarding the probability of success. We expect that bids computed here will, on average, be lower than those computed by the previous mechanism. As we mentioned above, this mechanism is most appropriate for an agent who purchases on behalf of a risk-neutral user.

The expected utility $U_i(b)$ of an agent from bidding b in a single auction i is given by

$$U_i(b) = \big(V(t) - b\big)q_i(b)$$

where $V(t)$ is the maximum the agent is willing to pay for the good at time t and $q_i(b)$ is the probability of bid b winning the auction. $q_i(b_0)$ for a specific bid b_0 can be computed by integrating the probability distribution as follows:

$$q_i(b_0) = \int_0^{b_0} f_i(b)\,db$$

We assume that, in the general case, $q_i(b)$ does not depend on our agent's activity, or such a dependency is negligibly small. This results from the probability being computed based on multiple auction encounters as well as a large number of auction participants.

The maximization of the utility in a single auction is arrived at by computing the bid for which $U_i(b)$ is maximal or, instead, the derivative is nullified. That is,

$$\frac{dU_i(b)}{db} = \frac{dq_i(b)}{db}(V(t) - b) - q_i(b) = 0$$

The value of b^*_i that satisfies this requirement on the relationship between $V(t)$, b and q_i can be computed numerically and, when $q_i(b)$ is a given analytic function, the result can be obtained analytically. Yet, our agent is facing multiple auctions, hence we must provide it with means for maximizing the expected utility with respect to this multiplicity. Since we assume no correlation between the auctions, the expected utilities from the auctions are independent as well. We also assume that the agent, when in need of a single good, will only actively bid in one auction at a time (though considering all available auctions). Consequently, the expected utility from the ability to participate in all of the available auctions is

$$U = \max_i(U_i(b_i^*))$$

The agent should bid in a sequential manner (and only for positive utilities), where the bid it should place is b^*_i, to be placed in auction a* such that

$$a^* = \arg\max_i(U_i(b_i^*))$$

Other competitors participating in a* may submit bids greater than b^*_i, thus outbidding our agent. To decide whether it should place a new bid in a* or move on to another auction, our agent should re-compute the expected utility of that auction. Here, $U_i(b)$ should be modified to consider the opponent's bid b_o, as follows:

$$U_i(b) = \big(V(t) - \max(b_o + \delta_i, b)\big)q_i(\max(b_o + \delta_i, b))$$

Note that here the opponent's bid was added the minimal required price increment. This sum is the minimal bid our agent can place to beat b_o. After this computation, the auction in which a bid should be placed will be selected as before.

At this point we have two bid computation mechanisms. We may return now to the agent's bidding algorithm.

The Algorithm (Continued): Bidding

The previous sections suggested two methods for bid computation. The decision on which method should be used will depend on user preferences, and in particular on risk attitude. Using the appropriate bid computation mechanism, the agent should follow the steps below:

```
10. While the requested item was not purchased and the deadline has not passed, do
11. Periodically, monitor sites and update $S_r$.
12. For each $i$ in $S_r$,
    a. Retrieve the active bid list $\mathbf{B}_i$, the time interval $\mathbf{T}_i$ and the price increment $\delta_i$.
    b. If ($t_c > d$) or ($t_c <$ current time), omit $i$ from $S_r$.
    c. If $b_k+\delta_i > V_{max}$, omit $i$ from $S_r$ (since the lowest bid is above the agent's maximal valuation of the good).
    d. If $V_{max} > b_k > V(t)$, skip $i$, but do not omit from $S_r$ (since the current maximal value $V(t)$ may increase thus allow for future beneficial bidding at $i$).
13. For auctions not omitted or skipped
    a. In the case of the minimal probability requirement
        i. Compute the candidate bid
        ii. Submit the computed bid to the auction in which the current probability of winning is highest. If the auction is not active, break.
        iii. Monitor sites, re-compute probabilities and bids
        iv. If bypassed by competing bids, switch to another auction when applicable (see (ii)).
        v. If bid withdrawal is allowed, always withdraw to switch to the auction with highest probability.
    b. In the case of utility maximization
        i. Compute the expected utility maximizing auction a* and its right bid b*.
        ii. Submit $b_k*+\delta*$ to a*. If a* is not active yet, break.
        iii. Monitor a* for changes. Bid above competitors up to b*. This bidding may be done with a time delay to prevent automatic acceleration of the bidding process.
        iv. Concurrently, monitor sites, re-compute expected utilities.
        v. Once a* has changed (i.e., another auction became a*), switch to new a*.
14. The above should be performed iteratively until the deadline has passed, or the item was purchased, or no beneficial auction was left.
```

The algorithm above allows an agent that attempts to buy a single item from multiple sites to do so while providing either a guarantee on the probability of winning the item, or a maximized *expected utility*, given the probability distributions of wins at the sales sites. Note that the algorithm can handle time-dependent changes in the user's valuation of the requested good and similar changes in the probability of winning. Yet, one may claim that other agents, once they become aware of our agent's strategy, can counter-react manipulatively, thus reducing the utility of our agent. We hypothesize that the claim may be valid in small markets, in particular in cases where identities of agents are known to each other. However, in large markets where agents are anonymous, the claim seems less likely to apply. This concern will be investigated in future work.

Buying Multiple Items

The algorithm above does not explicitly address the case of an agent buying multiple identical items from several sites. To handle such cases, the computation of bids is not changed. Nevertheless, the decisions on which bid(s) to submit and to what auction(s) will be somewhat more complex. In particular, when the agent considers an auction *i* where the current bid list is $\mathbf{B}_i$, it needs to examine the effect of placing a new bid in *i* on previous bids it may have in $\mathbf{B}_i$. Suppose the agent computed a bid b to be submitted to *i*. As we have seen before, *b* should satisfy $b+\delta_i > b_k$, where b_k is the lowest bid in $\mathbf{B}_i$. It may happen that b_k is a bid placed by our agent. Thus, a naïve use of the bidding strategy will result in the agent outbidding itself, possibly recurrently. To prevent such cases, two measures must be taken. Firstly, the agent should check that new bids in an auction do not drop other bids off it. Secondly, if they do, the average expected utility of all of the agent's current bids should be computed and compared to the same average computed given the effect of placing the new bids. Only in case that the newer average is greater than the older one should the agent submit the new bids. Otherwise, the agent should not submit bids to *i* prior to further changes in information relevant the bidding.

The concepts presented in this section provide initial guidelines for enhancing our bidding mechanism to allow buying multiple items from multiple auctions. Further investigation of this issue is necessary.

Related Work

Multiple studies on automated negotiation in general, and on bid computation in particular, were presented in recent years. Environments where auctions can be tested with various bidding mechanisms were provided by, e.g., AuctionBot [4], FishMarket [5] and Kasbah [6]. Although they support bid computation, these studies do not address the problem of a single agent participating in concurrent multiple auctions. Bidding in double auctions is similar to participating in multiple auctions. Methods for bid computation and agent participation in iterative and continuous double auctions are provided in, e.g., in [7,8]. These methods do not explicitly consider the effect of dynamic changes in the knowledge of the agent regarding the probability of winning, nor do they manage the risk attitude of the user. A multi-agent system that supports cooperative bidding in multiple auctions was presented in [9]. Although cooperative bidding appears beneficial, self-interest may prohibit participation of agents in it. Another approach to bid computation is presented by [10], where the bidding strategy may evolve based on dynamic changes in knowledge. Yet, that study refers to multiple auctions run sequentially by a single authority. Our work addresses multiple auctions running concurrently, possibly with different starting and ending times. The auctions run independently by multiple different sites. Similar to previous recent work [12], we provide methods for bid computation. However, unlike others, our methods allow consideration of risk attitudes. They also allow the user to provide time-dependent valuations of the requested good. Finally, our methods provide not only the bids to be submitted, but the auctions to which these should be submitted as well. To our best knowledge, no prior work provided a bidding method that supports all of these attributes.

Conclusion

In this study we present an algorithm as well as computation mechanisms that allow an agent to compute bids and actively participate, or at least take advantage of the ability to participate, in multiple auctions. The major contributions of the paper are mechanisms that provide either of the following: 1) maximization of the *expected* utility from buying a good; 2) a guarantee on the probability of winning the good. Based on these, our mechanisms allow agents to manage the risk of the user. This is different from the common approach to purchase via agents, where agents are assumed to behave rationally, in the sense of utility maximization. We allow the user to specify the level of risk, and the agent will place the right bids based on this risk. Yet, for the risk neutral user, we provide a utility-maximizing mechanism. An important attribute of our solution is that it allows for dynamic changes in user valuation of the requested good and computes future expected utilities (based on the relevant available information). These expected values are updated as changes unfold.

In future work, we need to address several issues. Firstly, we need to experimentally evaluate our mechanisms. The agents we currently develop in which the mechanisms are embedded shall afford this. Secondly, we need to analyze the effect our mechanism may have once large portions of the buyers in the electronic market implement it. Issues raised in [11] and the methods in which they were analyzed may prove of relevance here. Finally, we need to further enhance our mechanisms to fully address cases where the agent purchases multiple identical items from several sites. We have already analyzed the concepts of these enhancements. The details require additional research.

References

[1] http://www.mysimon.com/

[2] http://www.dealtime.com/

[3] D. Fudenberg and J. Tirole. Game Theory. MIT Press, 1991.

[4] P. Wurman, M. Wellman and W. Walsh. The Michigan Internet AuctionBot: a Configurable Auction Server for Human and Software Agents. Proc. of Autonomous Agents 1998, pp. 301-308.

[5] J. Rodriguez-Aguilar, P. Noriega, C. Sierra and J. Padget. FM96.5: A Java-based Electronic Auction House. In Proc. of PAAM 1997.

[6] P. Maes and A. Chavez. Kasbah: An Agent Marketplace for Buying and Selling Goods. Proc. of PAAM 1996.

[7] C. Preist. Commodity Trading Using an Agent-Based Iterated Double Auction. Proc. of Autonomous Agents 1999, pp. 131-138.

[8] S. Park, E. Durfee and W. Birmingham. An Adaptive Agent Bidding Strategy Based on Stochastic Modeling. Proc of Autonomous Agents 1999, pp. 147-153.

[9] K. Ito, N. Fukuta, T. Shintani and K. Sycara. BiddingBot: A Multiagent Support System for Cooperative Bidding in Multiple Auctions. Proc. of ICMAS 2000, pp. 399-400.

[10] E. Gimenez-Funes, L. Godo, J. Rodríguez-Aguilar, P. Garcia-Calves. Designing Bidding Strategies for Trading Agents in Electronic Auctions. Proc. of ICMAS 1998, pp. 136-143.

[11] A. Greenwald and J. Kephart, Shopbots and Pricebots. Proc. of IJCAI 1999, pp. 506-511.

[12] C. Preist, A. Byde and C. Bartolini. Economic Dynamics of Agents in Multiple Auctions. Proc. of Autonomous Agents 2001, pp. 545-551.

Cryptographic Protocols for Secure Second-Price Auctions

Felix Brandt

Institut für Informatik, Technische Universität München
80290 München, Germany
Tel. +49-89-28928416
brandtf@in.tum.de

Abstract. In recent years auctions have become more and more important in the field of multiagent systems as useful mechanisms for resource allocation, task assignment and last but not least electronic commerce. In many cases the Vickrey (second-price sealed-bid) auction is used as a protocol that prescribes how the individual agents have to interact in order to come to an agreement. The main reasons for choosing the Vickrey auction are the existence of a dominant strategy equilibrium, the low bandwidth and time consumption due to just one round of bidding and the (theoretical) privacy of bids. This paper specifies properties that are needed to ensure the accurate and secret execution of Vickrey auctions and provides a classification of different forms of collusion. We approach the two major security concerns of the Vickrey auction: the vulnerability to a lying auctioneer and the reluctance of bidders to reveal their private valuations. We then propose a novel technique that allows to securely perform second-price auctions. This is achieved using the announcement of encrypted binary bidding lists on a blackboard. Top-down, bottom-up and binary search techniques are used to interactively find the second highest bid step by step without revealing unnecessary information.

1 Introduction

The area of multiagent systems (e.g., [6, 16, 24]), which is concerned with systems composed of technical entities called agents that interact and in some sense can be said to be intelligent and autonomous, has achieved steadily growing interest in the last years. Electronic commerce, automated resource allocation and task assignment are topics that are of major interest in the agent community. As auctions are very flexible and efficient tools that can be used to address these problems, it has become common practice to apply well known results and insights from auction theory (e.g., [14, 15]) and well understood auction protocols like the English auction, the Dutch auction, and the Vickrey auction. Among the different protocols, the Vickrey auction [22] (also known as second-price sealed-bid auction) has received particular attention within the multiagent community (e.g., [9, 5, 23, 4, 3, 2]). due to three main characteristics:

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 154–165, 2001.

- it requires low bandwidth and time consumption
- it possesses a dominant strategy, namely, to bid one's true valuation[1]
- it is a sealed-bid auction; bids (expressing private values) remain secret

These characteristics make the Vickrey auction protocol particularly appealing from the point of view of automation. The Vickrey auction, in its original formulation and as it is used for *selling goods* or *resource allocation*, works as follows: each bidder makes a sealed bid expressing the amount he is willing to pay, and the bidder who submits the highest bid wins the auction; the price to be payed by the winner is equal to the second highest bid. In *task assignment scenarios* the Vickrey auction works exactly the other way round (and for that reason is often referred to as reverse Vickrey auction): each bidder willing to execute a task makes a bid expressing the amount he wants to be payed for task execution, and the bidder submitting the lowest bid wins the auction; the winner receives an amount equaling the second lowest bid (and his payoff thus is the second lowest bid minus his prime costs for execution). If there is more than one winning bid, the winner is picked randomly. As the standard and the reverse auction are fully analogical, all presented considerations and techniques do hold for both formulations of the Vickrey auction.

Despite its impressive theoretical properties, the Vickrey auction is rarely used in practice. This problem has been addressed several times [18, 19, 17] and it is an undisputed fact that the Vickrey auction's sparseness is due to two major reasons. First, the proper execution of a Vickrey auction depends on the truthfulness of the auctioneer. The highest bidder has to trust the auctioneer when he is told the second highest bid. There is a great risk of an insincere auctioneer overstating the second highest bid on purpose to gain more money (either for himself or his client, the seller). Secondly, bidders are usually reluctant to reveal their true private values as is required by the dominant strategy. In many scenarios auctions are not isolated events, but rather parts of a whole series of negotiations. Valuations of goods or tasks are sensitive information that agents intend to keep private. Even if the transfer of the sealed bids is fully protected applying encryption techniques, it remains unknown how the auctioneer treats this confidential information. The privacy of sealed bids is also a substantial topic in regular first-price auctions, but it is of even more importance in second-price auctions since bids represent unaltered private values. This paper addresses both crucial weaknesses of the Vickrey auction. We present a technique that ensures the inability of the auctioneer to manipulate the outcome of the auction and we develop methods that reduce the auctioneer's knowledge of bids to a minimum in order to weaken his position in collusive agreements.

The paper is structured as follows. Section 2 summarizes existing efforts in the field of secure Vickrey auctions and explains why we restrict ourselves to a single auctioneer. Section 3 defines vital attributes that ensure a smooth auction conduction and specifies significant types of collusions. The foundations for the new secure auction protocol are set up in Section 4 and Section 5 introduces and analyzes three methods that hide unneeded information from the auctioneer. Finally, Section 6 concludes the paper with a brief overview of advantages and disadvantages of the proposed technique.

[1] if bidders are risk-neutral and have private valuations of goods or tasks.

2 Related Work

Franklin and Reiter were among the first to address electronic auction security[7]. They covered many basic problems, combined cryptographic primitives such as secret sharing, digital cash and multicasts, and introduced their own primitive called “verifiable signature sharing”.

There are only very few publications devoted to second-price auctions [8, 12, 11] and all of them (as most of the first-price auction papers [21, 1, 13, 10]) rely on the (limited) security of distributed computation. This technique requires m auctioneers, out of which $\lfloor \frac{m-1}{3} \rfloor$ must be trustworthy. Bidders send shares of their bids to each auctioneer. The auctioneers jointly compute the selling price without ever knowing a single bid. This is achieved by using sophisticated, but sometimes inefficient, techniques of secure multiparty function evaluation, mostly via distributed polynomials. However, a collusion of e.g., three out of five auctioneer servers can already exploit the bidders' trust. It would therefore be desirable to remove the trust in the auctioneer(s) entirely which is the main goal of this paper.

3 Properties of a secure Vickrey auction service

We consider a situation where one seller and n bidders or buyers intend to come to an agreement on the selling of a good or task. The auctioneer is an agent that acts as an intermediary between the seller and the bidders and fixes the selling price. In the following, we will set up properties that are required for flawless conductions of second-price sealed-bid auctions. These properties are divided into two categories. The first one deals with the accurate execution of the auction (Figure 1) whereas the second one defines rules that guarantee the privacy of confidential information (Figure 2).

E1 The auctioneer is not capable of determining a false winner.
E2 The auctioneer is not capable of manipulating the selling price upwards.
E3 The auctioneer is not capable of manipulating the selling price downwards.
E4 The highest bidder cannot deny having made the highest bid.
E5 The second highest bidder cannot deny or alter his bid.
E6 The auction process cannot be paralyzed by malicious bidders.

Fig. 1. Auction properties (execution)

Figure 1 and 2 specify the properties of an ideal Vickrey auction. There is yet no protocol that meets all these demands and it seems arguable whether the second highest bid can actually be determined without revealing any information at all. Information that has to be revealed by the auctioneer should be kept to a minimum to prevent him from providing unauthorized persons with these sensitive data.

We will not consider the possibility of a malicious auctioneer, that is an auctioneer that deliberately hampers the auction mechanism. Under the assumption that **E1**-**E3** are

P1 the bids and the corresponding bidders' identities are unknown prior to the opening of the bids.

P2 the bids and the corresponding bidders' identities remain unknown even after the auction process is finished (except the declaration of the second highest bid).

Fig. 2. Auction properties (privacy of information)

fulfilled, this behavior cannot result in an incorrect auction outcome, but only in no result at all. As a consequence, bidders could choose another auctioneer and re-auction the item. Furthermore, we will not approach the problem how to enforce the payment from the winning bidder. See [7] for a discussion of this issue involving electronic money. Besides, we assume that all bids are digitally signed and no bidder can act on the behalf of another.

An interesting question that arises when applying Vickrey auctions is which information has to be declared by the auctioneer after the bid submission and evaluation period is finished: the winner's identity, the selling price or both of the above?

It is inevitable to declare the winning price in a secure Vickrey auction protocol in order to prevent the auctioneer from awarding the contract to a bogus bidder. Each bidder can compare the declared price with his bid and complain if his bid is higher than the winning price and he has not been notified by the auctioneer. The bidder's identity, however, can normally be kept as a secret to the winner and the auctioneer. Yet, this requires that the declared selling price is undoubtedly the actual second highest bid.

To enable analysis of our auction protocol, we will consider every reasonable form of collusive agreements. We distinguish the following types of collusion:

- auctioneer/seller (A/S)
- auctioneer/bidder(s) (A/B)
- bidder/bidder (B/B)

B/B collusion can be seen as the most common type of collusion. As the English auction, the Vickrey auction is in particular vulnerable to B/B collusions, i.e., agents that team up to eliminate rivalry, resulting in lower selling prices.

A classic example of A/S collusion is an auctioneer that overstates the second highest bid to increase the seller's revenue. Another important kind of A/S collusion is represented by an auctioneer that declares no or a non-existent winning bidder due to too low bids.

An often neglected form of collusion is A/B collusion, e.g., an auctioneer that collaborates with the winning bidder and therefore intends to understate the selling price. However, in most real-world scenarios, auctioneers gain a fraction of the selling price and A/B collusions do not seem to be realistic. We therefore consider it as a minor form of collusion.

Collusions involving the auctioneer (A/S and A/B) are of particular interest in secure auction protocols because they allow agents to receive sensitive information from the auctioneer.

4 Publishing encrypted bids

As stated above, a bidder that won an auction cannot be sure whether the price the auctioneer tells him to pay is really the second highest bid. In the general case, he is even incapable of detecting whether the given price is one of the submitted bids, e.g., the auctioneer could make up a high bogus bid to increase the seller's revenue. A completely fake second-highest bid can be prevented by cryptographic signatures. However, this does not hinder a criminal auctioneer from having a bidder agent that submits a strategic bid *after* the auctioneer received and evaluated the other bids.

For this reason, it seems to be a good idea to divide the auction process into two parts. In the initial phase the auctioneer receives encrypted bids from the bidders and publishes the anonymized bids on a blackboard or via a multicast. The bids are encrypted by arbitrary personal keys created by each bidder. The auctioneer is not yet capable of opening the bids (**P1**).

The second phase starts after the submission deadline. As each bidder can observe the encrypted bids on the blackboard, the auctioneer is now unable to (undetectedly) alter existing or add fake bids. In the following, each bidder sends his key (masked with a random number to avoid bidder identification after the auction) to the auctioneer using public-key encryption. After having received the keys, the auctioneer secretly opens all bids, determines the second highest one and publishes the corresponding key. Additionally, he notifies the winning bidder and sets up the transaction.

This procedure voids the auctioneer's ability to change submitted bids or add bids after the submission deadline. Additionally, the auctioneer is incapable of overstating the selling price (**E2**). If the bidders are not collaborating with the auctioneer, they will detect the announcement of understated selling prices (**E3**). For instance, if the auctioneer deliberately declares the third highest bid as the selling price (to support the winning bidder), the second highest bidder observes this and can prove his claim by supplying his key.

However, if this bidder has no incentive to clarify the auction outcome, a collusion of the auctioneer and the two highest (or more) bidders ("A/B/B" collusion) can result in an undetectable understatement of the selling price. Another form of cheating in this protocol may occur when the auctioneer and a single bidder collude. The auctioneer can declare the highest bid as the selling price and his partner as the winner (violation of **E1**). However, the bidder has to pay more than he bid for the good, which he usually will not be willing to do. In an A/S collusion setting, the auctioneer could declare the highest bid as the selling price and secretly cancel the selling of the good, as all bidders assume another bidder won the auction. This can easily be prevented by publicly declaring the winner. All of the above problems (some of them are not relevant due to the rareness of A/B collusion) will be solved in the subsequent section.

As the protocol described is interactive, there is a problem when agents decide to inhibit the auction protocol by supplying no or false keys. We call this the *key denial* problem. It is impracticable to wait for each key forever because of **E6**. As a consequence, the auctioneer has to take measures if bidders are refusing to supply valid keys in time. Unfortunately, the key denial problem cannot simply be solved by withdrawing the undecryptable bids because a B/B collusion (that received private information from the auctioneer) can take advantage of this and understate the selling price. In order to assure a smooth auction process, uncooperative bidders have to be fined. As imposing sanctions for future auctions is not always feasible due to the anonymity in electronic marketplaces, a reasonable fine that allows easy implementation is to assign a *default bid* to incommoding bidders. In a setting where bidders receive confidential information from the auctioneer, there are two reasons for strategically denying a key: avoiding to win the auction due to an already opened second highest bid or manipulating the selling price after the highest bid has been opened. In the majority of auction environments, the default bid should be as high as possible. However, as the denial of a key will always falsify the actual auction outcome, a more rigorous measure is to fine the refusing bidder by voiding his bid and forcing him to pay a penalty. Obviously, this still does not meet the demands specified in **E4** and **E5**, but it can render the refusal of a key uneconomic.

5 Restriction of the auctioneer's knowledge

Given the procedure of the previous section, the auctioneer is unable to alter the outcome of the auction by making up fake bids. What remains to be done is to restrain his ability to release confidential information (bids and their origins) and to prove that the price he declares is actually the second highest bid. We will achieve this by the iterative opening of binary bidding lists. First of all, we discretize the bid interval into k possible bids $\{p_1, p_2 \dots p_k\}$. Each bidder submits a bidding list that consists of k binary values denoting whether he is willing to pay a given price or not. Every bit in this list is encrypted using a different, arbitrary key K_{ij} generated by the bidder. These bidding lists are put together to the so-called *bid matrix* (see Table 1) and are published on a blackboard like in the previous section. As usual the functions $e(b, K)$ and $d(b, K)$ encode and decode a bid b with key K, respectively. To ensure anonymity of revealed bids the columns on the blackboard are presented in random order. The goal is to find an opening sequence that rapidly locates the second highest bid by revealing as little information as possible. Applying pre-defined sequence rules, the auctioneer requests key by key from the bidders until the second highest bid is undoubtedly detected.

	Bidder 1	Bidder 2	...	Bidder n
p_k	$e(b_{1k}, K_{1k})$	$e(b_{2k}, K_{2k})$	...	$e(b_{nk}, K_{nk})$
p_{k-1}	$e(b_{1,k-1}, K_{1,k-1})$	$e(b_{2,k-1}, K_{2,k-1})$	...	$e(b_{n,k-1}, K_{n,k-1})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
p_1	$e(b_{11}, K_{11})$	$e(b_{21}, K_{21})$	...	$e(b_{n1}, K_{n1})$

Table 1. Bid matrix

After having found the second highest bid p_y at position (x, y), the auctioneer publishes the *essential keys* E defined by the following equation.

$$E \quad = \quad \{K_{xy}\} \cup \{K_{i,\min(y+1,k)} \mid i \in \{1, 2 \dots n\}\}$$

Figure 3 shows an example bid matrix ($n = 10, k = 15, p_1 = 5, p_2 = 10 \dots p_{15} = 75$) and the keys to be published after the auction.

This proves to all participants that the declared selling price is actually the second highest bid. As a consequence, conditions **E2** and **E3** are fulfilled. Of course, keys for negative bids lower than p_{y+1} and for positive bids higher than p_{y+1} can be used as well, but this would give the bidders more information than necessary. It has to be decided as the case arises if requesting some additional keys can be afforded. As the columns of the bid matrix are shuffled, the bidders cannot relate bids and bidders.

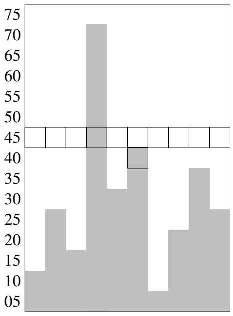


Fig. 3. Essential keys

In the following sections, we propose three different search procedures that locate and return the second highest bid. Each of them limits the auctioneer's knowledge of bids and their origins, resulting in a partial validity of **P2**.

Figure 4 formally summarizes the communication framework for the three procedures.

- ○ PHASE 1: Each bidder i supplies an ordered list of k encrypted bids (each one encrypted with an arbitrary key K_{ij}).
  — *Bid submission deadline* —
- – The auctioneer publishes the bidding lists on a blackboard (in random order).
- ○ PHASE 2: The following steps are repeated until the auctioneer has detected the second highest bid and (if desired) until he has received all essential keys (E).
  1. The auctioneer secretly[1] demands key K_{ij} from the corresponding bidder i (i and j are yielded by one of the algorithms in the subsequent sections).
  2. Bidder i sends K_{ij} and a random number encrypted with the auctioneer's public key.
     — *Key submission deadline* —
  3. The auctioneer verifies the key by attempting to decrypt b_{ij}. If the key is invalid or the bidder sent no key at all, a default bid is used and (if necessary) bidder i is fined.
- – The auctioneer publishes the keys in E (or another set of keys that proves the location of the second highest bid and the column of the winning bidder).
- – The seller and the winning bidder (who can identify himself by supplying the seller with all remaining keys of his column) get in contact and initiate the transaction.

[1] e.g., by using public key encryption.

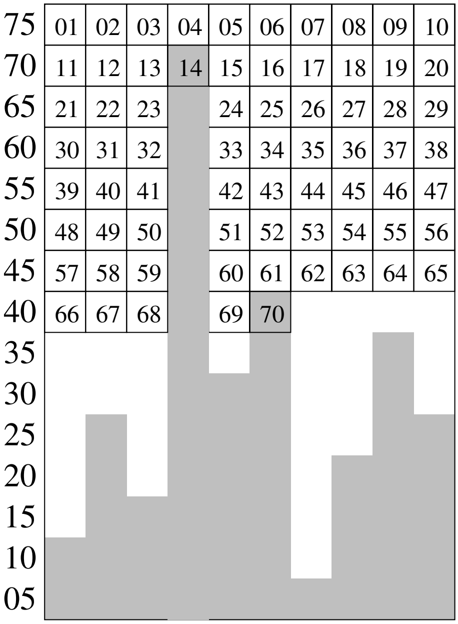
Fig. 4. Communication protocol

5.1 Downward bid search (dbs)

A straight-forward method to open the bids is to start at the highest price and open each row of bids downwards until at least two bidders are willing to pay a given price. This is similar to a *second-price* Dutch (descending) auction [22]. The following algorithm fulfills this task. To save space, we use $e_{ij} = e(b_{ij}, K_{ij})$ as an abbreviation for the encrypted bids and "$d(e_{ij}, K_{ij}) =$ true" if a bid is positive. The algorithm is decomposed into two separate procedures (dbs and dbs2) because we will reuse the second procedure for another search technique in a subsequent section. Decrypted bids are denoted by numbered frames in the example bid matrix, thus illustrating the opening sequence. The search begins in the upper left corner of the bid matrix by evaluating dbs(1,k).

```
procedure int dbs($i$, $j$)
  while $j > 0$ do
    for $n$ times do
      if $d(e_{ij}, K_{ij}) =$ true then
        return dbs2($i$, $j$, $\{i\}$)
      end if
      $i = i + 1$
      if $i > n$ then $i = 1$ endif
    end for
    $j = j - 1$
  end while

procedure int dbs2($i$, $j$, $F$)
  while $j > 0$ do
    for $n$ times do
      if $i \notin F \wedge d(e_{ij}, K_{ij}) =$ true then
        return $j$
      end if
      $i = i + 1$
      if $i > n$ then $i = 1$ endif
    end for
    $j = j - 1$
  end while
```



The number of bids to open is $O(nk)$. After the decryption process, the auctioneer knows just two out of n bids (the highest and the mandatory second highest) and has no clue concerning the other bids. Furthermore, this search procedure has the advantage that the auctioneer is not capable of requesting more keys than he is allowed to (those beneath the declared selling price) without being detected afterwards. Please note that only the keys marked in Figure 3 are presented to the bidders.

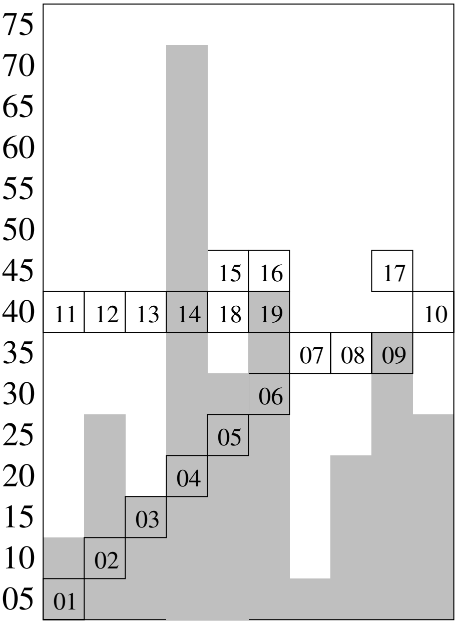
Although, revealing only one private value may seem a fairly good result, a disadvantage of this procedure is that unfortunately the highest bid usually requires the highest secrecy of all bids.

Bids that cannot be resolved due to denied keys should not be treated as default binary bids (neither negative nor positive) as this would enable easy price manipulation. Assigning a penalty of p_j for an undecryptable bid b_{ij} seems to be an appropriate measure. Bidders cannot take advantage of submitting *inconsistent lists*, i.e., lists that do not represent a single bidding value as only the first occurrence of a positive bid counts.

5.2 Upward bid search (ubs)

The following algorithm avoids the revelation of the highest bid by opening low bids first. When searching upwards, the auctioneer can skip to the next higher row when at least one bidder is willing to pay at a given price. The skipping of a line must not be triggered by the same bidder for two times consecutively. This technique resembles an ascending auction, in particular an English auction. The search starts in the lower left corner of the bid matrix (`ubs(1,1)`).

```
procedure int ubs($i, j$)
  $F = \emptyset$
  while $j \leq k$ do
    $p = 0$
    $F' = \emptyset$
    for $n - 1$ times do
      if $i \notin F$ then
        if $d(e_{ij}, K_{ij}) =$ true then $p = 1$
        else $F' = F' \cup i$ endif
      end if
      $i = i + 1$
      if $i > n$ then $i = 1$ endif
      if $p = 1$ then break endif
    end for
    if $p = 0$ then break endif
    $j = j + 1$
    $i' = i$
    $F = F \cup F'$
  end while
  $i = i'$
  for $n - 1$ times do
    if $i \notin F \wedge d(e_{i,j-1}, K_{i,j-1}) =$ true then
      return $j - 1$
    end if
    $i = i + 1$
    if $i > n$ then $i = 1$ endif
  end for
  return $j - 2$
```



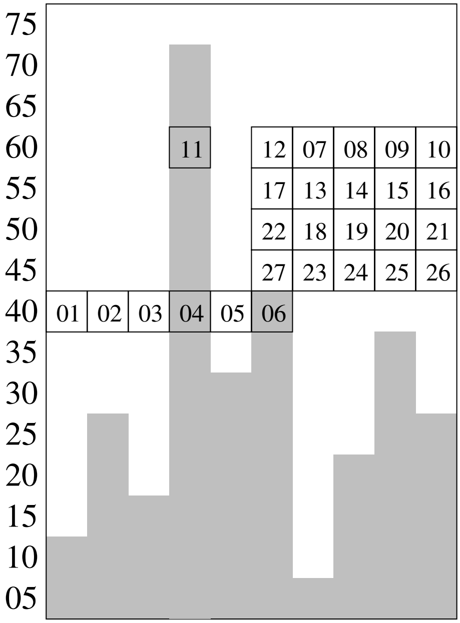
This algorithm is significantly faster than dbs ($O(\max(k, 2n))$). The auctioneer learns partial information about the losing bids and no information at all about the highest bid. This guarantees that the winning bid will remain secret even after the auction terminated. However, this type of bid searching may be unsuitable when $k \gg n$ because it reveals too much information about losing bids. The lowest bid can be determined to be in an interval of at most n bids. As bidders drop out when the price rises, the higher bids can be specified more precisely. The third highest bid is barely hidden from the auctioneer after ubs has been executed. It has to be one out of two possible values.

The key denial problem can be addressed more generously when searching upwards. It is sufficient for almost all scenarios to assess the value of denied bids as negative. It should be noted that due to the randomized order of the columns in the bid matrix, it is fair to open bids in linear order.

5.3 Binary bid search (bbs)

Related to well-known standard binary search, bbs begins in the middle of an interval by opening consecutive bids. After two positive bids have been found, the row is finished and bbs is called recursively for the upper half of the interval. If, after having opened all bids in a row, none of them is positive, the search is continued recursively in the lower half. If exactly one positive bid is found, dbs2 is called from this point. dbs2 reveals no additional information, except the required second highest bid. The search is initiated by executing bbs(1,1,k,$\emptyset$).

```
procedure int bbs($i, a, b, F$)
  $j = a + \lfloor \frac{b-a}{2} \rfloor$
  $p = 0$
  $F' = \emptyset$
  for $n$ times do
    if $i \notin F$ then
      if $d(e_{ij}, K_{ij})$ = true then $p = p + 1$
      else $F' = F' \cup i$ endif
    end if
    if $p = 2$ then break endif
    $i = i + 1$
    if $i > n$ then $i = 1$ endif
  end for
  if $p = 2$ then return bbs($i, j, b, F \cup F'$) endif
  if $p = 0$ then return bbs($i, a, j, F$)
  else return dbs2($i, j + 1, F \cup i$) endif
```



This method seems to be a good compromise between both previous techniques. Partial information about some bids is revealed, but this information is by far not as precise as the one revealed by ubs. Because this algorithm uses dbs2 to determine the second highest bid, it has the same round complexity as the downward bid search. Applying the binary search until the end would reduce the number of opened bids to $O(n \log_2(k))$, but this could reveal more information than needed. The search time can be further decreased by starting at the expected value of the second highest bid instead of the middle of the bid interval.

bbs is somewhat similar to the opening of bits in a standard binary representation of integers, but it has the advantage of uncovering less information.

It would be possible to execute this algorithm as an open-cry interactive auction. The auctioneer starts by asking the bidders in random order if they are willing to pay the price in the middle of the bidding interval. When two bidders accept the given price, he recursively performs this operation for the upper half of the interval. If all bidders reject the price, he continues with the lower half. In comparison with an open-exit English auction, this auction would have the advantages of lesser information revelation and faster execution time. However, the downward search (after the winner has been determined) had to be omitted as pricing could easily be manipulated by bidder collusions.

6 Conclusion

We presented a protocol that can be used in various ways to realize secure second-price auctions with a single auctioneer. Using a blackboard that displays encrypted binary bids, the auctioneer cannot alter the auction outcome without being detected. The needed trust in the auctioneer not to reveal private information can be vastly reduced by three search methods that restrict the auctioneer's knowledge in different ways. In fact, the auctioneer is reduced to a mediator that has no means to influence the result of an auction. It is even possible to completely omit the auctioneer and implement the suggested protocols in an open fashion that allows the bidders to resolve the auction on their own. This would equate all bidders since no-one could benefit from secret information received from the auctioneer. On the other hand, an auctioneer can be useful to *try* to conceal the sensitive data, at least. The bid opening protocol is interactive, but as it prohibits strategic interaction, agents are urged to supply all requested keys. Our technique meets the demands specified in **E1**, **E2**, **E3**, **E6** and **P1**. Practical compliance with **E4** and **E5** can be obtained by imposing fines to uncooperative bidders. `dbs`, `ubs` and `bbs` are three different ways to ensure the partial validity of **P2**.

Thinking of open networks, it would be necessary to carry out camouflage communication between the auctioneer and the bidders to hinder a bidder from drawing conclusions from the sequence of key requests. A drawback of our protocol obviously lies in its interactivity. The price determination might require lengthy communication between the auctioneer and bidders. However, in many real-world scenarios, the utmost secrecy of bids is more important than a rapid execution time.

Obviously, the discretization of the bid interval represents a limitation, but as bid values do not have to be equidistant, arbitrary bid sets, e.g., logarithmic scales can be used as well and enable an efficient partitioning of the bid interval.

In contrast to most existing second-price auction protocols, the proposed protocols do not have difficulties in detecting ties, i.e., more than one equal maximum bids.

The three suggested search techniques clearly illustrate the equivalence of Vickrey, second-price Dutch and English auctions for private value bidders. In addition, the binary search procedure demonstrates a novel method to locate the second highest bid. In the future, we intend to further investigate this new type of secure sealed-bid auction realization, implement, and evaluate the proposed procedures.

References

1. O. Baudron and J. Stern. Non-interactive private auctions. In *Pre-Proceedings of Financial Cryptography 2001*, pages 300–313, 2001.
2. F. Brandt, W. Brauer, and G. Weiß. Task assignment in multiagent systems based on Vickrey-type auctioning and leveled commitment contracting. In M. Klusch and L. Kerschberg, editors, *Cooperative Information Agents IV*, volume 1860 of *Lecture Notes in Artificial Intelligence*, pages 95–106, Berlin et al., 2000. Springer-Verlag.

3. F. Brandt and G. Weiß. Vicious strategies for Vickrey auctions. In *Proceedings of the 5th International Conference on Autonomus Agents*, pages 71–72. ACM Press, 2001.
4. K. Danielsen and M. Weiss. User control modes and IP allocation. http://www.press.umich.edu/jep/works/DanieContr.html, 1995. presented at MIT Workshop on Internet Economics.
5. K.E. Drexler and M.S. Miller. Incentive engineering for computational resource management. In B.A. Huberman, editor, *The Ecology of Computation*. The Netherlands, 1988.
6. J. Ferber. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. John Wiley & Sons Inc., New York, 1999.
7. M.K. Franklin and M.K. Reiter. The design and implementation of a secure auction service. *IEEE Trans. on Software Engineering*, 22(5):302–312, 1996.
8. M. Harkavy, J.D. Tygar, and H. Kikuchi. Electronic auctions with private bids. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pages 61–74, 1998.
9. B. Huberman and S.H. Clearwater. A multiagent system for controlling building environments. In *Proceedings of the 1st International Conference on Multiagent Systems (ICMAS-95)*, pages 171–176, Menlo Park, CA, 1995. AAAI Press.
10. M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. In *Proceedings of Asiacrypt-00*, pages 162–177, 2000.
11. H. Kikuchi. (M+1)st-price auction protocol. In *Proceedings of Financial Cryptography (FC 2001)*, 2001.
12. H. Kikuchi, S. Hotta, K. Abe, and S. Nakanishi. Resolving winner and winning bid without revealing privacy of bids. In *Proceedings of the International Workshop on Next Generation Internet (NGITA)*, pages 307–312, 2000.
13. M. Kudo. Secure electronic sealed-bid auction protocol with public key cryptography. *IEICE Trans. Fundamentals*, E81-A(1), 1998.
14. R.P. McAfee and J. McMillan. Auctions and Bidding. *Journal of Economic Literature*, 25:699–738, 1987.
15. P.R. Milgrom and R.J. Weber. A Theory of Auctions and Competitive Bidding. *Econometrica*, 50:1089–1122, 1982.
16. G.M.P. O'Hare and N.R. Jennings, editors. *Foundations of Distributed Artificial Intelligence*. John Wiley & Sons Inc., New York, 1996.
17. M.H. Rothkopf and R.M. Harstad. Two models of bid-taker cheating in Vickrey auctions. *Journal of Business*, 68(2):257–267, 1995.
18. M.H. Rothkopf, T.J. Teisberg, and E.P. Kahn. Why are Vickrey auctions rare? *Journal of Political Economy*, 98(1):94–109, 1990.
19. T.W. Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Proceedings of the 2nd International Conference on Multiagent Systems (ICMAS-96)*, Menlo Park, CA, 1996. AAAI Press.
20. A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.
21. D.X. Song and J.K. Millen. Secure auctions in a publish/subscribe system. Available at http://www.csl.sri.com/users/millen/, 2000.
22. W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.
23. C.A. Waldspurger, T. Hogg, B. Huberman, J.O. Kephart, and W.S. Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992.
24. G. Weiß, editor. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, MA, 1999.
25. A.C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, 1986.

Equilibria Strategies for Selecting Sellers and Satisfying Buyers*

Claudia V. Goldman[1], Sarit Kraus[1,2], and Onn Shehory[3]

[1] Department of Mathematics and Computer Science
Bar-Ilan University, Ramat-Gan 52900,Israel
{clag,sarit}@cs.biu.ac.il
[2] Institute for Advanced Computer Studies, University of Maryland
College Park,MD 20742
[3] IBM Research Lab in Haifa - The Tel-Aviv Site onn@il.ibm.com

Abstract. Dynamism of trade activity inevitably results in situations where sellers face local supply shortages. In such cases, sellers need to decide which buyer purchase requests to satisfy. Commonly, sellers satisfy purchase requests based on their arrival order, i.e., First In is First Served (FIFS). In electronic trade, sellers may follow strategies different from FIFS without the buyers being able to detect this difference. Buyers, in response to the sellers' strategic behavior, may themselves adopt strategies that will maximize their utility. Previous research has suggested strategies to be used by electronic seller-agents and buyer-agents. Yet, that research examined markets in which buyers are willing to accept partial satisfaction of their request and sellers' stocks are all the same. A simulation tool was developed under such conditions. This paper utilizes the simulation tool to explore equilibria in more realistic markets, where sellers' stocks are heterogeneous, and buyers suffer significant losses from partial satisfaction of their requests.

1 Introduction

Strategic behavior is inseparable from trade activity, and was implemented by traders for thousands of years. Electronic markets do not change this situation, as strategies can be devised and used to increase the gains of electronic trading parties too. However, in difference from traditional trade, electronic markets introduce a combination of conditions that does not exist in other markets. As a result, well-known strategies may become obsolete, and new, beneficial strategies will arise. This gives ground to research of the type presented in this paper.

Here, we examine scenarios where electronic sellers dynamically receive multiple purchase-orders. At times, the cumulative demand at a specific seller may surpass its stock level. Thus, the seller will not be able to address all orders, however, it would like to maximize gains nevertheless. We study strategies that allow this maximization. In business-to-consumer physical trade, sellers commonly satisfy purchase requests based on their arrival order, i.e., First In is First Served

* This research is supported in part by NSF under grant No. IIS9820657 and by IBM.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 166–177, 2001.

(FIFS). The FIFS strategy is considered fair, thus acceptable by buyers, and therefore sellers that follow it are not exposed to punishing buyer reactions. In electronic trade, sellers may follow strategies different from FIFS without the buyers being able to detect this difference. Such deviation from FIFS may increase the gains of the seller, due to preference of more profitable deals. Buyers, in response to the sellers' strategic behavior, may themselves adopt strategies that will maximize the chance of their orders being satisfied. Previous research [1] has suggested sets of strategies which are likely to be used by electronic seller-agents and buyer-agents in such trade conditions. Yet, that research examined markets in which buyers are willing to accept partial satisfaction of their request and sellers' stocks are all the same. While such conditions are excellent for developing a simulation tool (due to their simplicity), they do not hold in real markets. This paper utilizes the simulation tool of [1] to explore equilibria in more realistic markets, where sellers' stocks are heterogeneous, and buyers suffer significant losses from partial satisfaction of their requests. In particular, given these conditions, we study equilibria strategies that allow sellers to maximize gains and buyers to select sellers given the sellers' purchase-order satisfaction.

Our problem domain refers to multiple, possibly anonymous, buyers that electronically approach multiple automated sellers' sites. Buyers have no information regarding other buyers who visit the same sites, nor do they know in advance what is the stock level held by the sellers they approach. Sellers in our domain are uncertain regarding the number of buyers that will approach their site and the volume of their purchase-orders. The dynamism of electronic trade intensifies this problem. Given these settings, we devise strategies for sellers and counter-strategies for buyers, such that sellers maximize gains, buyers maximize satisfaction, and the market is in equilibrium. Due to the complexity of the problem, solving it analytically is practically infeasible, hence we opted for a simulation-based solution. The details are presented in the subsequent sections.

2 The SEMI System

The Simulation of Electronic Market Interactions (SEMI) system supports simulation of repeated encounters between sets of automated buyers and sellers. Although SEMI can be used for a broad range of settings, we use it to simulate sellers that hold limited stocks and buyers that place a purchase order with one seller at each encounter. In this study we are mainly concerned with markets in which, at each encounter, a seller may receive requests for goods that cumulatively exceed the amount available in its stock. Therefore, it needs to decide which of the requests to satisfy and which buyers to leave unsatisfied. Electronic sellers that face such decision problems should be equipped with strategies for making such decisions, or means for locating such strategies. Preferably, strategies followed by market players (sellers and buyers) should maximize gains and be in equilibrium. Unfortunately, analytical computation of such strategies is too complex for the market settings we study. Running simulations is important when analytical solutions are difficult to obtain. Experiments with SEMI can (and do) unravel equilibria between the buyers and sellers' strategies. These equilibria are the basis for recommendations for strategy design principles for automated buyers and sellers. SEMI was developed as part of an earlier study [1].

A SEMI simulation consists of a sequence of K encounters between sellers and buyers from the finite sets $\mathcal{S}$ and $\mathcal{B}$ respectively. In each encounter, k, a seller $S^j \in \mathcal{S}$ holds a stock of goods ST^{jk} for sale. Each buyer $B^i \in \mathcal{B}$ is associated with a type TY^i and a purchase-order PO^{ik}. TY^i indicates the average purchase-order size of B^i. B^i submits its purchase-order PO^{ik} to one of the sellers, S^j. S^j may sell B^i at most the amount specified in PO^{ik}. The utility of B^i increases in proportion to the portion of PO^{ik} supplied by S^j. A seller's utility increases with the cumulative amount that it sells to the buyers, and decreases with the amount of goods that is left in its stock at the end of the encounter.

We distinguish between two kinds of buyers: a *recognizable* and an *unrecognizable buyer*. A seller can use the information it obtained about the type of a recognizable buyer in future encounters. Information about the type of an unrecognizable buyer can only be used in the encounter in which it was obtained.

Here, we analyze the sellers and buyers agents' behaviors in three scenarios implemented with SEMI. All the sellers sell the same type of a good, where quality, price and units are the same as well. Thus, PO^{ik} specifies the number of whole units of the good that B^i would like to buy in encounter k. Even though sellers and buyers transact repeatedly, they do so for a rather short period, during which it is reasonable to assume that the price of a sold unit remains static.[1] We also assume that the prices are uniform; we do not handle auctions but transactions performed in fixed-price, catalogue-based, electronic stores. We further assume that each buyer is associated with only one purchase-order and each purchase-order is valid for only one encounter. That is, when a buyer B^i approaches a seller S^j with PO^{ik}, S^j can at most supply PO^{ik} in encounter k. It cannot delay the supply, and PO^{ik} cannot be supplied by another seller at any encounter. After B^i approached S^j at encounter k, we denote its order PO^{ik}_j.

In all of the experiments performed, a seller gains from the sale of each unit and it incurs a cost for each unit it holds unsold in its stock at the end of each encounter. In cases where a seller reveals buyers' types, it has an additional cost for doing this. The overall utility of a seller is the sum of its utility from each encounter. Formally, the utility of S^j in a given encounter k, when it receives r purchase-orders, is $U^j(PO^{1k}_j, \ldots, PO^{rk}_j)$ as described in table 1.

In the basic scenario, we assume that the size of the stock of all sellers at the beginning of each encounter is the same. A buyer B^i's utility is $U^i(PO^{ik})$ as presented in table 1. In the second scenario, we study non-conceding buyers. The utility from partial satisfaction of purchase-orders has a greater impact on non-conceding buyers. A non-conceding buyer B^i's utility is given by the function U^i_{nc} as described in table 1, where LS is the factor that reflects the level of buyer dissatisfaction with partial purchase-order fulfillment. We say that buyers who implement such utility function are non-conceding, since their willingness to accept partial purchase-order fulfillment is lesser than it was in the basic scenario.

[1] See reference to work by Kephart as explained in Section 6.

Note that here, the utility of a partially satisfied buyer may be nullified.[2] As the value of LS increases, the buyer's level of non-concession increases too.

Table 1. Summary of the SEMI's Notations

Notation	Description
k	The k^{th} encounter.
K	The number of encounters.
$\mathcal{B}= \{B^i\}, 1 \leq i < \infty$	The set of buyers.
$\mathcal{S}= \{S^j\}, 1 \leq j < \infty$	The set of sellers.
ST^{jk}	The stock that seller S^j holds at the beginning of encounter k.
PO_j^{ik}	The purchase-order placed by B^i to S^j at encounter k.
$sat(PO_j^{ik})$	The actual deal satisfied by S^j after B^i requested PO_j^{ik}.
C_s	The cost of holding one unsold item in stock for one iteration.
C_t	The price that a seller needs to pay to reveal one buyer's type.
G	The gain of a seller from selling one unit of good.
r^*	The number of times the seller bought information about the buyers' types in the given encounter.
$U^j(PO_j^{1k}, \ldots, PO_j^{rk})$	$U^j(PO_j^{1k}, \ldots, PO_j^{rk}) = G \cdot \sum_{i=1}^{r} sat(PO_j^{ik}) - C_s \cdot (ST^{jk} - \sum_{i=1}^{r} sat(PO_j^{ik})) - C_t \cdot r^*$
$U^i(PO^{ik})$	$U^i(PO^{ik}) = sat(PO^{ik})/PO^{ik}$
$U^i_{nc}(PO^{ik})$	$U^i_{nc}(PO^{ik}) = (sat(PO_j^{ik}) - (PO_j^{ik} - sat(PO_j^{ik})) * LS)/PO_j^{ik}$

In the third scenario, we consider heterogeneous markets (i.e., markets with different stocks' sizes). The buyers' utility is given by $U^i(PO^{ik})$ as in the basic setting. The sellers may hold stocks with different sizes.

Since we cannot provide a full set of strategies because many variations of strategies exist, we consider representative sets of strategies for the agents.

2.1 The Sellers Agents' Strategies (Σ^s)

A seller's strategy specifies which portion of each purchase-order received to supply. The decision that stems from such a strategy could be based on: (i) the arrival time of the purchase-order (within the encounter); (ii) the size of the purchase-order; and (iii) the type of the buyer that submitted the purchase-order, when these buyers are recognizable.

We categorize the strategies according to the way in which they use the available information. For simplicity, a strategy does not use historical information. This is reasonable in situations where the buyers are unrecognizable. Note that, when the requests to a seller are not cumulatively larger than its stock, the differences between the strategies become unimportant, as the seller agent will supply all the requests, regardless of the specific strategy.

[2] For example, when $LS = 0.5$, a buyer for which less than a third of the purchase-order was satisfied will gain zero, which is equal to not being satisfied at all.

Uninformed seller: The seller does not consider the size of the purchase-orders nor the buyers' types (issues (ii) and (iii)).

1. RandS (Random Seller) — A seller S^j chooses randomly which purchase-order PO^{ik}_j to fulfill constrained by its stock. Assuming that the order of arrival of the buyers to the sellers does not depend on any characteristic of the buyers, this behavior is equivalent to a FIFS behavior.

Greedy Seller: This seller uses the size of the purchase-orders (ii).

1. OPO (Ordered Purchase-Orders) — The order of fulfillment of the buyers' purchase-orders is in decreasing order of their size.
2. DPO (Distributed Purchase-Orders) — A seller satisfies the purchase-orders proportionally to their size. If S^j receives r purchase-orders in encounter k, then each buyer B^i will be supplied with $\lfloor (PO^{ik}_j / \sum_{l=1}^{r} PO^{lk}) \cdot ST^j \rfloor$. The remainder of this distribution is allocated to one buyer, selected randomly.

Intelligent seller: This seller uses the buyers' types in its decision (iii).

1. ORType or OType (Ordered [Recognized] Types) — The order of fulfillment of the buyers' purchase-orders is in decreasing order of the type of the buyers. R in the prefix denotes recognizable buyers. When the buyers are unrecognizable (no R), the seller needs to pay for the buyer's type each time it would like to use it.
2. DRType or DType (Distributed [Recognized] Types) — A seller satisfies the purchase-orders proportionally to the buyers' types. That is, if S^j receives r purchase-orders in encounter k, then for each buyer B^i, the seller computes $po_i = \lfloor (TY^i / \sum_{l=1}^{r} TY^l) \cdot ST^j \rfloor$. If $po_i \geq PO^{ik}_j$ then po_i is set to PO^{ik}_j. The remainder is allocated as in DPO. The prefix R settings of ORType and OType hold here as well.

We hypothesize that a seller will benefit from satisfying highly typed buyers so that the latter return to them, and increase their gain. Moreover, we hypothesize that if a seller assumes that an unsatisfied buyer will limit the purchase-orders that it sends to a seller which have disappointed it, the seller should try to satisfy these buyers as much as it can. Even when the type is unknown, sellers can use the size of the buyer's current purchase-order to estimate its type.

2.2 The Buyers Agents' Strategies (Σ^b)

In the market set in our paper, at each encounter, a buyer only knows what portion of its purchase-order was satisfied. Given such a history of past encounters, a buyer should decide which seller to approach in the current encounter. In this paper, for simplicity, we focus on strategies that take into consideration only the last encounter.[3]

[3] Long histories are more commonly taken into account in long-term interactions between buyers and sellers. In these cases, contracts are usually signed when the supply and the demand are known in advance. Here, we focus on short-term interactions.

1. RandB (Random Buyer) — The buyer chooses a seller randomly.
2. Loyal — If seller S^j has satisfied buyer B^i *completely* or *partially* at encounter k, then B^i returns to seller S^j at encounter $k+1$. If S^j has supplied B^i nothing at encounter k, then at encounter $k+1$ B^i approaches a seller from $\mathcal{S}$ chosen randomly (it may approach S^j again).
3. LoyalP (Loyal and Punish) — If seller S^j has satisfied buyer B^i *completely* or *partially* at encounter k, B^i returns to seller S^j at encounter $k+1$. If S^j has not supplied B^i anything at encounter k, then at encounter $k+1$ B^i approaches a seller from $\mathcal{S} \setminus \{S^j\}$ chosen randomly.
4. LoyalW (Loyal Weak) — If seller S^j has satisfied buyer B^i *completely* at encounter k, then B^i returns to seller S^j at encounter $k+1$. Otherwise, B^i randomly chooses from among the other sellers ($\mathcal{S} \setminus \{S^j\}$).
5. Prob (Probabilistic Buyer) — If buyer B^i has approached seller S^j at encounter k with a purchase order PO_j^{ik}, then B^i will approach S^j at encounter $k+1$, with a probability of $sat(PO_j^{ik})/PO_j^{ik}$. This probability expresses actual satisfaction of a buyer with respect to its purchase order.

The buyers will always return to a seller that has completely satisfied them. These strategies differ when the buyers are partially satisfied or not satisfied at all. A buyer *punishes* a seller it has approached at encounter k by not returning to this seller at encounter $k+1$. LoyalW is the most punishing strategy. The probabilistic strategy is less punishing than LoyalW since a buyer may return to a partial satisfying seller with a positive probability.

Our hypothesis was that not returning to a partially satisfying seller will be the best strategy for the buyer. However, our results do not always support this.

3 Evaluation Criteria

The utility of each agent participating in an E-market is strongly influenced by the decisions taken by each one of the agents. Since in E-markets as those that are studied here, agents are self-interested and competitive, there is no single evaluation criterion that fully captures the preferences of all the market participants. Some solutions might be optimal for part of the agents, but disadvantageous for others. Our study seeks solutions, in which all of the agents get the maximal benefit given the other agents' behavior, and therefore they will have no incentive to deviate from these solutions.

We assume 2 sets of strategies, Σ^s and Σ^b, are given. A strategy profile F is a sequence of strategies, one for each buyer and one for each seller from the relevant sets of strategies. Given a profile F we will use SEMI to compute the average utility of each agent and use it as an estimation of its expected utility. Then, we look for combinations of strategy profiles for the sellers and buyers agents that are in *experimental equilibrium* [1]:

Definition 1 (Experimental Equilibrium). *We say that a profile F is an experimental-equilibrium if, for any agent A who deviates from its strategy in*

F by using another strategy from Σ, A does not increase its estimated expected utility, given that the other agents follow their strategies in F.[4]

Definition 2 (Dominant Experimental Equilibrium). *We say that a profile F is a dominant-experimental-equilibrium if, for any other profile F' that is an experimental-equilibrium, both the sellers and the buyers obtain the largest expected utility by following F.*

We define the concept of level of liquidity of the buyers to analyze the results.

Definition 3 (Level of Liquidity of the Buyers). *The level of liquidity of the buyers is defined by the average number of times that a buyer changes a seller during one SEMI encounter. Formally, it is given by $\frac{\Sigma_{|Sim|,K,|\mathcal{B}|} SellerChanges}{|Sim| \cdot K \cdot |\mathcal{B}|}$, where $|Sim|$ denotes the number of simulations run, and SellerChanges are the number of times that one buyer has changed a seller during one encounter.*

We hypothesize that a lower level of liquidity of the buyers results in a higher expected utility of both buyers and sellers, as long as the buyers change sellers at least once. Our results support this hypothesis.

4 Experiments with the Basic SEMI System

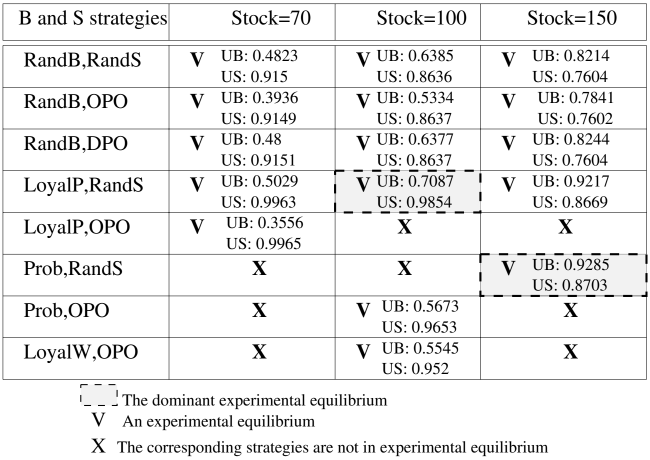
The experiments performed in the basic setting were extensively reported in our previous work [1]. Here, we summarize these findings to compare them with the newer results found in Section 5. We have chosen to report on experiments run on a market that consists of 9 buyers and 3 sellers. Although we have examined other market sizes, this size appears to be the smallest in which meaningful many-to-many buyer-seller interactions occur. The number of sellers must be at least 3 so that a buyer will have the opportunity to select between at least two sellers, even in the case it decided not to return to one of the sellers. The number of buyers should be at least 3 times the number of sellers, to allow an average of at least 3 buyers per seller, so that a seller can choose from among them. The settings of the parameters used for the basic experiments are presented in Table 2. Note that two agents of different types may have the same purchase-order in a given encounter. Thus, the type of a buyer cannot be determined from its purchase-order (though it can be learned from multiple orders). We tested the behaviors of the agents for three values of stocks. The basic case was set so that some purchase-orders are always satisfied and that sellers are not always able to sell their entire stock. The results presented below are averages over 100,000 runs. In each run we used the same parameter settings but randomly chose purchase-orders for each buyer in each of the 30 encounters.

[4] Since our market is complicated, we assume that the set of strategies that is considered for deviation is determined in advance. Note that this definition differs from Nash equilibrium since we use an estimation of the expected utility, rather than the actual value.

Table 2. Basic Settings

$\lvert\mathcal{B}\rvert = 9$, $\lvert\mathcal{S}\rvert = 3$
$G = 1$, a seller's gain from selling one unit.
Types were normally distributed with $\mu = 50$ and $\sigma = 40$.
$PO_j^{ik} = \{TY^i - 1, TY^i, TY^i + 1\}$
for any k *and* $S^j, S^{j'} \in \mathcal{S}$, $ST^{jk} = ST^{j'k}$
$C_s = 2$, a seller's cost for holding an unsold item.
$ST^{jk} = (\mu - X_\sigma \cdot \sigma) \cdot \lvert\mathcal{B}\rvert / \lvert\mathcal{S}\rvert$, X_σ constrains the stock to be smaller than the average, $\lvert\mathcal{B}\rvert/\lvert\mathcal{S}\rvert$ is the normalization factor.

We report on all the experimental equilibria profiles found for stocks of 70, 100 and 150 units.[5] The results appear in Figure 1, and their interpretation follows. In the Figure, the numbers stand for the average expected utility the buyers (UB) and the sellers (US) obtain when following the corresponding strategy.



B and S strategies	Stock=70	Stock=100	Stock=150
RandB,RandS	V UB: 0.4823 US: 0.915	V UB: 0.6385 US: 0.8636	V UB: 0.8214 US: 0.7604
RandB,OPO	V UB: 0.3936 US: 0.9149	V UB: 0.5334 US: 0.8637	V UB: 0.7841 US: 0.7602
RandB,DPO	V UB: 0.48 US: 0.9151	V UB: 0.6377 US: 0.8637	V UB: 0.8244 US: 0.7604
LoyalP,RandS	V UB: 0.5029 US: 0.9963	⬚ V UB: 0.7087 US: 0.9854	V UB: 0.9217 US: 0.8669
LoyalP,OPO	V UB: 0.3556 US: 0.9965	X	X
Prob,RandS	X	X	⬚ V UB: 0.9285 US: 0.8703
Prob,OPO	X	V UB: 0.5673 US: 0.9653	X
LoyalW,OPO	X	V UB: 0.5545 US: 0.952	X

⬚ The dominant experimental equilibrium
V An experimental equilibrium
X The corresponding strategies are not in experimental equilibrium

Fig. 1. Equilibria found for a market with $\lvert\mathcal{B}\rvert$=9,$\lvert\mathcal{S}\rvert$=3, and types normally distributed with parameters $\mu = 50, \sigma = 40$.

Notice that, for larger stocks, the sellers' expected utility decreases, since more units remain unsold, and the sellers pay for this. In contrast, the buyers' expected utility increases since the chances to be fully satisfied are larger.

Dominant experimental equilibria exist for [100] and [150]. In both cases, the sellers benefit most by implementing the RandS strategy. For [100], there are five additional experimental equilibria, three of which include sellers that benefit by choosing the buyers based on their deals (i.e., the sellers follow the OPO strategy). Sellers that implemented ORType, OType, DRType, DType

[5] For brevity, we annotate these settings by [70], [100] and [150], respectively.

with $C_t \in [0, 4]$ obtained poor expected utility; in all the cases there was a greedy or uninformed behavior that yielded a higher remuneration. For [150], there are four additional experimental equilibria.

For [70], five experimental equilibria were found, but neither dominates the others. Sellers that follow the OPO strategy gain a slightly (though significant) higher expected utility than sellers that implement RandS. Sellers that hold small stocks benefit from making buyers that behave LoyalP stay with them

When the buyers play RandB, the average utility of the sellers does not depend on their strategy. This can be explained by the fact that the sellers try to influence the choice of the buyers by their behavior. However, if a buyer chooses a seller randomly, regardless of the sellers behavior, the exact strategy that a seller uses is irrelevant. The three combinations of strategies comprising RandB buyers, and sellers who can act RandS, OPO, and DPO are experimental equilibria in the three settings of our simulations. Given that all of the buyers behave RandB, no single buyer will be motivated to deviate. Still, this equilibria is not dominant because, for the different stock sizes tested, there is always a punishing strategy for the buyers that yields higher expected utility for the buyers than when the buyers play RandB.

Interpretation of the Results. Since the dominant experimental equilibria include random sellers, learning the buyers' types cannot increase the sellers expected utility. We discuss this issue further in Section 5.

When a dominant experimental equilibrium is found, sellers that implement RandS conduce the market to a low and adequate level of liquidity. Sellers that act RandS also avoid arriving at a static local minima as it is achieved by the sellers that follow DPO.[6] As opposed to DPO, OPO sellers may cause the buyers to change sellers even when their distribution was good.

When the stock size is rather small, it seems that the appropriate recommendation for the sellers is to implement the OPO strategy. Since the stock is small, the seller will benefit most by not loosing the buyers he can satisfy, and should not take risks by following the RandS strategy.

As explained in Section 2.1, our implementation of RandS resembles the FIFS strategy, when the buyers approach a certain seller in a random order. FIFS is commonly used in physical stores. There, since buyers can see the arrival order, FIFS provides a sense of fairness. In virtual stores, though, a buyer is usually not aware of the other buyers. As resulted in our simulations, FIFS is a beneficial strategy for automated sellers as well when they hold equally-sized stocks, and the order of the arrival of the buyers to a given seller is random.

Our results show that the buyers should follow the strategy that punishes most the sellers, given their stock size. As the stock size becomes larger, the competition among the buyers decrease, and therefore the buyers can more severely

[6] DPO sellers lead to a market with the lowest possible level of liquidity because DPO implies that the majority of the buyers are at least partly satisfied, so they do not switch to other sellers. This is also true for "bad" situations, those in which buyers and sellers could benefit from the change.

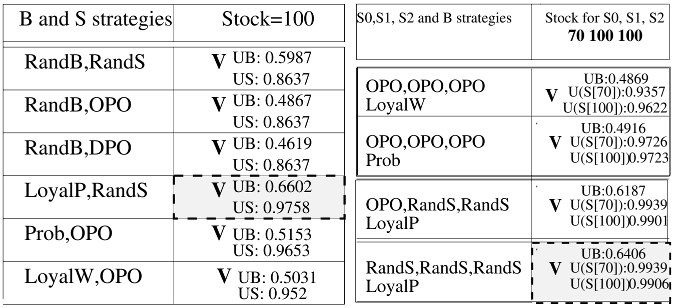
punish the sellers who do not satisfy them. All of the equilibria found included a punishing strategy for the buyers that yielded a higher expected utility than a less-punishing strategy.

5 Experiments with Non-conceding Buyers and Heterogeneous Markets

Non-conceding Buyers. In real markets, sellers seem to consider the type of the buyers for decisions regarding serving them. The results presented in Section 4 contrast this expectation, as sellers maximize benefits by random, type independent, choice of the buyers to be served. One may hypothesize that this resulted from utility functions implemented in the basic SEMI setting. These functions imply that a buyer's utility is proportional to the level to which his/her purchase-order was satisfied.

To examine this claim, we ran another set of simulations, where non-conceding buyers react to partial satisfaction of their purchase-order in an amplified dissatisfaction, expressed via a reduced utility. The utility of a buyer B^i was changed to be $U^i_{nc}(PO^{ik})$ as explained in Section 2.

The equilibria found for a market with 9 buyers, 3 sellers, a stock of 100 units and LS=0.5 are exactly the same equilibria found for the basic setting (see Figure 2). Notice that in all of the cases, the expected utility of the buyers is smaller than the one obtained in the basic setting due to the LS factor influence. Moreover, there exists a dominant experimental equilibria that is achieved when the buyers implement the LoyalP strategy and the sellers, the RandS strategy.



B and S strategies	Stock=100
RandB,RandS	V UB: 0.5987 US: 0.8637
RandB,OPO	V UB: 0.4867 US: 0.8637
RandB,DPO	V UB: 0.4619 US: 0.8637
LoyalP,RandS	V UB: 0.6602 US: 0.9758
Prob,OPO	V UB: 0.5153 US: 0.9653
LoyalW,OPO	V UB: 0.5031 US: 0.952

S0,S1, S2 and B strategies	Stock for S0, S1, S2 **70 100 100**
OPO,OPO,OPO LoyalW	UB:0.4869 V U(S[70]):0.9357 U(S[100]):0.9622
OPO,OPO,OPO Prob	UB:0.4916 V U(S[70]):0.9726 U(S[100])0.9723
OPO,RandS,RandS LoyalP	UB:0.6187 V U(S[70]):0.9939 U(S[100])0.9901
RandS,RandS,RandS LoyalP	UB:0.6406 V U(S[70]):0.9939 U(S[100])0.9906

Fig. 2. Equilibria found for a market with $|\mathcal{B}|$=9,$|\mathcal{S}|$=3, and types normally distributed with parameters $\mu = 50, \sigma = 40$. The leftmost figure considers non-conceding buyers. The rightmost figure considers sellers whose stocks are $Stock(S^0) = 70$, $Stock(S^1) = Stock(S^2) = 100$

The level of liquidity of the non-conceding buyers is larger than the level found in the basic setting. Even when the buyers' strategy induces them to return to a seller that has partially satisfied them, the buyers may behave as if the sellers have not satisfied them at all. This change of behavior applies to strategies that allow buyers to return to a partially satisfying seller.

Heterogeneous Markets. As the results of the previous experiments suggest, when sellers are equal, sellers will maximize benefits by following RandS. Yet, when the sellers hold different stocks, competition among the sellers can arise such that acquiring the buyers' types may help the sellers. We have run another set of simulations for a market with 9 buyers whose expected utility is $U^i(PO^{ik})$ as in the basic setting, and 3 sellers that hold different stocks to examine this hypothesis. Without loss of generality, seller S^0 holds a 70 unit stock, and sellers S^1 and S^2 hold each a stock of 100 units. The results are shown in Figure 2. Note that the utilities of the sellers were computed for each stock, i.e., $U(S[70])$ is the utility calculated for S^0 and $U(S[100])$, the average utility obtained by S^1 and S^2).

In the basic setting, RandB buyers were always part of an experimental equilibria. The results of the the heterogeneous experiments are different. Buyers are better-off deviating from RandB to LoyalP regardless of the sellers' strategies. The increased competition among the sellers enables the buyers to punish the sellers more severely. As can be seen from Figure 2, the combination of LoyalP buyers and RandS sellers is still the dominant experimental equilibria as it was in the basic setting for an homogeneous market with [100]. Note that, except for equilibria that include RandB buyers in the basic case (for [100] and [70]), the experimental equilibria arrived at in the heterogeneous case are the same as those arrived at in the basic case.

S^0 benefits less in the heterogeneous market than it does when all sellers hold stock 70 ($U(S[70])$=0.9939 in the heterogeneous market, and $U(S[70])$=0.9963 in the homogeneous market). This is due to the fact that larger sellers can satisfy more buyers' deals than S^0 who holds a smaller stock. These LoyalP buyers will keep returning to S^1 or S^2 and will not approach S^0 as long as they are at least partially satisfied. In contrast, S^1 and S^2 benefit from the competition ($U(S[100])$=0.9906 in the heterogeneous market, and $U(S[100])$=0.9854 in the homogeneous market). The buyers' benefit from the heterogeneous market is greater than their benefit in a homogeneous market where all the sellers hold 70 units though smaller than their benefit in a homogeneous market with sellers holding each 100 units.

Notice that the FIFS strategy, which was the dominant experimental equilibrium in the basic case, is still in equilibrium. And, the experiments suggest no evidence that seller diversity effects the need for type information.

6 Related Work

Scheduling algorithms (e.g., [3]) cannot be applied to E-markets because there is no global function which all the agents are trying to minimize. Agents react to each other's behaviors. Game theory and economics research have addressed issues of competition among buyers and of sellers that need to choose which buyer to supply (e.g., [4]). Nevertheless our assumptions differ from those assumed in these works. In our framework, buyers may consider the sellers' reputation when they choose which seller to approach. Reputation was studied as a function (e.g., [6]), as a social mechanism for avoiding untrustworthy parties [5], and as an expectation for quality. In our case, the buyers' strategy may change according to the service received from the sellers. The behavior of automated sellers situated in E-markets was investigated by Kephart et al. [2]. In particular, their focus was on dynamic changes in prices. There, time range was longer, and variations in stock were not handled as we do in this paper.

7 Conclusion

We examined strategies that seller agents can use to select buyers' purchase-orders in the face of limited stocks. We studied strategies that buyer agents can use to select sellers, based on their level of satisfaction.

It is unlikely to assume that both buyers and sellers will act as symmetric groups in E-markets. But, surprisingly, dominant experimental equilibria were found in our experiments, that show such symmetry. This suggests that the strategies' profiles that are in such equilibria are advantageous, and designers of automated electronic sellers and buyers should implement them. Agents will only loose utility by deviating from the group behavior.

We have found all of the experimental equilibria for the three scenarios of markets tested when all the sellers hold equally-sized stocks, variable-sized stocks and when the buyers are non-conceding. The main conclusion of this work is that sellers should behave randomly. RandS leads to a lower level of liquidity of the buyers in the market. As we hypothesize and show, lower levels of liquidity of buyers usually entail increased utility. Nevertheless, designers of agents for E-markets should be aware that local minima can be arrived at. The recommendation for the buyers is to punish the non-satisfying sellers as much as the stock size enables them, i.e., the severity of the punishment is inversely proportional to the amount of competition among the buyers.

References

1. C. V. Goldman, S. Kraus, and O. Shehory. Agent strategies: for sellers to satisfy purchase-orders, for buyers to select sellers. In *Proceedings of MAAMAW01*. 2001.
2. J. O. Kephart, J. E. Hanson, and A. R. Greenwald. Dynamic pricing by software agents. *Computer Networks*, 32(6):731–752, May 2000.
3. J. Sgall. On-line scheduling. *LNCS*, 1442:196–231, 1998.
4. D. R. Vincent. Modelling competitive behavior. *RAND Journal of Economics*, 23(4):590–599, 1992.
5. B. Yu and M. P. Singh. Small-world reputation management in online communities. *CIA2000*, LNAI 1860:154–165, 2000.
6. G. Zacharia. Collaborative reputation mechanisms for online communities. Master's thesis, MIT, 1999.

On the Logical Aspects of Argument-Based Negotiation among Agents

Luís Brito, Paulo Novais, and José Neves

Departamento de Informática
Universidade do Minho
Braga PORTUGAL
{lbrito,pjon,jneves}@di.uminho.pt

Abstract. The use of agents in Electronic Commerce (EC) environments leads to the necessity to introduce some formal analysis and definitions. Negotiations which take into account a multi-step exchange of *arguments* provide extra information, at each step, for the intervening agents, enabling them to react accordingly. This *argument-based negotiation* among agents has much to gain from the use of logical mechanisms. Although the use of logic to express arguments in Law is well known, EC poses new challenges. Concepts such as *round* and *probable conflict* are important in the formalization of negotiation arguments. The ideas of *conflict/attack* and *victory/defeat* are some of the most important to be formalized.

Incomplete information is common in EC scenarios therefore, arguments must take into account the presence of statements with an *unknown* valuation.

The set of rules that is to express the knowledge of an agent needs *priorization*; i.e., some rules have a higher priority than others. Priorities are usually set by the relative ordering of clauses, however, through *knowledge classification* and the introduction of *embedded priority rules*, an higher flexibility is reached.

Keywords: argument-based negotiation, agents, logic, formal specification

1 Introduction

Multiagent teamwork is an important area of agent research, with a growing number of applications, namely in the field of Electronic Commerce (EC), which has experienced an exponential growth in the past few years. On the other hand, innovations in network technology, combined with developments in areas such as Database and Knowledge-based Systems provided the basis for a quite successful kind of EC, the catalog sales one [10]. However, EC extends beyond this traditional vision, into areas such as those of automated negotiation and information feedback in production lines [2].

Indeed, the use of agent-based technology supplies the framework that is well suited to set the virtual environments that are found in EC, which, in turn, are

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 178–189, 2001.

characterized for being highly complex and dynamic, eminently distributed and anthropomorphic in their properties.

Although the use of multiagent systems in EC is not new, especially at the negotiation level, this approach has been hampered by the lack of a formal basis. Due to the abstract features present in each possible deal, a solution that shifts the burden of "téte-a-téte" negotiation to the method itself, is usually chosen. This is the reason why auction-based systems are more pervasive.

Arguments are common in real-world scenarios and provide a way of stating claims taking into account support knowledge (information); i.e., an agent is able to gather information (otherwise unavailable or of difficult access) from the arguments put forward by its counterpart. Therefore, an argument follows a logical line that starts at the first argument put forward, and finishes with the election of a possible winner or looser. The need for a formal basis to model argument-based negotiations leads to the use of logic. Logical formulas are extremely powerful, unambiguous and possess a set of interesting advantages [5]:

> *Expressing information in declarative sentences is far more modular than expressing it in segments of computer programs or in tables. Sentences can be true in a much wider context than specific programs can be used. The supplier of a fact does not have to understand much about how the receiver functions or how or whether the receiver will use it. The same fact can be used for many purposes, because the logical consequences of collections of facts can be available.*

The main contributions of this work are: (i) the definition of a common ground to situate the agent's reasoning mechanisms in EC environments; (ii) the use of formal tools (logic) to describe the rational behavior of agents involved in EC; (iii) the bridging of legal argumentation and argument-based negotiation; and (iv) the establishment of sound syntactic and semantic tools for argument-based negotiation.

2 Argument-Based Negotiation

When a set of agents meet under a Virtual Marketplace (VM), some kind of interaction may take place, namely by a process of offers and counter-offers, to support the modeling, analysis and enactment of the business process. The soundness of the process arises from the set of facts taken into consideration to produce an offer or counter-offer; i.e., the facts, taken from an ordered logic theory, lead to a logical conclusion, organizing themselves into an *argument.*

The players, namely the receiving agents, act on the conclusions reached by their counterparts and on the set of premises used to determine their behavior. There are two main ways of attacking the argument of a counterpart agent: *conclusion denial*, which assumes that the conclusion reached by the counterpart's argument is false; and *premise denial*, which assumes that one of the premises used to form the logical chain of a counterpart's argument is false.

The use of logic for the formalization of argument-based negotiation does not aim at the definition of the best dealing strategies (although the construction of problem-solving methods for that purpose may turn out to be more stable, taking into account the concepts stated in the formal theory). There are two main objectives: offers and counter-offers are logically justified and, the definition of *conflict/attack* among opposing parties is clearer. Without arguments, each agent has no way of ascertaining why their proposals/counter-proposals are accepted or rejected, due to the limited amount of exchanged information [4].

The importance of an *argument* has much to do with the time at which it arises; i.e., an argument may be deemed as a *looser* or a *winner* when facing a counter-argument, taking into account its sequence of evaluation. The definition of the concept of *round* is very important. A *round* is defined as the time-slot reserved for an offer/counter-offer launch by some agent. Notice, however, that although the argument of an agent may be able to *win* over some other argument, that does not mean the *negotiation process* ends there. This *negotiation process* is seen as the set of arguments exchanged among two agents, with several *victories*, *losses* and *ties*. Victory means *preponderance* over a counterpart's argument but it is not the end of an argumentative process.

The exchange of offers and counter-offers must stop when some conditions are satisfied. These conditions may or may not permit the definition of a winning set of arguments, and are to be considered in systems where the main concern is argument optimality. The strength of an argument can only be measured by the definition of some metrics on the receptor's side. A possible solution for ending an argumentative process could be based on a *threshold*; i.e., when the sum of the evaluations provided by the the receptor's metrics (over the set of arguments exchanged so far) reach a specific *threshold* value, *defeat* is assumed and stated by a predefined argument which is sent to the counterpart. Notice that the exchange of a predefined number of arguments might be taken as a limiting condition, to prevent large argumentative processes (leading to possible *tie* situations).

2.1 Global vs. Local Knowledge

Each element that composes an argument may come from one of two main sources: *global* or *local* knowledge. Global knowledge is shared by the intervening entities and is, therefore, independent of a particular experience or local state. Local knowledge derives from sources that are not common to every agent, giving way to the possibility of contradictory conclusions upon confrontation.

Contrary to the definitions found in logical formalizations in Law [7], the Knowledge Base (KB) embedded in each agent may be quite different. The use of global or local knowledge conditions the capacity to determine the *winner* of a confrontation. As expected, local knowledge is not the best starting-point for a *premise denial attack* (e.g., a claim such as "*my experience* tells me I sold item X for Y monetary units" is difficult to be stated as false by a the counterpart agent, because he can not say what are the particular experiences of the other agent).

However, *victory/defeat* are difficult to define under local knowledge, this knowledge is essential for the exchange of information among agents in a justifiable way. In Business-to-Business (B2B) or Business-to-Consumer (B2C) argumentation there is often no winner or looser, however, exchange of arguments (which are justifiable) among agents is essential so an acceptable situation for both parties is reached (even if an agent decides to drop, at any time, the negotiation). *Local* knowledge is important for an agent to reach another agent's acceptability region faster, during some deal [4].

2.2 Priorization

The different premises that compose the KB of an agent have different *priorities*; i.e., the application of a given premise is preferred to the application of another premise for a set of reasons, among which:

- **temporality**: some premise imposes itself over another due to the fact that it is more recent (e.g., the price of X was 20, nevertheless, it is now 25);
- **obligation/exception**: a premise imposes itself due to the fact that some conditions has to be fulfilled or a condition exception arises (e.g., the price of X is 20 for every country except for Portugal, where it is 15);
- **internal criteria**: due to some particular decisions, some agent may define its own priority rules over its KB (e.g., the price of X was 20 and is now 15, nevertheless, the price of 20 is to be considered as the valid one).

In logic programming languages, such as Prolog, some priority is established through the ordering of clauses. However, this kind of priority is too weak, giving way to the definition of new priority rules with well-specified semantics.

3 The Formal System

3.1 Basics

The formal system bases itself, as previously stated, in logic and, particularly, in logic programming (incorporating important concepts such as nonmonotonicity). Taking into account the Closed-World Assumption (CWA) (i.e., what is not stated as true must be false), the concept of negation-as-failure may be introduced [1].

A language is established in order to express the premises that compose each argument in a negotiation. Therefore, a definition for the representation of knowledge within each agent can be stated, as can some notation definitions.

Definition 1. (knowledge clauses) *The knowledge of each agent is expressed by a set of clauses with the form* $r_k : \forall_{x_1}, ..., \forall_{x_n} \varphi$*, where* $n \in N_0$*,* $k \in N_0$ *and* φ *is of the form* $\neg P_1 \vee \neg P_2 \vee ... \vee \neg P_{i-1} \vee \neg not\, P_i \vee ... \vee \neg not\, P_{i+j} \vee P_{i+j+1}$ *except for the disjunction's commutativity, where* $i, j \in N_o$ *(if* $i = j = 0$*,* $\varphi = \perp$*),* $P_1, ..., P_{i+j+1}$ *are atoms and 'not' is "negation-by-failure".*

Notation 1 *Clauses of the form* $\neg P_1 \vee \neg P_2 \vee ... \vee \neg P_{i-1} \vee \neg not\, P_i \vee ... \vee \neg not\, P_{i+j} \vee P_{i+j+1}$*, except for commutativity, are written as* $r_k : P_{i+j+1} \leftarrow P_1 \wedge P_2 \wedge ... \wedge P_{i-1} \wedge not\, P_i \wedge ... \wedge not\, P_{i+j}$*. These clauses are called "rules",* P_{i+j+1} *is called the "consequent" and* $P_1 \wedge P_2 \wedge ... \wedge P_{i-1} \wedge not\, P_i \wedge ... \wedge not\, P_{i+j}$ *is called the "antecedent".*

Notation 2 *Clauses of the form* $r_i : P_1 \leftarrow$ *are represented as* $r_i : P_1$*. These clauses are called "facts".*

An Extended Logic Programming (ELP) program (Π_{ELP}) is seen as a set of knowledge clauses as the ones presented above.

Each agent's KB (for systems which only take into account the priority established by the relative ordering of clauses – as it happens in Prolog) is taken from an ordered theory $(T, <)$, where T is a set of rules (premises) and $<$ is a non-circular ordering over T. The necessity for this non-circular ordered theory lies on two reasons: relative importance of rules (some rule is chosen over some other rule) and computational usability (a logic program written in a language such as Prolog needs some concrete ordering over the set of clauses).

The ordered theory $(T, <)$ may be extended to encompass the introduction, within each agent's KB of rules that express new priorities. This non-circular ordered theory for negotiation purposes is represented by $TN = (T, <, (S, \prec))$, where T is a set of rules that express knowledge such as an agent's *previous experiences*, $<$ is a non-circular ordering over T and $(S, \prec)$ represents the set of priority rules and their relative ordering.

3.2 Incomplete Information

Traditional Logic Programming (LP) programs are restricted in their semantics to a two-fold logic. Due to the CWA, everything that is not stated as being true is considered to be false. Therefore, these programs react only to the existence of positive information in the KB of an agent. This is, at least for agents in VMs, extremely restrictive due to the enormous quantities of negative and even unknown/incomplete information in negotiations (e.g., "product X *can not* be sold in Portugal"). Special emphasis must be given to representing unknown/incomplete information which is quite common. ELP programs reveal themselves extremely flexible in handling both positive, negative and even unknown/incomplete information. A new structure for knowledge clauses must be defined:

Definition 2. (extended knowledge clauses) *Knowledge clauses of the form* $r_k : P_{i+j+1} \leftarrow P_1 \wedge P_2 \wedge ... \wedge P_{i-1} \wedge not\, P_i \wedge ... \wedge not\, P_{i+j}$*, where* $k \in N_0$ *and* $P_1, ..., P_{i+j+1}$ *are atoms, become extended knowledge clauses of the same form when* $P_1, ..., P_{i+j+1}$ *are literals, i.e, formulas of the form* p *or* $\neg p$*, where* p *is an atom.*

Taking into account the possibility to define positive and negative information through extended knowledge clauses, a meta theorem-solver can be defined in order to establish theorem proving in a three-fold logic scenario (*true*, *false*, *unknown*). This meta theorem-solver is, itself, defined as a set of productions in traditional LP (with signature $demo : T, V \rightarrow \{true, false\}$, where T is the theorem and V the theorem's valuation [3]).

The concept of unknown/incomplete information is connected to that of *null values*. These elements are atoms that represent abstract concepts with no particular definition; i.e., elements which have a well-defined (or even non-defined) *range* of values as valid options. In commercial (negotiation-based) scenarios, the existence of *null values* with valid instantiations from a *well-defined set of values* is common. To illustrate this concept consider the following elements present at an agent's KB:

$$\begin{array}{l} sells(john, p1). \\ sells(mark, p2). \\ sells(james, p3). \\ sells(charles, someproduct). \\ imported(p1). \\ imported(p3). \end{array}$$

It is known that *charles* sells an imported product, however it is not known, specifically, which one. Using the meta theorem-solver, a set of queries might be put to agent's KB. The following set of queries (and respective valuations) demonstrates the desired knowledge behavior in an incomplete information scenario:

$$\begin{array}{ll} demo(sells(john, p1), V) & V = true \\ demo(sells(john, p4), V) & V = false \\ demo(sells(charles, someproduct), V) & V = true \\ demo(sells(mark, p3), V) & V = false \\ demo(sells(charles, p1), V) & V = unknown \\ demo(sells(charles, p4), V) & V = false \end{array}$$

This behavior is reached by complementing the previously presented KB with a set of clauses that states *negative information* and *exceptions*. Having defined *positive information* (present in the original KB), *negative information* (present in the extension) and the set of *exception clauses* (related to the set of elements with incomplete information), the meta theorem-solver can then be used. The extension to the original KB is as follows:

$$\begin{array}{rl} \neg sells(S, P) \leftarrow & not\, sells(S, P), \\ & not\, exception(S, P). \\ exception(charles, P) \leftarrow & sells(charles, someproduct), \\ & imported(P). \end{array}$$

3.3 Negotiation Arguments

After a theory and a language have been established, in order to represent each agent's knowledge/information (from which it will draw the justification for each offer/counter-offer), a definition for *argument* must be reached. An *argument* is to be constructed progressively, being the antecedent of each rule composed by the consequents of previous rules. This definition is, perhaps, the most important one in the logical formalization of argument-based negotiation.

Definition 3. (negotiation argument) *Taking the ordered theory TN, a negotiation argument is a finite, non-empty sequence of rules $\langle r_1, ..., r_n \rangle$ such that, for each rule r_j with P as a part of the antecedent, there is a rule r_i $(i < j)$ on which the consequent is P.*

This definition may encompass, through the ordered theory TN the meta theorem-solver as a set of clauses that might be used in a negotiation argument. However, to clarify the construction of negotiation arguments and, taking into account that the meta theorem-solver provides to all agents the same "semantic icing" for clause valuation, the definition of such meta theorem-solver might be external to composition of an argument. Therefore, a new definition of *argument*, for scenarios where incomplete information is available, is to be stated.

Definition 4. (negotiation argument with an implicit meta theorem-solver) *Taking the ordered theory TN, a negotiation argument is a finite, non-empty sequence of rules $\langle r_1, ..., demo(r_i, V_i), ..., r_n \rangle$ such that, for each sequence rule r_j with P as a part of the antecedent, there is a sequence rule r_i $(i < j)$ on which the consequent is P.*

Notation 3 (argument) *A negotiation argument represented by a finite, non-empty sequence of rules $\langle r_1, ..., demo(r_i, V_i), ..., r_n \rangle$, may be expressed as $\langle r_1, ..., \ r_n \rangle$ without loss of meaning, taking into account that $demo(r_i, V_i)$ is nothing more than a rule by itself.*

The use of such arguments, extended by a three-fold logic, is important due to their informative nature; i.e., one of the advantages of using argument-based negotiation lies in the fact that information is conveyed in such a way that the counterpart agents are able to evolve their counter-arguments in a parallel way (reaching a *cooperative* usage of knowledge).

The conclusion of an argument relates to the consequent of the last rule used in that same argument. Therefore, having in mind the use of such premise in further definitions, a formal statement of argument conclusion is to be reached.

Definition 5. (argument conclusion) *The conclusion of an argument $A_1 = \langle r_1, ..., r_n \rangle$, $conc(A_1)$, is the consequent of the last rule (r_n).*

As it has been stated, the nature of the knowledge each agent has (local/global) is relevant for arguments and counter-arguments. By composing an argument with rules or facts that spawn from local knowledge (e.g., previous experiences), the *attack* or counter-argument launched by the opposing agent during its *round* is conditioned (due to the fact that local knowledge is hard to deny).

Taking into account the two forms of argument attack (*conclusion denial* and *premise denial*), a *conflict* among two opposing agents (e.g., buyer/seller) can be formally specified.

Definition 6. (conflict/attack over negotiation arguments) *Let* $A_1 = \langle r_{1,1}, ..., r_{1,n} \rangle$ *be the argument of agent 1 and* $A_2 = \langle r_{2,1}, ..., r_{2,m} \rangle$ *be the argument of agent 2. Then,*

(1) if $r_{1,i} \in A_1$ *or* $r_{2,j} \in A_2$ *are local, the arguments are said to be in "probable conflict";*

(2) A_1 *attacks* A_2 *iff* A_1 *executes a conclusion denial attack or a premise denial attack over* A_2*;*

(3) A_1 *executes a conclusion denial attack over* A_2 *iff there is no local knowledge involved and* $conc(A_1) = \neg conc(A_2)$*;*

(4) A_1 *executes a premise denial attack over* A_2 *iff there is no local knowledge involved and* $\exists r_{2.j} \in A_2 - conc(A_2) : conc(A_1) = \neg r_{2,j}$*.*

Having in mind the use of rational agents (i.e., those that do not undermine their own actions and are able to formulate *coherent arguments*), a proper definition of *coherency* must be formulated and, on the other hand, posterior definitions for *victory/defeat* (which agent, if any, wins or looses an argument-based negotiation) must take into account this formulation.

Definition 7. (argument coherency) *An argument* $A_1 = \langle r_1, ..., r_n \rangle$ *is said to be "coherent" iff* $\neg\exists_{a_i,a_j} a_i, a_j \in subarguments(A) \wedge i \neq j : a_i \, attacks \, a_j$*.*

As stated earlier, the concept of *round* is extremely important for agents in VMs. All the logical definitions presented take into account a *mutual exclusion* principle over the set of active agents; i.e., at each turn, an agent expresses its judgment to a previously active agent. This *order*, defined by the *round* structure, implicitly states the direction of the *attacks* and the *victory/defea*t orientation (if it is A_1 that defeats A_2, or A_2 that defeats A_1). Therefore, taking into account the definition of *conflict/attack* and the concept of *round* it is possible to logically define the *victory/defeat* pair.

Definition 8. (victory/defeat of negotiation arguments) *Let* $A_1 = \langle r_{1,1}, ..., r_{1,n} \rangle$ *be the argument of agent 1 and* $A_2 = \langle r_{2,1}, ..., r_{2,m} \rangle$ *be the argument of agent 2 and* A_2 *is presented at a later "round" than* A_1*. Then,* A_1 *is defeated by* A_2 *(or* A_2 *is victorious over* A_1*) iff*

(1) A_2 *is coherent and* A_1 *is incoherent;*

(2) A_2 *is coherent, executes a conclusion denial attack over* A_1 *(coherent) and the conclusion rule of* A_2 *is prioritary (taking into account the* TN *theory) over* A_1*;*

(3) A_2 *is coherent, executes a premise denial attack over* A_1 *(coherent) and the conclusion rule of* A_2 *is prioritary (taking into account the* TN *theory) over* A_1*.*

3.4 Priorities

The necessity to establish priorities, within the set of clauses that compose an agent's KB, arises, as stated earlier, either from computational reasons or from the necessity of establishing new semantics.

The ordering of clauses in a KB (based on an ordered theory T) offers a simple priority mechanism. However, in large and dynamic KBs (as it is the case for EC-oriented agents) this order-based priority is difficult to maintain. On the other hand, it is common knowledge that a negotiating agent does not usually work with knowledge by the chronological ordering set upon the clauses (which is traditional in KBs for logical programming). Therefore, the introduction of priority rules in the KB itself is an interesting solution; i.e., the priority semantics is embedded in the agent's KB.

Typically, in negotiating agents, priority setting takes place with different kinds of knowledge (e.g., the *current knowledge* of an agent overpowers its *past experiences*). Taking this *knowledge classification* into account, the structure of each rule may be extended (notice, however, that this new definition for knowledge clauses is made with no prejudice towards previous definitions, such as *conflict/attack* and *victory/defeat*).

Definition 9. (knowledge clauses with classification) *A rule* $r_k : P_{i+j+1} \leftarrow P_1 \wedge P_2 \wedge ... \wedge P_{i-1} \wedge not\, P_i \wedge ... \wedge not\, P_{i+j}$*, takes the form* $K_l : r_k : P_{i+j+1} \leftarrow P_1 \wedge P_2 \wedge ... \wedge P_{i-1} \wedge not\, P_i \wedge ... \wedge not\, P_{i+j}$*, where* K_l *expresses the kind of knowledge represented by* r_k*, whenever the classification of knowledge is important.*

The ordered theory TN, expressing two relation orders ($<$, over the set of all clauses and $\prec$, over the set of all priority rules), may now be redefined taking into account the knowledge classification embedded in each rule. TN gives way to a new non-circular ordered theory, defined as $TN' = (T, <, (S, \prec), \sim)$, where the new relation $\sim$ defines a non-circular ordering over the different kinds of knowledge.

For example, the following KB expresses the supremacy of the knowledge of *market prices* over the knowledge of *previous experiences*:

$$\begin{array}{l} MP : r_1 : price(x, y). \\ PE : r_2 : price(x, z). \\ PRIO : r_3 : priority(PE, MP). \end{array}$$

Although the MP clause appears before the PE one, a priority ($PRIO$) rule is established stating that PE has priority over MP. The inference engine needs to take into account such ordering.

Definition 10. (priority clauses) *Priority clauses, which are embedded in the KB of each agent, are rules of the form* $PRIO : r_k : priority(K_i, K_j)$ *where* K_i *and* K_j *represent different knowledge classifications and* r_k *is the rule identification.*

Notice that priority rules are, by definition, set towards groups of clauses and not towards individual rules. This fact reduces computational complexity and expresses what is construeded as the behavior of an human on similar circumstances.

3.5 Examples

Some examples may be presented to illustrate the previous definitions. The mechanism used for argument checking is not presented due to space limitations. Let agents E and F be engaged in the process of buying/selling product *p1* in an environment with priority rules embedded in the KBs. Agents E and F share *general knowledge*, *market knowledge* and the set of *priority rules.*

$Agent\, E:$	
$PE : r_5 : price(p1, 143).$	$\%(experience)\, price\, for\, p1\, is\, 143$
$MK : r_7 : price(p1, 147).$	$\%(market)\, price\, for\, p1\, is\, 147$
$GK : r_1 : price(p1, 150).$	$\%(global)\, price\, for\, p1\, is\, 150$
$PRIO : r_4 : priority(PE, GK).$	$\%(priority)\, PE\, overpowers\, GK$
$PRIO : r_6 : priority(MK, PE).$	$\%(priority)\, MK\, overpowers\, PE$

$Agent\, F:$	
$MK : r_7 : price(p1, 147).$	$\%(market)\, price\, for\, p1\, is\, 147$
$GK : r_1 : price(p1, 150).$	$\%(global)\, price\, for\, p1\, is\, 150$
$PRIO : r_4 : priority(PE, GK).$	$\%(priority)\, PE\, overpowers\, GK$
$PRIO : r_6 : priority(MK, PE).$	$\%(priority)\, MK\, overpowers\, PE$

The argument given by agent E might be $A_E = \langle r_4, r_5 \rangle$, however, agent F might argue with $A_F = \langle r_6, r_7 \rangle$, representing a *conclusion denial attack* taking into account the priority rules shared by the community. Agent F is considered the *winner* due to the fact it uses an higher priority rule on the set of priority rules.

Consider the existence of agent I (for *"I"ncomplete information*). For an example of the use of incomplete information in the generation of arguments, take into account the statement of interest in the purchase of *p1* (which is an *imported good*) from agent *J*. Agent I intends to state, in a logical way, that its interest in that purchase is unknown (in order, perhaps, to ignite a price reduction by the counterpart – a simple information exchange situation). The KB of agent I might be:

$Agent\, I:$	
$GK : r_1 : sells(agentJ, importedgood).$	$\%(global)\, agentJ\, sells\, an\, imported\, good$
$GK : r_2 : imported(p1).$	$\%(global)\, p1\, is\, imported$
$LK : r_3 : interested(S, P) \leftarrow$	$\%(local)\, interest\, in\, product\, P$
$\quad - : - : sells(S, P).$	$\%\, sold\, by\, S$
$LK : r_4 : \neg interested(S, P) \leftarrow$	$\%(local)\, not\, interested\, in\, product\, P$
$\quad not - : - : interested(S, P),$	$\%\, sold\, by\, S,\, if\, interest\, or\, exception$
$\quad not - : - : exception(S, P).$	$\%\, fail\, to\, be\, proved$
$LK : r_5 : exception(S, P) \leftarrow$	$\%(local)\, exception\, if\, product\, P$
$\quad - : - : sells(S, importedgood),$	$\%\, is\, imported\, and\, S\, sells\, some$
$\quad - : - : imported(P).$	$\%\, imported\, good$

The argument generated by *I*, taking the previous KB into account, might be $A_I = \langle r_1, r_2, r_3, r_5, r_4, \quad demo(interested(agentJ, p1), unknown) \rangle$. This argument provides new information to the counterpart agent, upon which some action may be taken.

4 Related Work

The use of logic, and especially LP, for formalizing the concepts of arguing is not new. Much work has been done in the area of AI and Law, trying to establish models and frameworks for the process of legal arguing. However, areas such as EC have not experienced the same effort of logical definition.

The work of Prakken [7] and Sartor [8], [9] serves as the basis for the development of a set of logical definitions for agent-based EC. The relation established between legal argument and legal language, especially concepts such as dynamic priorities serve as a bridge for the world of automated commerce. In the area of priorities, the present work focuses on an *hybrid system*; i.e., the priorities are established as rules similar to the other rules present in the system, however, the inference engine (probably through a *demo* meta-predicate) needs to be adapted to contemplate such rules in a special fashion.

The establishment of an agent architecture that suits the logical characteristics imposed by the present model has been studied in the work of Novais, Brito and Neves [6]. The EBM agent serves as the basis for agents that are able to argue in a logical fashion.

The present framework (agent architecture/logical formalization) establishes a consistent bridge among arguing as it is seen in Law and argument-based negotiation.

5 Conclusions

The process of arguing in EC negotiation is similar to that found in the Law. However, concepts such as *round* and *probable conflict* spawn from the particular area of agent-based EC. An architecture that supports commercial dealings and logical arguing needs, therefore, to be considered. Simple rejection or acceptance of proposals launched by each agent in a particular deal lead to time consuming and inefficient negotiations. This rises from the fact that the proposer is “blindly” looking for a particular instance that might fall in the acceptability boundaries of the counterpart agent. Arguments, on the other hand, provide a way to cooperatively share information.

The use of incomplete information in the argumentation process is extremely important; indeed, arguments which rely on the *unknown* valuations of many predicates are able to extrapolate new courses of action. Therefore, a formal definition of argument-based negotiation that relies only on positive an negative clauses is, by nature, incomplete. As it is known, in many real-world situations, the true valuations of clauses on which a negotiation is based, are *unknown*.

Logic clauses representing facts and rules are well-suited for such task, establishing a theory easily fed into an inference engine. The establishment of a clear definition of what is understood as an argument permits later considerations of *conflict/attack* strategies.

For computational reasons, and due to the necessity of establishing priorities on the set of clauses, the knowledge each agent has needs to form an non-circular ordered theory. The simplest priorities, which spawn from a particular ordering among the clauses, is not enough to express some of the possible scenarios. Priorities are usually set over *knowledge bodies* (e.g., previous experiences, market knowledge, general knowledge) and may be expressed as simple rules. Systems that enforce knowledge classification over the set of rules do not differ significantly from other systems where such enforcement does not exist. Knowledge classification is achieved by a simple change in the skeleton of each rule and on its interpretation by a specific problem solver.

Through the present formalization, agents may be built in a logical fashion and arguing amongst agents has now a logical justification. The proximity to implementation languages (such as Prolog) is also an advantage.

References

1. Baral, C., Gelfond, M., *Logic Programming and Knowledge Representation*, Research Report, Computer Science Department, University of Texas at El Paso, El Paso, Texas, 1994.
2. Brito, L., Novais, P., Neves, J., *A General Agent-Based Architecture for Production Planning in Electronic Commerce Scenarios*, ESM2000 - European Simulation Multiconference, Ghent, Belgium, 2000.
3. Brito, L., Novais, P., Neves, J., *Temporality, Priorities and Delegation in an E-Commerce Environment*, 14th Bled Electronic Commerce Conference, Bled, Slovenia, 2001.
4. Jennings, N. R., Parsons, S., Noriega, P., Sierra, C., *On Argumentation-Based Negotiation*, International Workshop on Multi-Agent Systems, Boston, USA, 1998.
5. McCarthy, J., *Programs with Common Sense*, Proceedings of the Teddington Conference on the Mechanization of Thought Processes, 75-91, London, 1959.
6. Novais, P., Brito, L., Neves, J., *Experience-Based Mediator Agents as the Basis of an Electronic Commerce System*, Workshop 2000 - Agent-Based Simulation, Passau, Germany, 2000.
7. Prakken, H., *Logical Tools for Modelling Legal Argument*, Doctoral Dissertation, Free University, Amsterdam, 1993.
8. Sartor, G., *A Simple Computational Model for Nonmonotonic and Adversarial Legal Reasoning*, Proceedings of the Fourth International Conference on Artificial Intelligence and Law, ACM Press, 192-201, 1993.
9. Sartor, G., *A Formal Model of Legal Argumentation*, Ratio Juris 7, 212-226, 1994.
10. Tian, Z., Y Liu, L., Li, J., Chung, J., Guttemukkala, V., *Business-to-Business e-Commerce with Open Buying on the Internet*, Research Report, IBM T. J. Watson Research Center, 1998.

Introducing a Multi-agent, Multi-criteria Methodology for Modeling Electronic Consumer's Behavior: The Case of Internet Radio

Nikos Manouselis[1], and Nikos F. Matsatsinis[2]

[1,2]Technical University of Crete, Dept. of Production and Management Engineering,
Decision Support Systems Laboratory, University Campus,
Kounoupidiana, GR-73100, Chania, Greece
[1]nikoslovesyou@ergasya.tuc.gr
[2]nikos@dias.ergasya.tuc.gr

Abstract. This paper presents a step-by-step methodology for intelligent systems analysis and design, used in modeling and supporting the decision process of the electronic consumer. Borrowing methodologies and techniques from Operational Research and Decision Making, detailed and accurate models of the consumer buying and negotiating behaviors are constructed. Furthermore, a multi-agent architecture is proposed and outlined, encompassing and engaging these models in order to cooperate with electronic marketplaces, for the needs of the consumer to be best met. This methodology is applied to a specific case of study, the case of the Internet radio listeners.

1 Introduction

In order to model the behavior of a consumer a lot of different and often non-quantitative parameters have to be measured; the contribution of a multi-criteria approach in analyzing consumer's preferences is thus invaluable. Research regarding answers to these matters is under development by several research teams throughout the world. The ideas outlined by the Software Agents Group of MIT are very attractive, as far as modeling of electronic consumers behavior is concerned (Guttman *et al.* [4], Moukas *et al.* [8], Maes *et al.* [7], Chavez & Maes [2]). Moreover, Terpsidis *et al.* [10] propose an agent-based re-engineering modeling process for electronic commerce having in mind the way electronic commerce will affect retail market behavior; the guidelines they are defining for modeling the Consumer Buying Behavior generally adopt Marketing concepts. Most of these approaches show the necessary attention towards the multi-attribute aspect of modeling the needs of electronic consumer, but we believe there is still work to be done in the area of equipping the mediating agents with robust and dynamic (not static) models of the consumer buying and negotiation behaviors.

In this paper we are going to describe a new approach to this problem concerning electronic consumers, borrowing techniques and methodologies from the sciences of Operational Research and Decision Making. The preference disaggregating analysis techniques are a perfectly adaptive tool in creating a behavior model of the consumer, based on the criteria taken under serious consideration when he/she chooses a product. The case under study is going to be that of an Internet radio listener, who makes decisions regarding the selection of the radio station he prefers, depending on various criteria that affect his choice. The model of the radio listener cannot be easily approached by the classic methods of marketing because his characteristics vary through time and location, and the creditability of the personal profile he imposes cannot be safely verified. The way to avoid statistically describing (and hence locating) a potential consumer is simple: monitor each specific listener's tastes and habits.

The paper is organized as follows. Section 2 presents the definition of the decision modeling process of the problem. In Section 3 the agent-based modeling methodology is presented and applied. Section 4 summarizes and concludes the basic concepts of the methodology; it also outlines some basic guidelines for future expansions of the methodology

2 Decision Problem Modeling

Consumer Behavior can be defined as the decision process and acts of people involved in buying and using process. The general methodology of decision making-problems includes four modeling steps, beginning with the definition of the object of the decision and ending with the activity of decision aid [9]. This methodology is closely followed in our procedure of defining the model of the decision problem of the Internet radio listener. The proposed steps are the following:

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 190–195, 2001.

i. Defining the object of decision, that is the set of potential actions (products or services) and the problematic of the decision. That includes definition of the decision variables in form of a consideration set A and definition of whether we want to rank, sort, choose one or simply comment on the action included in A.
ii. Studying the parameters influencing decision and defining a set of criteria that will help to model the consumer behavior. This step regards modeling a consistent family of criteria, assuming that these criteria are non-decreasing value functions, exhaustive and non-redundant.
iii. Choosing an appropriate multi-criteria aggregation method. The decision maker is asked to express his global preferences on a reference set A_R, taking into consideration the evaluations of the reference actions in A_R on all criteria. When then use this previous experience (expressed by the evaluations on the actions in the reference set) to predict the evaluations of the actions belonging in the real set A [5].
iv. Implementing the decision aid system, according to the infrastructure proposed. To achieve this goal, we built an agent-based "surveillance" system that monitors the consumer's preferences and behavior, and creates a dynamically changing model of this behavior while interacting with him via a suitable designed graphical environment.

These steps are described in details, at the following sub-sections.

2.1 Object of Decision, Potential Actions and Problematic

The decision concerns the choice of an Internet broadcasting radio station, therefore the set of potential actions is the choice of one or more of the available broadcasting radio stations - that is the definition of a sub-set B from the original set A of the available actions. Each consumer makes a choice of a certain number of actions (a.k.a. radio stations) and defines strictly the preferable number of actions to be included in set B. Therefore, we conclude that the problem regards rather "ranking" than "sorting" of the potential actions (problematic γ [9]). Then the first N actions are selected.

2.2 Family of criteria

The next step is modeling a consistent family of criteria, assuming that these criteria are non-decreasing value functions, exhaustive and non-redundant. Each criterion is defined on A as it follows:

$$g_i : A \rightarrow [g_{i*}, g_i^*] \subset \Re / a \rightarrow g(a) \in \Re \,, \qquad (1)$$

where $[g_{i*}, g_i^*]$ is the criterion evaluation scale, with g_{i*} the worst level of the ith criterion, g_i^* the best level of the ith criterion, $g_i(a)$ the evaluation or performance of action a on the ith criterion and $\underline{g}(a)$ the vector of performances of action a on the n criteria [5].

A consistent family of criteria in this case of study should contain only qualitative ones and fully describe the Internet radio station as a product or service, according to the judgment of the analyst (i.e. speech, advertisements, experimentation, quality of the signal and maximum number of listeners connected).

2.3 Preference model

There are many different methods of multi-criteria analysis which can be recommended depending on the circumstances of decision making. A well-known method, belonging to the subset of total aggregation methods that compute trade-off coefficients between criteria, is the UTA method. We chose to use the UTA method as the preference modeling method, so let us briefly introduce some of its basic ideas.

The UTA method [5] aims at inferring one or more additive value functions from a given ranking R on a reference set of actions A_R. The method uses special linear programming techniques to assess these functions so that the rankings obtained through these functions on A_R are as consistent as possible with the given one. In this way we obtain closely optimal modeling of the consumer's behavior, taking both quantitative and qualitative criteria under consideration. The criteria aggregation model in UTA is assumed to be an additive value function of the following form:

$$u(\underline{g}) = \sum_{i=1}^{n} p_i u_i(g_i), \tag{2}$$

subject to normalization constraints

$$\begin{cases} u_i(g_{i*}) = 0, u_i(g_i^*) = 1, \forall i = 1,2,...,n \\ \sum_{i=1}^{n} p_i = 1 \end{cases} \tag{3}$$

where u_i (i= 1, 2, ..., n) are non-decreasing real valued functions (named marginal value or utility functions and normalized between 0 and 1), g_{i*}, g_i^* are the lower and upper bounds respectively for the value of each criterion, and p_i is the weight of u_i. The UTA method has several interesting features [1]: it makes possible the estimation of a nonlinear additive function which is obtained by the use of a linear program that provides convenient piecewise linear approximation of the function, and the only information required from the decision maker are the global stated preferences between the different actions of the reference set A_R.

We have to note here that preference aggregation methods like UTA are generally suitable for use when there are difficulties in directly obtaining from the user the values of the preference model. In most simple cases [3], it may be better for a user to just express the attribute preferences rather than spend time making a series of product choices which will (at best) infer the same preferences. We chose UTA in the case of e-commerce applications, since it generally copes well with noisy or inconsistent data [1], it is conducive to changes in product preferences and its not as time-consuming, redundant and boring as the direct definition of each preference attribute for each category of products.

2.4 Decision aid

In order to support the decision maker (hence the electronic consumer) to make his/her selection, we propose a multi-agent based methodology. The basic idea of this method is creating one behavior model for each consumer entering an electronic market, and not just simulating his behavior using statistical tools as it is done using the classic methodologies. The way this is achieved by the multi-agent system is presented here:

i. The decision system obtains from the consumer a first evaluation of a test set of products and a weak-order ranking of them. The system also gathers information regarding the negotiating attitude of the user. The user indicates how flexible he/she is on the value of each criterion used in describing a radio station.

ii. The system uses the UTA method in order to evaluate the marginal and global utilities, creating thus a model of the consumer's behavior and a model of the negotiating attitude and trade-offs [6].

iii. The system electronically checks new competitive products entering the target market, and proposes to the consumer those fitting his model. During the search in the electronic market, the system uses the negotiating parameters defined by the user, as guidelines for the searching and negotiation procedures.

iv. The system monitors the reaction of the user on these proposals, by monitoring changes of the personal ranking of preferred products and of the way he/she explains the satisfactory or non-satisfactory points of the products proposed.

v. Whenever a change of ranking occurs, the model of the consumer is re-constructed.

In order to analyze and design this system, we utilize a methodology proposed by Wooldridge *et al.* [11]. This methodology is founded on the view of the system as a computational organization consisting of various interacting roles, and it deals with both the macro-level (societal) and the micro-level (agent) aspects of systems. Both the methodology and the way it is implemented in our case of study are presented in the next section.

3. Agent-Oriented Analysis and Design

The methodology proposed by Wooldridge *et al.* [16], allows an analyst to go systematically from a statement of requirements to a design that is sufficiently detailed to be directly implemented. The main agent-specific concepts this methodology provides, can be divided into two categories: abstract and concrete. Abstract entities are those used during *analysis phase* to conceptualize the system, but which do not necessarily have any direct realization within the system. Concrete entities, in contrast, are used within the *design process*, and are aimed to have direct counterparts in the run-time system. The objective of the *analysis phase* is to develop an understanding of the system and its structure, without reference to any implementation detail. The analysis stage is summarized in the following steps:

i. Identify the roles in the system.
ii. For each role, identify and document the associated protocols. Protocols are the patterns of interaction that occur in the system between the various roles.
iii. Using the protocol model as a basis, elaborate the roles model.

Through the *design process*, our aim is to transform the analysis models into a sufficiently low level of abstraction that traditional design techniques may be applied. The design process involves generating three models:

i. Create an *agent model*, by aggregating roles into agent types, refining to form an agent type hierarchy and documenting the instances of each agent type using instance annotations.
ii. Develop a *services model*, by examining protocols and safety and liveness properties of roles.
iii. Develop an *acquaintance model* from the interaction model and agent model.

3.1 Analysis phase

Following the above described methodology we first go through the *analysis phase* of the system. Throughout this phase, three main models have to be defined: the *prototypical roles model*, the *interaction model* and the *full roles model*.

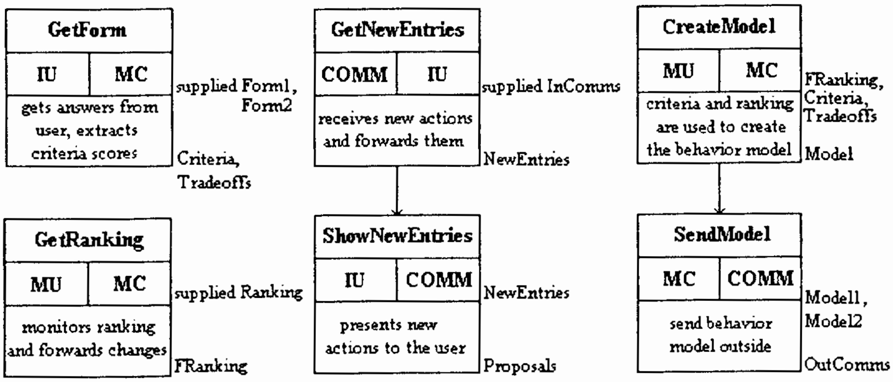

GetForm
IU
MC
gets answers from user, extracts criteria scores
supplied Form1, Form2
Criteria, Tradeoffs
GetNewEntries
COMM
IU
receives new actions and forwards them
supplied InComms
NewEntries
CreateModel
MU
MC
criteria and ranking are used to create the behavior model
FRanking, Criteria, Tradeoffs
Model
GetRanking
MU
MC
monitors ranking and forwards changes
supplied Ranking
FRanking
ShowNewEntries
IU
COMM
presents new actions to the user
NewEntries
Proposals
SendModel
MC
COMM
send behavior model outside
Model1, Model2
OutComms

Fig. 1. Interactions Model (protocol definitions)

The *prototypical roles* needed in the agent-based DSS are mainly four, since there are four different categories of actions to be elaborated. First an entity must interact with the user, gather information needed from him/her, and present the results of all the undergoing procedures. A second entity needed is one to monitor user's actions, without directly interacting with him/her, and without allowing interferences with the ingoing procedures. The third entity has to be involved with the construction of the behavior models using the information provided by the previous two entities, and by utilizing the UTA method. Finally, one last entity is needed in order for the system to communicate with the outside world, hence the electronic market. This communication entity sends the consumer behavior profile of the user to agents utilized in the electronic market and receives sets of new actions matching this profile, in order to be proposed to him/her.

All the interactions between the roles described above, are clearly depicted in the identification and documentation of the associated protocols, in the *interaction model* of Figure 1. After defining the protocols associated with each role, it is possible to create a fully detailed schema for each role, containing its description, and the permissions and responsibilities attributes associated with it.

3.2 Design Process

The purpose of the *agent model* is to document the various agent types that will be used in the system and the agent instances that will realize these agent types at run-time. As it is depicted in Figure 2, three agents are required for the implementation of the proposed architecture. These three agents (namely James, Mr Q and Manypenny) incorporate the necessary roles, as shown. More specifically:

- James: the structure of this agent encapsulates the functions of the two roles that demand interaction with the user, either directly or indirectly. These roles are InteractWithUser (performing the direct interaction with the user) and MonitorUser (performing the indirect interaction with the user).
- Mr Q: this agent is responsible for utilizing the multi-criteria algorithms, in order to create the behavior models. This agent type corresponds to the Model Creator role and can exist in the system in mostly two instances: creating the consumer and creating the negotiating behavior model.
- Manypenny: this agent is responsible for gathering the necessary information from the agent system (e.g. the behavior models) and formulating it in a comprehensive from the electronic market form. Similarly, the incoming messages (e.g. new proposed rankings) are translated and forwarded to the regarded agents of the model.

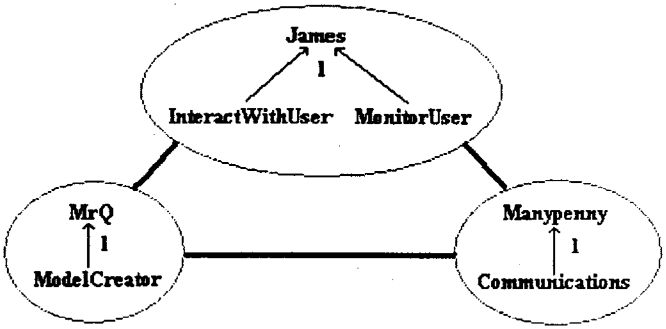


Fig. 2. Acquaintance model incorporating the Agent model

It should be noted that implementation details regarding the tools used by the agents (e.g. the multi-criteria methods used in the models creations or the communication and negotiation templates) are independent from the infrastructure of the agent model, and from the architecture proposed. It is obvious then that these implementation details or characteristics can be dynamically altered without influencing the validity of this model.

3.2 Methodology Application

The proposed methodology is implemented and tested in a prototype multi-agent testbed. The prototype system implements interactions taking place between the agents of the customer-based decision support system, and also simulates the environment of an electronic market, where the representatives of the customers communicate with a great number of merchants making different multi-attribute proposals. Several negotiation models are also tested and the methodology has proven to provide remarkable results [6], especially in cases where the customer is choosing several products from a very large number of available proposals. The user's agents use a multi-attribute preference model in order to filter the proposals and select the most interesting ones.

4. Conclusions and Future Work

As it is stated in [4], there are several limitations in modeling consumer behavior. For example, research focuses primarily on retail markets (although many concepts pertain to business-to-business and consumer-to-consumer markets as well). Even within retail, not all shopping behaviors are captured (e.g. impulse purchasing). Also, electronic commerce covers a broad range of issues, some of which are beyond the scope of a consumer behavior model (e.g. back-office management and other merchant issues). Nevertheless, the consumer behavior model is a powerful tool to help us understand the roles of agents as mediators in electronic commerce. In this way, our methodology provides the analyst with a fully described method of implementing a multi-agent consumer behavior modeling system. This methodology includes aspects of the analysis not addressed in other approaches, like encompassing both buying and negotiating models when modeling the consumer behavior. The models can be created using the described UTA method or with every other appropriate multi-attribute modeling method, without the system validity being affected.

There is still enough work to be carried as far as the proposed methodology is concerned. For example, parameters like choice of detailed negotiation tactics or use of reputation mechanisms are not addressed here. Moreover, integrity rules for the overall system are not yet fully specified. A compatible electronic marketplace should be fully deployed, in order to fine-tune the characteristics of the cooperating systems. Regarding the application of the methodology in the specific case of Internet radio listeners, there is still work to be done in modeling important details of the consumer buying and negotiating behavior. As we noted in Section 2, the enhanced model should include special criteria, depending on the judgment of the system analyst (e.g. music content, or news vs. fun orientation). Moreover, negotiation parameters should definitely include a reputation mechanism, since listeners tend to form strong group "alliances", with similar tastes. All of the above are under consideration and will be included into the prototype version of the radio listener aiding system.

References

1. Beuthe, M., Scannella, G.: Comparative analysis of UTA multicriteria methods. European Journal of Operational Research, 130, 2001.
2. Chavez, A., Maes, P.: Kasbah: An Agent Marketplace for Buying and Selling Goods. Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, UK, April 1996.
3. Guttman, R., Maes, P.: Agent-Mediated Integrative Negotiation for Retail Electronic Commerce. Proceedings of 8th MAAMAW'97, Sweden, May 1997.
4. Guttman, R., Moukas, A., Maes, P.: Agent-mediated Electronic Commerce: A survey. Knowledge Engineering Review, June 1998.
5. Jacquet-Lagreze, E., Siskos, Y.: Preference disaggregation: 20 years of MCDA experience. European Journal of Operational Research, Vol. 130, pp. 233-245, 2001.
6. Karampiperis, P., Manouselis, N., Matsatsinis, N.F., Siskos, Y.: UTAGREE: Multi-Attribute Utility Models for Automated Negotiation. Paper under review.
7. Maes, P., Guttman, R., Moukas, A.: Agents that Buy and Sell: Transforming Commerce as we Know It. Communications of the ACM, March 1999.
8. Moukas, A., Guttman, R., Maes, P.: Agent-mediated Electronic Commerce: an MIT Media Laboratory Perspective. Proceedings of the International Conference on Electronic Commerce, 1999.
9. Roy, B.: Multicriteria Methodology for Decision Aiding. Kluwer Academic Publishers, 1996.
10. Terpsidis, I., Moukas, A., Pergioudakis, B., Doukidis, G., Maes, P.: The potential of Electronic commerce in re-engineering consumer-retailer relationships through Intelligent Agents. In: J.-Y. Roger et al. (eds.): Advances in Information Technologies: The Business Challenge. IOS Press, 1997.
11. Wooldridge, M., Jennings, N., Kinny, D.: A Methodology for Agent-Oriented Analysis and Design. Proceedings of 3rd International Conference on Autonomous Agents, Seattle, pp. 69-76, 1999.

Modeling Commercial Knowledge to Develop Advanced Agent-Based Marketplaces for E-commerce

Martin Molina

Department of Artificial Intelligence, Technical University of Madrid
Campus de Montegancedo s/n, 28660 Boadilla del Monte (Madrid), Spain
mmolina@fi.upm.es

Abstract. This paper argues about the utility of advanced knowledge-based techniques to develop web-based applications that help consumers in finding products within marketplaces in e-commerce. In particular, we describe the idea of model-based approach to develop a shopping agent that dynamically configures a product according to the needs and preferences of customers. Finally, the paper summarizes the advantages provided by this approach.

1 Introduction

In the knowledge engineering field, a set of methods and techniques has been proposed the last decade to decrease the effort of building large and complex knowledge systems. One of the important ideas of this set of solutions is that it is useful to follow a *model-based* approach, which can be particularly appropriate in the context of web-based applications. In e-commerce, special kind of software tools conceived as shopping agents [1] are oriented to simulate the behavior of an employee of a company that helps a customer in finding and selecting an appropriate products. Shopping agents are able to perform different specialized tasks such as: customer needs interpretation, product configuration according to the needs, justification of proposals based on utility factors, etc. To carry out these tasks, this type of agents must combine different types of deep knowledge about the commercial relation, which requires a flexible knowledge representation to automatically provide efficient answers and an adequate user-system interaction.

Thus, this paper presents, first, a summary of the concept of model-based development of knowledge systems and, then, the paper presents how this technology can help in building advanced intelligent sales assistants, illustrated with the case of a shopping agent for configurable products. Finally, the paper presents a summary of the advantages provided by the model-based approach for the development of shopping agents.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 196–201, 2001.

2 Model-Based Development of Knowledge-Systems

The model-based approach has been recently followed by different methodologies for system analysis and design. Based on this approach, explicit abstractions about an observed system are formulated by using a particular formal representation that facilitates an adequate comprehension of the system architecture and, consequently, an appropriate level of flexibility for maintenance and reuse. In the field of knowledge engineering, a knowledge model can be formulated as an abstraction of the knowledge that an observer (the knowledge engineer) ascribes to a human expert to support a particular problem-solving competence. Some recent methodologies for knowledge system development follow this model-based approach [2],[3].

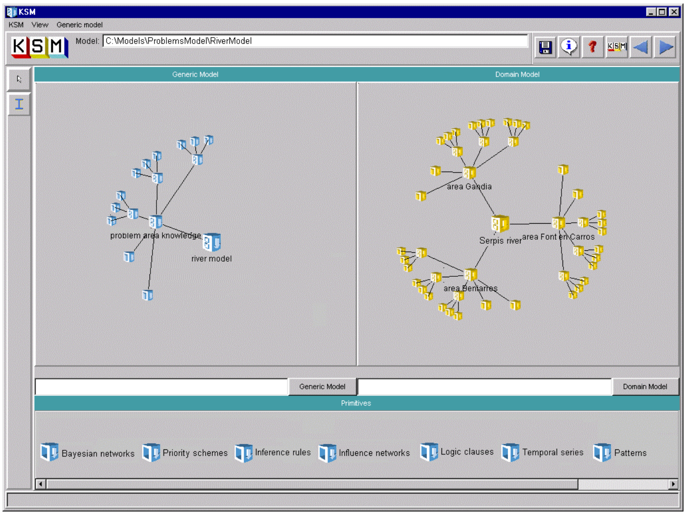
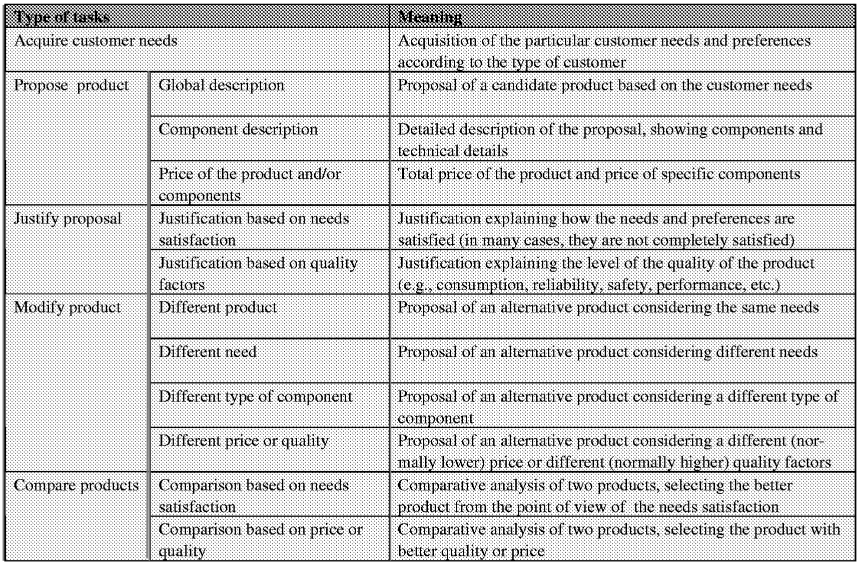


Fig. 1. User interface of the KSM software environment for knowledge modeling.

This type of methodologies normally provide a set of abstract description entities based on natural intuitions that capture different epistemological issues better than the traditional view of information processing, using a particular logical level for system description called *knowledge level* [4]. Entities such as tasks, problem-solving methods, domain ontologies, types of knowledge bases, etc. are normally used in this context. The model-based approach also facilitates the management of abstract reusable designs (conceived as general problem solving methods, PSM) that can be used to guide the development of new applications and, therefore, decrease the effort of knowledge acquisition. Specific software tools such as KSM [5] (figure 1) can facilitate the development of the operational version of such models. The models developed with this tool that can be also combined with an agent-based approach [6]. KSM produces the operational version using knowledge-based software components and help developers and end-users in creating and maintaining complex sets of knowledge bases with different symbolic representations.

3 Modeling Commercial Knowledge of Sales Assistants

The analysis of commercial knowledge corresponding to a sales assistant shows the presence of different types of expertise and reasoning processes that need to be adequately structured. In the following, we describe how we applied the recent knowledge modeling approach to produce a final model that support an advanced virtual web-based system for configurable products. Basically, we followed three main phases: (1) analysis of user-system interaction to identify top-level tasks, (2) knowledge-level analysis to formulate a generic model that supports the previous user-system interaction, and (3) system design at symbolic level to produce the operational version. This section presents a summary of the first and second phases. Information about the third one (symbolic representation) together with an application in the field of photography equipment can be found at [7].



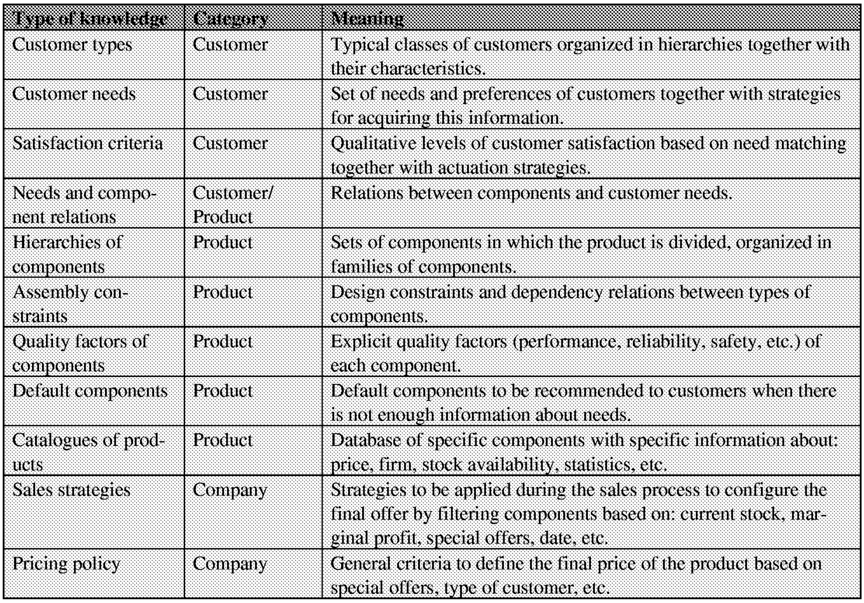
Type of tasks		Meaning
Acquire customer needs		Acquisition of the particular customer needs and preferences according to the type of customer
Propose product	Global description	Proposal of a candidate product based on the customer needs
	Component description	Detailed description of the proposal, showing components and technical details
	Price of the product and/or components	Total price of the product and price of specific components
Justify proposal	Justification based on needs satisfaction	Justification explaining how the needs and preferences are satisfied (in many cases, they are not completely satisfied)
	Justification based on quality factors	Justification explaining the level of the quality of the product (e.g., consumption, reliability, safety, performance, etc.)
Modify product	Different product	Proposal of an alternative product considering the same needs
	Different need	Proposal of an alternative product considering different needs
	Different type of component	Proposal of an alternative product considering a different type of component
	Different price or quality	Proposal of an alternative product considering a different (normally lower) price or different (normally higher) quality factors
Compare products	Comparison based on needs satisfaction	Comparative analysis of two products, selecting the better product from the point of view of the needs satisfaction
	Comparison based on price or quality	Comparative analysis of two products, selecting the product with better quality or price

Fig. 2. Types of tasks performed by a sales assistant.

The interaction between sales assistant and customer is based on a kind of negotiation process where: (1) the assistant recommends candidate product configurations based on the interpretation of the customer needs (i.e., a successful relation with the customer must be based on *the needs* instead of *the technical details of the product* [8]), (2) the assistant must be able of justifying the proposals with convincing explanations, for example, based on utility factors, and (3) the customer must be able of changing (total or partially) the proposals. Figure 2 shows the set of tasks that we identified for a sales assistant according to the previous requirements.

In a second step, we analyzed the commercial knowledge that could provide the previous interaction. Basically, the sales assistant needs to bring together at least three kinds of expertise: (1) knowledge about *the products*, (2) knowledge about *the cus-*

tomer, and (3) knowledge about *the company* interests. The reasoning process developed by the assistant considers different issues from these three different knowledge sources, sometimes combining contradictory criteria that must be solved with additional strategic knowledge. Figure 3 shows a summary of the corresponding types of knowledge to support the tasks identified in the previous section.



Type of knowledge	Category	Meaning
Customer types	Customer	Typical classes of customers organized in hierarchies together with their characteristics.
Customer needs	Customer	Set of needs and preferences of customers together with strategies for acquiring this information.
Satisfaction criteria	Customer	Qualitative levels of customer satisfaction based on need matching together with actuation strategies.
Needs and component relations	Customer/ Product	Relations between components and customer needs.
Hierarchies of components	Product	Sets of components in which the product is divided, organized in families of components.
Assembly constraints	Product	Design constraints and dependency relations between types of components.
Quality factors of components	Product	Explicit quality factors (performance, reliability, safety, etc.) of each component.
Default components	Product	Default components to be recommended to customers when there is not enough information about needs.
Catalogues of products	Product	Database of specific components with specific information about: price, firm, stock availability, statistics, etc.
Sales strategies	Company	Strategies to be applied during the sales process to configure the final offer by filtering components based on: current stock, marginal profit, special offers, date, etc.
Pricing policy	Company	General criteria to define the final price of the product based on special offers, type of customer, etc.

Fig. 3. Types of knowledge ascribed to the sales assistant for configurable products.

According to the recent knowledge-engineering methodologies, as it was presented in the previous section, a general knowledge model was formulated as a set of hierarchies of tasks, problem-solving methods and types of knowledge bases. The figure 4 shows a general view of the knowledge model. The figure 4 shows the main global tasks that support the interaction with the customer, associated to the corresponding knowledge models. Figure 4 also shows in more detail the model of one of the main tasks of the sales assistant: *configure product*. This model is graphically represented as a hierarchy of tasks (circles) and methods (rectangles) with types of knowledge bases (cylinders).The task *configure product* is carried out by an adaptation of a general artificial intelligence method for design problems, called *routine design* [9]. The basic idea of this method is to divide the whole design decision in partial classification tasks corresponding to the different components. The whole design is found through a tentative search that proposes hypotheses of design that are rejected when the corresponding design constraints are not satisfied, which forces to backtrack to generate alternative proposals.

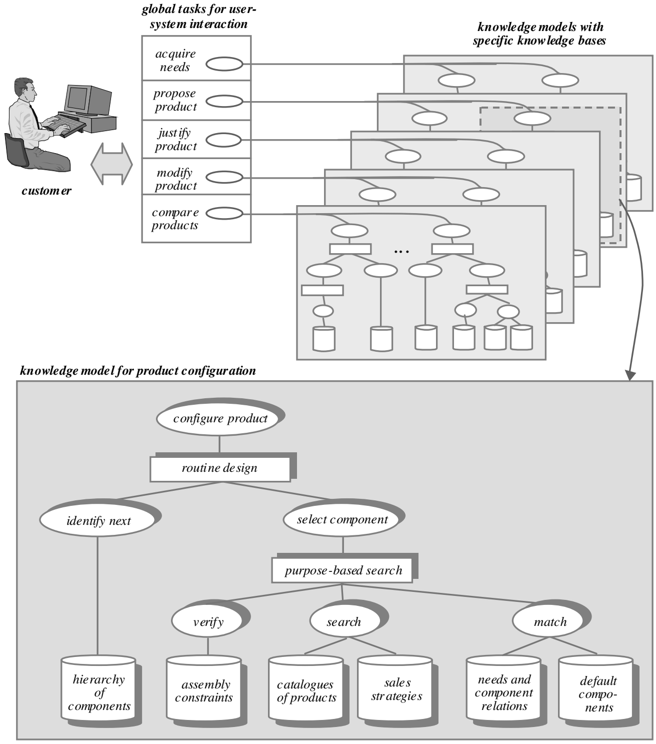


Fig. 4. General view of the knowledge organization for the intelligent sales assistant.

4 Discussion and Conclusions

In summary, the paper shows how the recent advances in the field of knowledge modeling can be applied to develop advanced and complex agent-based systems in e-commerce. In order to cope with the analysis of this knowledge and the implementation of the system, the model-based approach provided the following advantages:

1. *Guided knowledge acquisition*. The identification of standard classes of tasks (e.g., diagnosis, configuration, classification, etc.) allows developers to use existing knowledge-based patterns that guide the knowledge acquisition process. In the case of sales assistance, the interpretation of customer needs can be viewed as a classification task (to be done, for example, with the *heuristic classification* method) and the dynamic assembly of components for product configuration is a configuration task (e.g., *routine-design* method).
2. *High level of representation*. The *knowledge-level* concept provides an additional logical level for application formulation. With this, it is possible to identify the cognitive resources of the system without considering symbolic implementation issues, which is useful to formulate the structure and organization of the system before the implementation with more intuitive, natural and abstract terms.

3. *Generic reusable design for different domains*. The resulting knowledge model can be abstractly defined using an abstract terminology and types of knowledge bases. Thus, the model is not committed with a specific domain, so it can be reused for different fields. For example, the model described in this paper identifies the knowledge and reasoning processes of a virtual sales assistant for configurable products. But its organization is general, i.e., it can be applied to different domains (e.g., photography equipment, computers configuration, etc.).
4. *Advanced software environments for implementation*. For the implementation of the model, advanced software environments can be used to help developers in building the final system with the help of knowledge-based software components (e.g., KSM [5]).

The paper presents an innovative knowledge-based model of a virtual assistant that simulates the dynamic configuration of a product according to the interpretation of customer needs. This type of service supposes a significant advance compared to the general tools that can be found in the current marketplaces. The proposed model together with the modeling techniques can be the basis for a new generation of agent-based marketplaces that provides more assistance to customers.

References

1. Doorembos R., Etzioni O., Weld D.: “A Scalable Comparison-Shopping Agent for the World Wide Web”. Proceedings of the First International Conference on Autonomous Agents. Marina del Rey, California, USA. February, 1997.
2. Schreiber G., Akkermans H., Anjewierden A., De Hoog R., Shadbolt N., Van de Velde W., Wielinga B.: “Knowledge engineering and management. The CommonKADS methodology” MIT Press, 2000.
3. Puerta A.R., Tu S.W., Musen M.A.: “Modeling Tasks with Mechanisms”. International Journal of Intelligent Systems, vol 8, 1993.
4. Newell A.: "The Knowledge Level" In Artificial Intelligence Vol 18 pp 87-127.
5. Cuena J., Molina M.: “The role of knowledge modelling techniques in software development: a general approach based on a knowledge management tool” International Journal of Human-Computer Studies. No. 52. pp 385-421. Academic Press, 2000.
6. Molina M., Cuena J.: “Using Knowledge Modelling Tools for Agent-based Systems: The Experience of KSM” in “Knowledge Engineering and Agents Technologies” Cuena J., Demazeau Y., García-Serrano A., Treur J. (eds.) IOS Press, 2001 (in press).
7. Molina M. “An Intelligent Sales Assistant for Configurable Products” Proc. of The First Asia-Pacific Conference on Web Intelligence Maebashi (Japón). Lecture Notes in Artificial Intelligence series, Springer Verlag. 2001.
8. Manning G.L., Reece B.L.: “Selling Today”. Prentice-Hall, Inc. A Simon & Schuster Company. ISBN 0-205-16446-3. 1997.
9. Brown D., Chandrasekaran B.: “Design Problem-solving: Knowledge Structures and Control Strategies”, Morgan Kaufman, 1989.

Arms Race within Information Ecosystems[1]

Bengt Carlsson and Rune Gustavsson

Blekinge Institute of Technology, 371 25 Ronneby, Sweden
{bengt.carlsson, rune.gustavsson}@bth.se

Abstract. Interacting agents of exploiters and users within an information ecosystem may be regarded both as biological beings and as part of an economic system of infohabitants. A protection system can be implementing as a filter governing the access to assets. Typically we will have a chain of attacks and countermeasures concerning this access to the desired assets. We model this process as an arms race. We base our model on microeconomics and a process model of a protection system based on exposure time. A user's reaction against an exploiter measure could either be a direct response to the measure or trying to anticipate future attacks by more general means of defeating the protection of the exploiter agent. When anticipating future attacks and countermeasures, both users and exploiters will improve their methods and tools due to an arms race. Our arms race model refines the competition as modeled in computational markets to model aspects which typically arise when societies grows beyond what can be controlled in a centralized manner. A dynamic, evolving and robust ecosystem of autonomous agents is a preferred and possible outcome of the arms race.

1 Background

Information ecosystems are often modeled as societies of agents, that is, Multi Agent Systems (MAS). A key aspect of MAS is coordination of tasks and activities. Distribution of subtasks and coordination of team activities has been a focus of MAS and DAI (Distributed AI) R&D since early 1980's [3,8]. Since the end of 1980's R&D in MAS has also addressed larger sets, societies, of agents. Issues such as articulation work, team formation and contracts as well as societal aspects of obligations, norms and social order are topics of present day research agendas [4,6, 10]. Of specific concerns are issues of resource allocations.

With computations markets we can model certain aspects of coordination in agent societies. The introduction of a common exchangeable commodity of 'money' and an efficient mechanism for settling the market value has turned out to be very fruitful. However, the market mechanism does not cover all aspects of society behavior related to coordination. In large and open societies as anticipated in information ecosystems and witnessed on the web we have to address other aspects as well. In order to model explicit non-friendly activities that can be seen on the web today such as tampering,

[1] This work is partly supported by the European Project ALFEBIITE (A Logical Framework for Ethical Behavior between Infohabitants in the Information Trading Economy of the Universal Information Ecosystem): IST-1999-10298.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 202–207, 2001.

cyber crime, and parasitic behavior, we introduce a *arms-race model of coordination* based on a dynamic model of security and an evolutionary model [2,7,11].

Our arms-race model refines the competition as modeled in computational markets to model aspects which typically arise when societies grows beyond what can be controlled in a centralized manner and coordination also has to include protective or 'hardening' mechanisms for the survival of 'computational species'.

Time is an important factor when adapting a successful behavior. Fitness or money as a measurement of estimated resources may be seen as another way of describing the amount of time needed. The purpose for doing this reduction of an economic system is to emphasis on conflict resolution within information ecosystem.

Within a developed economy there will be an exchange of value between infohabitants. A "fair" exchange, a use-value U, based on calculating the proper amount of resources needed for the exchangeable good can be settled. But nothing prevents infohabitants to be more selfish than just fair in their actions. As a matter of fact, this extra incentive is a necessarily ingredient in an evolving economic system. Let us distinguish two groups of infohabitants; the exploiter and the user agents, both groups controlled by humans.

The reason for an exploiter to manipulate a user is to get a surplus value, or profit, S. Instead of the ideal use-value U there is a higher total price T, its exchange-value, expressed as:

$$T = U + S\,. \tag{1}$$

It is in the exploiters interest to make S as big as possible. We claim that within information ecosystem the dynamics between exploiters and users can be modeled as an arms race focusing on the protection and access of assets. An improvement of the protection by an exploiter starts a counter-measure by users or other exploiters. Because of the increased competition between the antagonistic groups and because of the autonomous individuals within a group, an accelerating arms race may also improve the overall robustness of the ecosystem. Within biology arms race is often used to explain the origin of robust ecosystems. The success of a single infohabitant may be favored by coalitions and vigilance against new intruders.

In an information ecosystem the exploiter agents may be a useful part of a manufacturer production line making extra profit, or belong to humans with malicious intentions. These different roles of an exploiter agent are sometimes hard to separate. Malicious intentions include agents creating viruses or cause information trespass. Some users may regard a company having agents that send spam mails or spywares, as malicious. Response agents' form on the user side acts as a defense against exploiters and their agents.

Together these different agent societies constitute the participants in an arms race. Advanced tools for both exploiting and creating responsive actions by the different societies will be developed. In the ecosystem it will be hard to predestinate which tool that will be used by one or another agent. Exploiters and users evolve a dynamic fashion based on arms race within a refined set of exploiter and defense agents. The success will be determined by the reaction time against a certain attack by an agent and the strength of the attack versus the strength of the protection system. We will, in the next section outline a generic model of arms race.

2 A Model of Arms Race

We begin by defining a generic computational model of Arms Race based on microeconomics. Based on this model, we outline an Arms Race model based on dynamics of a protection system.

Our model is based on balancing supply-demand of an asset (good) in a market. In micro-economy we can calculate (e.g., by auctions) the ideal price, use-value U above, of a good given the supply and demand. However, an exploiter can take advantage of different constraints or policies to ask for and get a higher price (adding extra profit or generating surplus value S) in accordance with eq. (1). above. Some well-known surplus enablers are: copyrights and patents, distribution restrictions, regulations, and added value. These surplus value mechanisms can be enforced in a computational Ecosystem by implementing 'filtering mechanisms' protecting assets by controlling access. In this paper we focus on a suitable dynamic protection model of access control.

An exploiter wants to protect surplus enabling mechanisms to his advantage, or to weaken those of his competitor. A user wants to get rid of those extra costs of getting the good. A user can thus legally or illegally attack suitable surplus generating mechanisms. Other exploiters can also enforce a user's intentions in a hostile take-over attempt.

The protection of the assets is based in interactions between three components. These components are a protection mechanism P , a detection mechanism D , and a response mechanism R . An implementation of those components acts as a 'filter' protecting the access to the assets. The filter is engineered to meet a class of attacks, a , it has a certain strength s , which can be improved, and it is time dependent, t . Our model is based on a model, Time Based Security, in network security [9]. An extended model [5] gives a process model of general protection systems. Let ΔP denote the duration of protection of a system at an attack a starting at time t_0 and with protection strength s . Let ΔD denotes the time it take to detect an attack a during the given circumstances. Finally, let ΔR denote the time it takes to implement sufficient measures to eliminate the threats of the attack a after the attack has been detected. Eq. (2). captures the relations between those time intervals and introduces an exposure time $\Delta E(a, s, t_o, t)$, where $t > t_0$. The protection system is protecting the surplus granting mechanisms if and only if $\Delta E(a, s, t_o, t)$ is non-positive. If the exposure time $\Delta E(a, s, t_o, t)$ is positive the exploited part will have full control of the situation during that time interval. In all realistic situations $\Delta P(a, s, t_o, t) = 0$ for all t larger than a certain value.

$$\Delta E(a,s,t_0,t) = \Delta D(a,s,t_0,t) + \Delta R(a,s,t_0,t) - \Delta P(a,s,t_0,t) \tag{2}$$

The eq. (2). may be used for proposing a surplus value formula based on eq. (1). The behavior of antagonistic agents within information ecosystems is discussed in [1]. The idea behind linking together a protection formula with surplus values is the extra cost these exploiter agents causes the users.

We can now formulate the total profit or surplus value for the exploiter, given that the protection system will stand against all attacks during the time interval 0 to T, as eq. (3). below. The function f is non-negative and eventually zero (e.g., when copyrights or patents ends).

$$S = \int_0^T f(a,s,\tau)d\tau \tag{3}$$

Eq. (2). includes all kinds of protections. It could be a security attack sending viruses or denial of service messages, but as well an economic based attack sending spam mails or using spy-wares. All activities are supposed to involve some advantage, the surplus value, for the exploiter behind the protection system.

Arms race can be modeled as follows. If we have a successful attack, i.e., $\Delta E > 0$ for $t > t_1$, where t_1 denotes a point in time and $t_1 > t_0$, then the exploiting agent can estimate a loss L according to eq. (4). The function h is non-negative and includes not only estimated loss of surplus value but also loss of, e.g., reputation.

$$L(t) = \int_{t_1}^t h(a,s,\tau)d\tau \tag{4}$$

The exploiter can decide to take countermeasures to bring back the favorable situation of a non-positive ΔE at some later time than t_1. This decision will be based on eq. (4). and an assessment of potential future profits based on eq. (3). The positive countermeasure, from the exploiters point of view, is modeled as an increase in strength (from s to $s_1, s_1 > s$, in the formulas above). Hence after a new point in time $t_2, t_2 > t_1$, we will again have a non-positive ΔE for some time $t > t_2$.

The endurance F of the exploiter during the time interval [0, t_2] is captured by eq. (5). The functions g_i models the actual losses to the exploiter due to active users exploiting the breakthrough of the protection system. Other exploiters aiming at building up opportunities for generating surplus for them can facilitate the spread of the knowledge of the breakthrough (i.e., increase the cardinality of index I).

$$F(t_1,t_2) = \int_0^{t_1} f(a,s,\tau)d\tau - \sum_{i\in I}\int_{t_1}^{t_2} g_i(a,\tau)d\tau - C_{s_1} \tag{5}$$

The constant C_{s_1} denotes the cost of strengthening the system from s to s_1. The endurance, i.e., the value of $F(t_1, t_2)$ has to be positive for the exploiter to remain competitive.

An arms race can in our scenario be modeled as the sequence of change in sign of ΔE of eq. (2). This changing in sign induces a sequence of time points $\{t_0, t_1, .. \}$ and a sequence of strengths of the protection system $\{s, s_1, .. \}$. The eq. (3)., eq. (4)., and eq. (5). can be re-interpreted accordingly.

The success of the arms race for the different combatants will depend on their resources and agility as well as of explicit or implicit coalitions as indicated in eq. (5). The success of a strengthening of the protection system will for instance depend on the detection and reaction times for the user agent. A prevented virus attack is one example of a successful protection system. Spam mails and spy-wares are hard to fully protect against because there does not exist a crucial situation like in the virus situation, i.e., the expected loss L of eq. (4). is not in general sufficient high to motivate an upgrading of the protection system.

The user's reaction against an exploiter measure could either be a direct response to the new measure or trying to anticipate future possibilities, including waiting for others to defeat the defense system (e.g., by using sniffing tools).

We can now state a robustness criterion. A robustness of the ecosystem, with respect to a protection system, of the ecosystem will emerge if and only if:

$$\forall a \in \mathrm{A} \,\&\, \forall t' \geq t, \exists s_0 : \Delta E(a, s_0, t') < 0 \qquad (6)$$

where A denotes all types of attacks and t, t' denotes points in time. If ΔE has small oscillations around zero the possibilities of reaching a robust state increases according to our formulas. It is the dynamics of the system, which influence the success of the infohabitants. Arms race will not destroy all intruders but make the system more robust, as in immunology. The lesson to learn is how to minimize the effect of intruders by finding almost stable robust states of the ecosystem.

3 Discussion and Summary

We propose a model of arms race between users and exploiters in a computational ecosystem. Exploiter agents try to, on the behalf of their human owners, make some extra profit (surplus) out of the ecosystem by exploiting user agents. The mechanisms generating surplus value is protected by a system implemented as a filter of access to the assets asked for. An arms race is modeled as a sequence of attacks and counter-measures strengthening the protection system. We have modeled the dynamics of the arms race by stating some basic formulas. The arms race might evolve in equilibrium of a stable robust system with respect to the protection mechanism under study.

Examples of protection systems include security and integrity systems in e-business. A typical attack in these systems includes spam mails. Spam mails direct advertising message to an unspecified group of receivers. A spam mail may look harmless compared to a virus. It does not cost the user any security harm or threats against stored data, but it costs time. A user agent filtering out spam mails costs processor time and delivered spam mails cost the user time for reading and cleaning up.

So, from a status quo perspective of the ecosystem, any changes involving some profit making is described as an attack, irrespective of if it is deleterious or evolve the information ecosystem.

There are two cases concerning the behavior of a protection system. A stable system or a dynamic system:

- A full secure stable system presupposes a total protection against every attack. The attacks must be detected and taken care of while the protection is still up. This ideal situation is rarely at hand. It could at best be attained as an emergent behavior.
- A dynamic system might have a strengthening process modeled as an arms race. An arms race presupposes some negative effects on the participants, but may not cause any immediate danger for the functionality of the system. A defense may include making anticipation against the intruder besides responding to the attack itself.

If all possible attacks are taken under consideration it is very unlikely that we can design a fully protected system. Instead we should expect an ecosystem based on arms race. The arms race is based on profit making activities. These activities make an improvement of the information ecosystem possible. A dynamic, evolving and robust ecosystem of autonomous agents is a preferred and possible outcome of the arms race.

The reported work is preliminary in many respects. We are at present evaluating our model in arms race scenarios based on Napster and other peer-to-peer ecosystems emerging on the Internet.

References

1. Carlsson, B. and Davidsson, P. A Biological View on Open Agent Systems to appear in IAT'2001, Maebashi City, Japan (2001)
2. Dawkins, R.: The Selfish Gene. 2nd edn. Oxford University Press, Oxford (1989)
3. Durfee, E., Lesser,V., and Corkill,D., Cooperative distributed problem solving, in ed. Barr,A., Cohen,P., and Feigenbaum, E., The Handbook of Artificial Intelligence, volume IV, p. 83-147, Addison Wesley (1989)
4. Durfee, E.H. Coordination of distributed problem solvers Kluwer (1988)
5. Gustavsson, R.: Security Issues and Power Line Communication. Invited Key address. In Proceedings of the Fifth International Symposium on Power Line Communication, (2001)
6. Jennings, N.R., Commitments and Conventions: The foundation of Coordination in Multi-Agent Systems, The Knowledge Engineering Review, 2 (3) p. 223-250, (1993)
7. Maynard Smith, J., Evolution and the theory of games, Cambridge Univ. Press, (1982)
8. Sandholm,T.W., and Lesser, V.R., Coalitions among computationally bounded agents, Artificial Intelligence, 94(1) p. 99-137 (1997)
9. Schwartau W. Time Based Security. Practical and Provable Methods to Protect Enterprise and Infrastructure, Networks and Nation Interpact Press (1999)
10. Wellman, M., A computational market model for distributed configuration design, Proc. 12th National Conference on Artificial Intelligence (AAAI-94) p. 401-407, Seattle, WA (1994)
11. Wilson, E.O.: Sociobiology - The abridged edition. Belknap Press, Cambridge (1980)

Information Agents: The Social Nature of Information and the Role of Trust

Cristiano Castelfranchi

Dept. of Communication Sciences, University of Siena, Italy
castelfranc@unisi.it
http://www.ip.rm.cnr.it/iamci/p/index-cc.htm

In IT information should not be considered as mere 'data', like empirical findings in the direct perception of the physical environment; but rather as **a social construct**. It is created and tailored on purpose by somebody for somebody else, it is believed or not, compared, transmitted and propagated through social relations and interactions. We will illustrate different facets of this social nature of information, with special attention to some trust issues.

- **Ontology**

The growing problem ontology should address is its intrinsically 'social' nature. Ontology exists to be shared, to be institutionally standardized or to be negotiated among the agents for mutual understanding, information exchange and collaboration. A common ontology creates a community of practice and a working community presupposes a common ontology. Or -at least- ontologies must be clear in order to have some efficient 'translation' system.

- **Access to Information**

Access to information is more and more explicitly a social process, not simply a technical process. In fact, the issue of "who should admitted to (provided with) which information, under which conditions, and playing which role" is becoming the central issue. Information is s*ocially restricted* for reasons of security, but also of business (what did you pay for?), of selection and relevance, of personalization, etc.

- **Relevance and Overload**

A serious problem in IT and especially on the web is the overload of information and its importance. Information must be selected not only for *limitations of cognitive processing*, or for excluding low quality information (not reliable because wrong or corrupted). Overload is also relative to importance. Information should be selected because -although correct and comprehensible- it can be quite irrelevant, unimportant. However, this criterion of selection is intrinsically relational (i.e. social) because the

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 208–210, 2001.

relevance of the information depends on context, on the user's Goals, and the user's Knowledge. A piece of information is *irrelevant for a given user* or agent because it is not useful for his needs and problems, or because is already known or marginal relative to his/her competence. Something is 'informative' if it adds something to my knowledge for my purposes.

- **Presentation and Tailoring**

Moreover, information should be frequently tailored and presented in different ways to different addressees, depending on their title and role, on their level of expertise, on their 'personality' or preferences, etc.

- **Search and Interaction**

Languages and protocols (among agents and between user and agents) are increasingly socially inspired (for example speech-act inspired); and especially between the user and its assistant we have true 'dialogue' systems. Moreover, the problem of interactivity and collaboration in H-Agent interaction become crucial. And this is a social model issue. In fact real collaboration presupposed the possibility of **over-help**: providing something different in order to satisfy the real need beyond the request. Over-help is based on user/agent modelling, on plan recognition, on goal-adoption and theory of mind, on initiative, autonomy and helpful attitudes (or personalities): All social issues.

- **Social Navigation**

In particular, it is very interesting to analyze the need for and the development of 'social navigation' in the web, as an answer to the problem of disorienting, of lack of confidence and trust, of selecting information, of building identities and community, and the possible role of agents in such a social navigation. By social navigation let's mean the fact that a navigating entity (user, agent) use as navigation and choice criteria in such huge information land footprints, traces and behaviors of previous navigators. And leaving traces can also be -either a functional or an intentional-*cooperative* behavior and a form of communication. More precisely, one should distinguish between:

- navigators who intentionally leave traces (thus, send messages: stigmergic or symbolic) to the others (for example, active reputation systems);
- navigators who simply leave traces which some entity (agents) organize for providing the others with the information about diffuse or typical behaviors (for example, hit parades and best-sellers);
- users or agents who individually observe and follow or catch information about previous behaviors and choices of others.

Also in this imitation or differentiation behavior trust is a crucial issue.

In this perspective **"Cooperative Information Agents"** is an enlightening title (and slogan). It perfectly focuses on the above mentioned issues of shared ontology, source reliability, collaboration with the user, collaboration among the agents; etc.

After a brief discussion of those issues, aimed at emphasizing the social nature of information, We will mainly examine the **theory of trust**, and compare our socio-cognitive approach with other approaches used in agents literature (economic and game-theoretic, computational, etc.). Specific attention will receive the application of trust to information and its sources which is a major problem on the web, where everybody can introduce -without any filter- the information s/he wants.

- **Information Credibility and Sources Trustworthiness**

Information is for being believed or at least used, but one must rely on it to decide to believe it or to use it. This is why trust is a crucial issue: in order to use information, agents must (either explicitly or implicitly) trust it. Our thesis is that *information credibility is strictly function of sources*: it depends on the kind of source, the credibility of the source (a matter of trust), the number of (converging) sources, the possible conflicts between different sources. Many of these sources are social ones; not perception, memory or internal reasoning but the observation of other agents or communication from other agents. Thus social trust (trust in other agents) becomes a crucial basis of information dependability.

We will apply a cognitive model of trust based on perceived safety, perceived competence and reliability to information sources. We will claim that

- this is only a specific application domain for a valid general theory of trust, and it does not require an *ad hoc* notion of trust;
- an explicit model of different aspects and dimensions of trust is necessary (and we use a belief-based model): one number or measurement is not enough;
- what matters in cyberspace is not simply trustworthiness and safety but *perceived* trustworthiness and safety, and the problem is how to improve them.

A Framework for the Exchange and Installation of Protocols in a Multi-agent System

Tadashige Iwao[1], Yuji Wada[1], Makoto Okada[1], and Makoto Amamiya[2]

[1] IP Server Project, Fujitsu Laboratories Ltd.
2-2-1 Momochihama, Sawara-ku, Fukuoka 814-8588, Japan
{iwao, wada, okadamkt}@flab.fujitsu.co.jp
[2] Graduate School of Information Science and Electrical Engineering,
Kyushu University
6-1 Kasuga-Koen, Kasuga, Fukuoka 816, Japan
amamiya@is.kyushu-u.ac.jp

Abstract. A new framework for exchange of protocols, called the Virtual Private Community (VPC) enables agents to exchange and install protocols dynamically by accepting policy packages that define protocols for use in sharing the processing of applications among agents. The agent's inner state, state-transition rules, and transitions are separated from each other in the VPC. Protocols are defined as rules for state transitions and the transitions are defined as policy packages. Agents which accept common policy packages form 'communities'. Agents are able to join communities by exchanging policy packages, to dynamically install protocols by evaluating policy packages, and to then communicate with each other through the communities. This paper also gives an example of the definition of protocols as policy packages.

1 Background

Multi-agent systems apply collaboration among agents to solve problems given from their users. One feature of multi-agent systems is the reconfiguration of collaboration among the agents to solve particular problems. To reconfigure collaboration among the agents, the agents need to change partners. They also need to change the protocols they use according to the partners with which they wish to communicate. Protocols are an important element of collaboration among agents. Agents are unable to communicate with each other if they do not share a common protocol. Hence, to allow changes in the configuration of collaboration among the agents of a multi-agent system, agents are given the ability to install protocols of services

In distributed systems, protocols are affixed to applications. Even in multi-agent systems, the protocols are, in many cases, built into the agents. The dynamic installation of protocols is thus prevented and the protocols for a given application are fixed. Vendors that join the 'communities' which are established by these organizations build the specified protocols into their systems. FIPA [1] and KQML [2], on the other hand, attempt to define their agent's protocols on

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 211–222, 2001.

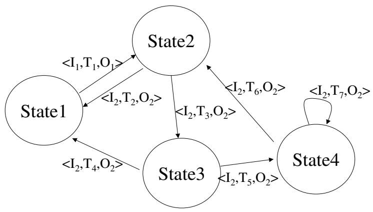


Fig. 1. State Transition

the basis of a model of the act of speech. Protocols in FIPA and KQML are abstract protocols that are independent of applications. For systems of agents that are built around FIPA or KQML, individual protocols are determined for use as application-level protocols during the design phase. The protocols used among agents are thus usually fixed to those initially set up for the individual applications provided by the agent system. To allow changes in the configurations of agent systems according to the problems that are encountered, a framework that allows the agents to dynamically install protocols is thus required.

We propose a new framework that allows agents to dynamically exchange and install protocols. The new framework is called the *Virtual Private Community* (VPC). The VPC provides functions according to which agents are able to install protocols and exchange protocols among themselves by using *policy packages* that define the protocols used among the agents in running a variety of applications. The VPC architecture allows developers to define protocols as policy packages. Policy packages separate the agent's inner state, its state transitions, and the rules for state transitions. Agents use policy packages to create communities, and can join new communities by receiving the corresponding policy packages. Agents are able to dynamically install protocols by evaluating policy packages, and to communicate with each other through the communities to which they belong.

Section 2 describes the relationship between an agent's inner state and a protocol. Section 3 describes a platform for VPC. Section 4 gives an example of collaboration among agents and a description of a policy package.

2 Protocols and Agents

The protocols that are used in transactions among agents are executed according to the inner state and state transitions of the individual agents. Agents communicate with each other by exchanging messages such as requests and responses. The inner states of agents determine the messages that they send in response to the messages which they have received. The inner state of an agent changes when it receives a message which triggers a state transition. The agent then sends messages that are the result of this transition.

Figure 1 is the state-transition diagram of an agent. The circles are states and the arrows are transitions. There are several transitions to, from, or within

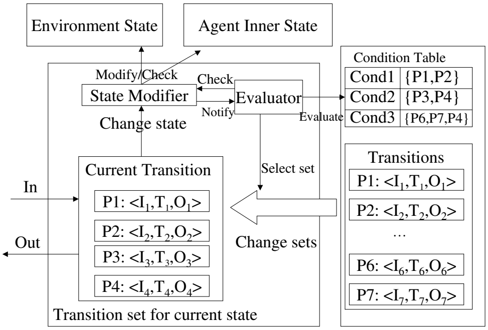


Fig. 2. Basic Model of VPC

each inner state. Each transition consists of the set of an acceptable message (I_n), a process (T_n), and a messge in response (O_n). An agent changes its inner state by going through the transition that corresponds to the received message.

Agents communicate with each other according to the protocols of applications. These applications' protocols are based on state-transition rules. The set of possible transitions for an agent changes, in accord with the state-transition rules, when the agent changes its own inner state. The protocols for the relevant applications are built into the agents since the state-transition rules and the agent's inner state are closely related.

An agent's inner state, its state transitions, and the rules for these state transitions should all be separate so that it is possible to dynamically change the protocols used among agents. To change a protocol is to change the rules for state transitions or the transitions themselves. An agent should also be able to introduce state-transition rules and transitions without regard to its own state at the time of introduction. The separation of the agent's inner state, its rules for state transitions, and the transitions themselves enables the use of agents which are designed without the inclusion of explicit protocols. Instead, the agents include the capacity to apply protocols. Platforms for multi-agent systems should thus provide mechanisms that allow the separation of an agent's inner state, its state transitions, and the rules for its state transitions.

3 The Virtual Private Community (VPC)

Our framework, the *Virtual Private Community* (VPC), allows the dynamic installation of protocols by agents. The VPC provides a mechanism for separating an agent's inner state, its state transitions, and the rules for these state transitions. Protocols are defined as *policy packages* that consist of state-transition rules, specifications of transitions (referred to as *roles*), and the data required in order to execute the protocol. All agents in a VPC platform have their own inner states. Agents communicate with each other through the communities that are created by agents which have accepted common policy packages. A service is offered by the interactions among the agents which form a community.

3.1 VPC Basic Model

Figure 2 shows the basic model of a VPC. The basic model separates the agent's inner state, transitions, and state-transition rules. The transitions and state-transition rules are defined outside the agents. The agent's inner state is not defined in terms of state labels. In the basic model, the agent's inner state is expressed as a set of values of variables. Agents replace their own sets of transitions according to the results of tests of the conditions in the condition tables. An agent uses its current set of transitions to react to messages, and sends back messages that correspond to the transitions it undergoes.

The main parts of VPC are a state modifier and an evaluator. The evaluator evaluates condition tables that consist of state-transition rules, deduces a set of transitions that is appropriate according to the state-transition rules and the agent's current state, and changes the current set of transitions to the deduced set of transitions. A state modifier changes agent's state according to requests for modification, and notifies the evaluator of all modifications of state. The state-transition rules take the form of pairs. Each pair consists of a condition and a set of the names of transitions that are to happen when that condition is fulfilled. It is possible for several conditions in the condition table to be simultaneously satisfied. In such a case, the transitions which correspond to the rules become active. Conditions are not only based on the agent's inner state, but may be based on tests of elements of the outside environment. Such elements include the states of outside processes and values in databases. A table of this type allows agents to append and remove rules dynamically. Agents are also able to modify rules. An agent's inner state and the state of the environment are persistent in that current states are maintained until they are modified. An agent's inner state consists of pairs of variables and values. Transitions don't have states. Instead, they consist of a set of an acceptable message, a process, and a response message. Transitions only react to messages that are acceptable. Various calculations are performed in processes of transitions. Processes also change the inner states of agents and the environment. The evaluator re-evaluates all state-transition rules when a state is changed. Then, new sets of transitions are selected for the new states.

In the basic model, protocols are changed by replacing condition tables and sets of transitions. Protocols are defined as a set of a condition table and definitions of transitions. Protocols in the VPC architecture are defined as policy packages. Agents in a VPC are able to install protocols dynamically by installing a condition table and sets of transitions from policy packages which are exchanged among the agents.

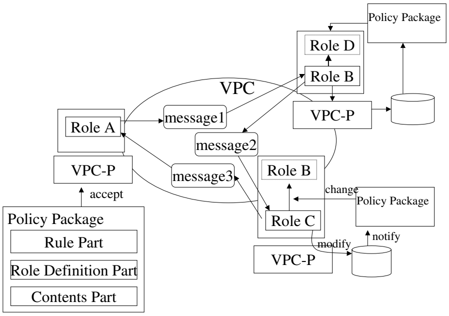


Fig. 3. VPC Overview

3.2 VPC Platform (VPC-P)

Figure 3 shows an overview of a VPC. A VPC platform (VPC-P) is an agent that is able to exchange policy packages with other such agents, and to use the policy packages thus gained to communicate with other agents. A community is made up of VPC-Ps that have accepted a common policy package. The VPC-Ps in a community provide a service by collaborating with each other. A VPC-P is also able to become a new member of an existing community by accepting the policy package that allows a VPC-P to operate as part of that community. A VPC-P parses policy packages and deduces its own role according to the rules of the policy packages as they apply to its state. Roles are not merely the rules for transitions but are also the means by which the agents in a given community carry out the functions of the corresponding service. A VPC allows the implementation of any role, such as database access, calculation, and the invocation of existing systems.

A VPC-P provides the functions that are necessary to parse policy packages, to manage the *attributes* that represent the agent's inner state and the state of its environment, and to determine roles on the basis of these attributes. A community consists of VPC-Ps which accept a common policy package. A community consists of its *sub-communities* on individual VPC-Ps. A sub-community is created on each VPC-P as an entity of a community. The elements of a community thus reside on the distributed VPC-Ps that have joined that community. VPC-Ps thus form communities by connecting with each sub-community.

A VPC-P parses the policies applied by the communities to which it belongs, deduces and assigns appropriate roles, and performs according to the roles they have been assigned within the community. A VPC-P consists of three parts: a profile-management part, a role-assignment part, and a role-collaboration part. The profile-management part is a state modifier. It manages the VPC-P's own inner state and has interfaces to modify the state of the VPC-P's environment. VPC state and environment are stored into databases. The role-assignment part is an evaluator. It determines its own roles from the policy and attributes. The role-collaboration part provides an event-driven method with messages as the events.

3.3 Policy Package

Figure 4 depicts the structure of policy packages. A policy package consists of a set of rules in the form of a table of conditions, a set of roles in the form of definitions of transitions, and content. Each rule consists of a condition and a set of role names. The conditions take the form of logical expressions with attributes as terms. Each attribute consists of the name of a database, the name of a variable, and the value of that variable. A role consists of its name, the name of a program, and the method by which the role is initialized. A package's content consists of the name of the content and the path to the actual content data. Content includes the program code that implements roles. A policy package is written in XML.

Policy packages are encoded as S/MIME [4] messages. S/MIME allows a VPC to detect any falsification of policy packages by checking the hash codes of the packages. When a VPC detects a falsified policy package, the VPC discards that policy package.

```
<policy package>   ::=   <rules> <roles> <contents>
<rules>            ::=   <rule> | <rule> <rules>
<rule>             ::=   <condition> <role names>
<role names>       ::=   <role name> | <role name> <role names>
<condition>        ::=   "TRUE"
                    |    "and" <condition> <condition>
                    |    "not" <condition>
                    |    "eq" <attribute>  |  "<" <attribute>
<attribute>        ::=   <db name> <variable name> <value>
<roles>            ::=   <role> | <role> <roles>
<role>             ::=   <role name> <program name> <init description>
<contents>         ::=   <content> | <content> <contents>
<content>          ::=   <content name> <content path>
```

Fig. 4. Structure of Policy Packages

3.4 State of the Environment

The profile-management part of a VPC-P has interfaces that allow it to modify the environment in accord with the VPC-P's roles. The environment is represented by databases. The state of the environment is stored as pairs, each consisting of a variable and its value, in the databases as the environment. The VPC-P's roles and its role-assignment part are able to access the data in the databases by passing a specification of the environment (the name of a database) and the name of a variable to the VPC-P's profile-management part.

The profile-management part evaluates expressions in terms of variables and values. A profile-management part has a set of evaluation modules, each of which corresponds to individual databases. It uses these evaluation modules to evaluate given expressions in a policy package. The profile-management part thus allows the evaluation of expressions in which types are available.

3.5 The Evaluation of Rules in the VPC-P's Policy Packages

The role-assignment part of a VPC-P deduces roles by evaluating the rules in the VPC-P's policy packages according to the VPC-P's state. The role-assignment part refers to the conditions of rules given in policy packages, applies the profile-management part to evaluate each term of each condition, and decides the appropriate roles for the current state. It then compares the list of currently active roles with the roles that it has determined as appropriate, installs the required roles that are not currently active in sub-communities of a VPC-P, and removes roles that are no longer required from the sub-communities of a VPC-P.

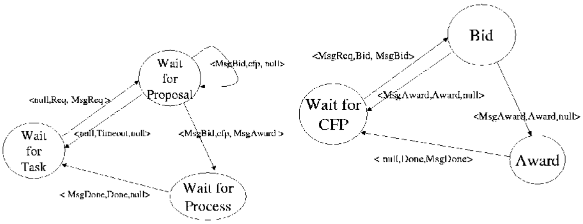
The role-assignment part requests that the profile-management part obtain the values of the terms of the condition for applying a rule of a policy package. The profile-management part accesses the rule's terms and then gets the values of the variables used in the terms from the environment. It then uses these values to evaluate the terms, and returns TRUE or FALSE as the result of evaluation. The role-assignment part combines these results to determine the validity or invalidity of the condition. The role-assignment part deduces appropriate role names by evaluating all conditions in the rule on the basis of the current values in the environment.

The role-assignment part gets, from the policy package, the role programs indicated by the fulfilled condition. Role programs are Java objects in VPC-Ps. The role-assignment part creates instances of role programs that are initialized as defined by the policy package, and adds these instances to sub-communities.

3.6 Protocols in Communities

The agents of a community collaborate by passing messages. Agents collaborate by passing messages. A message will be received by an agent that is playing a corresponding role. Our framework adopts the mechanism of collaboration among agents that play roles which was developed as the Field Reactor Model [5], that is, the so-called 'coordination model' [3], based on Dataflow Computing [6] is applied to coordinate agents playing related roles. The Field Reactor Model (FRM) provides a flexible method for collaboration among agents by applying pattern matching.

VPC-Ps are provided with a form of pattern-based invocation that is based on FRM. Patterns of roles in VPC-P correspond to patterns in FRM. Pattern-based invocation provides a method of invoking the functions of roles without having to specify signatures of functions or addresses of roles. When a message is sent into



(a) Manager (b) Contractor

Fig. 5. State Transition of Contract Net Protocol

a community, the VPC-Ps automatically invoke the appropriate functions with pattern matching. Values returned from the functions then enter the community as messages. Collaboration among roles takes the form of message-passing along chains of VPC-Ps. Roles do not have to be invoked by specifying their addresses. The VPC-P allows the user to define patterns and roles.

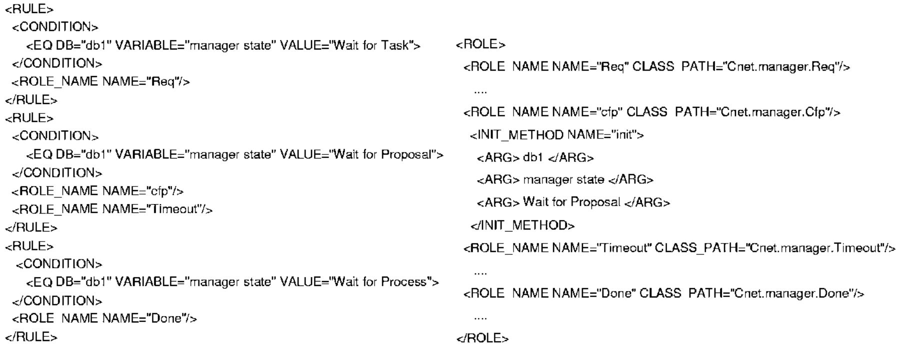
Roles are calculated in the following way. $P = \{p_1, \ldots, p_m\}$ is a set of patterns that is used in a community C. Roles $R_1, \ldots, R_n$ are played by the agents of community $C (C = \{R_1, \ldots, R_n\})$. Each role R_k is expressed as $R_k =< I_k, f_k, O_k >$. f_k is a function that has pattern $I_k (\in P)$ as its argument and returns a pattern $O_k (\in P)$ in response to $I_k (\in P)$. The pattern matcher $M(m)$ checks whether a message m matches the pattern of function f, and if it does match, the matcher invokes function f. The function f then returns messages as its values. This process is described as $M(m) = \{\cup_{i=1}^{n} O_k | I_k = m\}$. Thus the computation carried out by the roles played in the community C proceeds in the form $M(m), \forall m_n \in M(m_{n-1})$. The community C repeats this process until full collaboration among the roles has been established. This method of collaboration is similar to those used in models of coordination such as Linda [7].

Patterns of roles are patterns of object types. A VPC-P allows any Java objects to act as roles and messages. VPC-Ps provide a form of invocation on type matching in which the methods of objects are invoked with a pattern of types that matches the pattern of types in the arguments of the methods in the communities. The return values of the methods are put back into the communities when those return values are not null. The methods of roles are merely declared as normal methods.

4 Application

This section describes an example of VPC operating according to the contract net protocol [9]. The example provides a description of policy packages. These examples are currently working on VPC-Ps running in standard Java VMs and KVMs [8] under PalmOS.

In the contract net protocol, a manager sends a call for proposal (CFP) to several contractors and determines which contractors have placed bids that satisfy the CFP. Figure 5 shows the state transitions of a manager and contractors.



```
<RULE>
 <CONDITION>
   <EQ DB="db1" VARIABLE="manager state" VALUE="Wait for Task">
 </CONDITION>
 <ROLE_NAME NAME="Req"/>
</RULE>
<RULE>
 <CONDITION>
   <EQ DB="db1" VARIABLE="manager state" VALUE="Wait for Proposal">
 </CONDITION>
 <ROLE_NAME NAME="cfp"/>
 <ROLE_NAME NAME="Timeout"/>
</RULE>
<RULE>
  <CONDITION>
   <EQ DB="db1" VARIABLE="manager state" VALUE="Wait for Process">
 </CONDITION>
 <ROLE  NAME NAME="Done"/>
</RULE>
```

(a) Rule part

```
<ROLE>
 <ROLE  NAME NAME="Req" CLASS  PATH="Cnet.manager.Req"/>
   ....
 <ROLE  NAME NAME="cfp" CLASS  PATH="Cnet.manager.Cfp"/>
  <INIT_METHOD NAME="init">
   <ARG> db1 </ARG>
   <ARG> manager state </ARG>
   <ARG> Wait for Proposal </ARG>
  </INIT_METHOD>
 <ROLE_NAME NAME="Timeout" CLASS_PATH="Cnet.manager.Timeout"/>
   ....
 <ROLE  NAME NAME="Done" CLASS  PATH="Cnet.manager.Done"/>
   ....
</ROLE>
```

(b) Role part

Fig. 6. Contract Net Protocol for Manager

The initial state of a manager is “wait for task”. The state of the manager changes to “wait for proposal” after it has sent a “MsgReq” message as a CFP. This happens after the manager has received a request for a task from a user. The manager waits for proposals until the contractors have placed bids that satisfy the requirements of the CFP, or until time is up. The state of the manager changes to ”wait for task” and notifies the user that there have been no suitable proposals when the time is up. The state of the manager changes to “wait for processing” when proposals that satisfy the CFP have been submitted, and it sends a “MsgAward” message to the successful contractor. The manager's state then changes to ”wait for task” until it receives a “MsgDone” message that means that the task is finished. The manager finally returns the result of the task to the user. The initial state of the contractors is “wait for CFP”, in which the contractors wait for CFPs. The state of a contractor changes to ”bid” after it has sent a “MsgBid” message in response to a CFP. The contractor then waits for an award message. The states of contractors that have sent satisfactory bids change to the ”award” state, and no messages are sent to correspond to this transition. The contractors then execute the task, and send a “MsgDone” message to indicate completion of the task. A manager and contractors collaborate with each other via the above state transitions.

The policy package for the contract net protocol is packed into a single policy package for both managers and contractors. A given VPC-P is able to act as either a manager or a contractor. We give a more detailed description of policy packages here, based on the role of the managers. The rule part of the contract net protocol for a manager is as shown in figure (a) in 6. The variable “manager state” in database “db1” indicates. The rules for the manager state refer to the variable, evaluate it, and are applied if this evaluation is successful. Three rules correspond to the manager state; “Wait for Task”, “Wait for Proposal”, and “Wait for Process”. The role ”Req” that accepts requests from users is activated by the first rule when the manager is in the “Wait for Task” state. The second

```
public class Cfp {
  String db, key, value;
  ...
  public void init(String db, String key, String value) {
    this.db=db;
    this.key=key;
    this.value=value;
  }
  public MsgAward proposed(MsgBid bid) {
    if (!satisfy(bid)) return null;
    ProfileManager.setValue(db, key, value);
    MsgAward award=new MsgAward(bid);
    return award;
  }
   ...
}
```

Fig. 7. Sample Role Class

rule means that the roles “cfp” and “timeout” are active when the manager is in the “Wait for Proposal” state. The role “cfp” waits for proposals and the role “timeout” is the watchdog for the time-up condition. The third rule activates the role “Done” that waits for processing to be finished when the manager's state is “Wait for Process”.

Figure (b) in 6 illustrates the role-definition part for a manager. Each role is mapped to a Java object. Objects are created when the conditions of the rules have been satisfied, and are destroyed when the conditions are not satisfied. Role definition allows the initialization of objects by invoking specified methods with set arguments. “INIT_METHOD” is an initialization script. The object “Cnet.manager.Cfp” of role “cfp” is initialized by invoking the ”init” method with three arguments; “db1”, “manager state”, and “Wait for Proposal”. The object holds arguments that indicate its environment, and these database names, variable names, and values persist through transitions to subsequent states.

```
<CONTENT NAME="role programs" PATH="roles.jar"/>
      ...
```

Fig. 8. Contents Part of Contract Net Protocol

Figure 7 gives the class definition for a role “Cfp” as an example. Objects are created and initialized by the initialization descriptions of the role definitions when the corresponding rules have been satisfied. In the case of “Cfp” objects, the “init” method is invoked with the arguments “db1”, ”manager state”, and “Wait for Proposal”, according to the statements under the role part of Fig. 7. The object stores the given parameters in its member variables. The object is then added to a VPC-P of a community that is performing the contract net protocol. The method “proposed” is invoked with a put message when a message of type “MsgBid” enters the community. This method checks whether a proposal in the message satisfies the conditions of the request. If the proposal satisfies the conditions, the object changes the agent's inner state by reinitializing its parameters through the profile-management part of the VPC-P. The object then creates the messages that award the contract, and returns those messages. The VPC-P puts the values returned by its roles into communities. When a proposal does not satisfy the conditions, the object returns null. Null return values are discarded by the VPC-P. The object is discarded by changing the agent's state according to the above rule. The new object "Done" is also created according to the rule because the agent's inner state is "Wait for Process".

The content part of the policy package is shown in figure 8. The content part is a list of content names and the paths by which they are reached. VPC-Ps transfer these files along with policy packages when the VPC-Ps are exchanging policy packages. "Role programs" is a set of roles for use in a given policy package. "Roles.jar" is an "archive" of Java programs which includes roles such as "Cnet.manager.Cfp". VPC-Ps load objects from this file of programs.

VPC-Ps that are not equipped the policy package for the contract net protocol are able to interact with other VPC-Ps which are using the contract net protocol by accepting this policy package. The policy package includes the modules and state transitions required for participation in contract-net-protocol transactions. VPC-Ps get the policy package from other VPC-Ps that already reside in the community. The ability to play a part in contract-net-protocol transactions is installed in those VPC-Ps that get the policy package by their acceptance of the policy package. Such VPC-Ps then join the community and collaborate with the other VPC-Ps that are using the contract net protocol.

5 Related Work

FIPA and KQML include abstract-level protocols. The VPC-P approach allows the agents of agent-based systems built around FIPA and KQML to dynamically exchange and install protocols. VPC-Ps are able to use FIPA or KQML protocols to exchange application protocols.

AgenTalk [10] provides a way to describe protocols for agents. Description under AgenTalk covers the definition of the state transitions of agents in the form of a script. State names and action programs, which agents perform when they receive messages while in a corresponding state, are defined in the script. State definitions in the script include transition programs, and the state is defined as a label. Transitions of state and state-transition rules are not separated from each other. State definitions are sets of a transition of state and state-transition rules. Hence, AgenTalk does not allow agents to dynamically install transitions (message handlers). Also, the way scripts are handled in AgenTalk takes no account of the inner states of agents. Script entries in AgentTalk are merely labels. The VPC architecture separates transitions from states and state-transition rules, according to the current state. Hence, the VPC-P allows agents to dynamically install transitions for a single state by the exchange of transitions among agents with messages that are higher order. VPC gives AgenTalk a method for exchanging transitions.

6 Conclusion

A new framework for the dynamic exchange and installation of agent's protocols, called the Virtual Private Community (VPC), has been outlined in this paper. In the VPC architecture, an agent's inner state, state-transition rules, and transitions are separated. Protocols are defined as rules for state transitions and transitions are defined as policy packages that contain content which includes transition programs. The VPC allows agents to communicate with each other by dynamically installing policy packages. Agents which have accepted common policy packages form communities for collaborating with each other. Communities are created through the confederation of VPC platforms. Agents offer services by collaborating with each other through these communities.

We have developed two editions of VPC-P, a standard edition for enterprises and a compact edition for mobile devices. The standard edition manages a VPC for the many users who access it through Web browsers. The compact edition is for personal use. We are now testing these packages. The requirements for the standard edition of VPC are durability, robustness, security, and high-speeds of response, to allow large-scale application. The requirements for the compact edition are small memory consumption and a high degree of responsiveness. In particular, we must ensure security in order to protect products from illegal roles. We have also started to develop case tools for the design of policy packages in which a GUI is used to eliminate the need for direct writing of XML.

References

1. FIPA, The Foundation for Intelligent Physical Agents, http://www.fipa.org, 2001
2. Tim Finin, Yannis Labrou, and James Mayfield, in Jeff Bradshaw (Ed.), "Software Agents", MIT Press, Cambridge, to appear, (1997)
3. G. A. Papadopoulos and F. Arbab, "Coordination Models and Languages", Advances in Computers Vol. 46, pp 329-400, 1998
4. B. Ramsdell, Editor, "S/MIME Version 3 Message Specification", http://www.faqs.org/rfcs/rfc2633.html, 1999
5. Tadashige Iwao, Makoto Okada, Yuji Takada and Makoto Amamiya, "Flexible Multi-Agent Collaboration using Pattern Directed Message Collaboration of Field Reactor Model", LNAI 1733 pp.1-16, PRIMA'99
6. Amamaiya, M., Hasegawa, R.: Dataflow Computing and Eager and Lazy Evaluation, New Generation Computing, 2, pp.105-129, OHMSHA and Springer-Verlag (1984).
7. Carriero, N. and Gelernter, D, "Linda in Context", Communicaations of the ACM Vol. 32-4, pp 444-458, 1989
8. Sun Microsystems, "Java 2Platform Micro Edition", http://www.java.sun.com/j2me/?frontpage-javaplatform, 2000
9. R.G.Smith, "The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver", IEEE Transaction on Computers Vol C-29 No.12, pp 1104-1113, 1980
10. Kuwabara, K.: "Meta-Level Control of Coordination Protocols", Proc. Second International Conference on Multi-Agent Systems (ICMAS '96) pp. 165-172 (1996)

Ontology Negotiation as a Basis for Opportunistic Cooperation between Intelligent Information Agents

Sidney C. Bailin[1] and Walt Truszkowski[2]

[1] Knowledge Evolution, Inc.
1050 17th Street, NW, Suite 520
Washington, DC 20036 USA
sbailin@kevol.com
[2] NASA Goddard Space Flight Center
Information Systems Center / Code 580
Greenbelt, MD, 20771
USA
Walt.Truszkowski@gsfc.nasa.gov

Abstract. This paper describes an approach to *ontology negotiation* between agents supporting intelligent information management. Our objective is to increase the opportunities for "strange agents to communicate in solving tasks when they encounter each other on the web.

1 Introduction

Ontology negotiation is a means for agents to discover mutual ontology conflicts and then, though incremental interpretation, clarification, and explanation, establish a common basis for communicating with each other. We have developed an ontology negotiation protocol (ONP) and implemented it as a component that can be added to a variety of agents, irrespective of their underlying ontological assumptions. In this paper, we discuss the motivation for ontology negotiation, and the issues, both logical and practical, that arise in supporting it. Our emphasis in this paper is on the conceptual basis for the ONP. The mechanics of the ONP are described elsewhere [1]; our implementation is only a first step towards realizing the more comprehensive capabilities that we describe here.

Ontology negotiation is not directly addressed by current work in agent cooperation and socialization. Research into multi-agent systems does, however, address certain closely related phenomena. For example, there is considerable ongoing work on methods for learning models of other agents or other ontologies [6,10,11]. These efforts have not suggested a protocol for dynamic negotiation of ontology, however. There is a significant body of work on ontology conflict and sharing [2,5,7,8,9]. This work tends to focus on ways for distinct ontologies to be merged, joined, or otherwise modified to form a richer ontology. The type of interaction that we are investigating lies, in a sense, mid-way between these two areas. In a negotiation of ontologies, agents provide each other with information about themselves. This distinguishes the process from autonomous learning. We do not, however, assume that the proffered information is immediately meaningful to the receiving agent, or, if it is meaningful, that it has the same meaning as it does to the

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 223–228, 2001.

providing agent. Despite explicit information sharing between agents, there is a role for incremental growth in understanding.

2 Metaphors for Ontology Negotiation

In order to describe an ontology negotiation process, we have developed a couple of metaphors that suggest the kinds of dialogue that might occur. The first is that of a cocktail party; the second concerns information seeking in an office environment.

Agents in Hollywood: The Cocktail Party Metaphor. Consider a party attended by a multitude of agents, some of whom are strangers to each other. Imagine that two agents introduce themselves to each other and embark on a conversation. A frequent springboard for such conversations is the question, "What do you do?" In response, each agent would describe its job to the other. Depending on the level of mutual interest and the effectiveness of the communication, the conversation could progress to various levels of understanding. At the most superficial level, each agent can answer the other, and the other agent, treating the exchange as a simple ritual of courtesy, might take no notice. A deeper level of communication would occur if each agent tried to understand what the other agent does. A still deeper level would ensue if one of the participants intended to make practical use of that information. These distinctions can be expressed in terms of the progress of the conversation, with the deeper levels occurring as the dialogue progresses:

Stage 1: Talking distinct languages. Since we assume that the agents are *strange* to each other, they may not even have the concepts necessary to understand what the other one does. Suppose agent A_1 has described its job to agent A_2 in terms that A_2 does not understand. Three outcomes are possible:

- A_2 does not understand, and does not care. The conversation ends at this point.
- A_2 thinks it understands, although it does not. In this case, the process moves to the next stage, in which A_1 tries to clarify.
- A_2 does not understand, but wants to. The conversation proceeds to the next stage.

Stage 2: Guess and confirm or clarify. In the next stage, A_2 makes a guess as to what A_1 meant. This involves A_2 reformulating A_1's description in terms that A_2 understands. Three outcomes are again possible:

- A_1 understands A_2's reformulation, and confirms that it is correct.
- A_1 understands A_2's reformulation, but it is not correct. A_1 offers another description, trying to avoid the terms that A_2 evidently did not understand correctly.
- A_1 does not understand A_2's reformulation. The process may recursively enter the first stage again. This time, it is A_2's reformulation of A_1's description that requires clarification.

Stage 3: Take note of the other agent's self-description. If the conversation has arrived at this stage, then A_2 thinks he understands A_1's self-description. This perception may or may not be correct (note the first possible outcome of Stage 1), but there is not necessarily any immediate need to test it. A_2 may then make a note of A_1's self-description for possible future reference. In the cocktail party metaphor, this is the point at which the agents ask for each other's card. Since further cooperation is deferred until a later conversation, there is no need for the agents to try to reconcile or merge their respective ontologies at this point If the desire for future cooperation is strong, the agents might expend more effort ensuring that they understand each other correctly. This would prevent them from embarking down a blind alley in the future, e.g., if there really were no opportunity for cooperation. A_2 will then try to make full semantic sense of A_1's self-description. This entails placing A_1's self-description within A_2's ontology. If A_1's self-description involves concepts that are new to A_2, then A_2's ontology must now expand. This is likely to involve a back-and-forth conversation between the agents.

Back at the Office: The Information Retrieval Task. We now consider how the process might unfold when one agent tries to enlist the help of another in performing an information retrieval task. As a model of how the process might go, consider what happens in a purely human information access situation when the parties involved do not speak the same specialized language. The following series of steps illustrates a growing realization among the parties that there is a conceptual disconnect that must be resolved:

Stage 1: Respond to query with minimal effort. The agent whose help is requested tries to acquit himself of his obligation with as little effort as possible. He returns information that matches the query at a superficial level, but is not really what the requester wanted. This is a common phenomenon in human organizations. Under time and resource pressure, the person whose help is being sought — the *server* — will try to minimize the work required to satisfy the request. This entails spending a minimum amount of effort trying to understand the request. The server may quickly match terms in the request to a known category in his own conceptual organization. He will return whatever is found in that category, instead of trying to understand the requester' needs, goals, constraints, and any other context of the request. This is the state of practice in database systems, as well.

Stage 2: Express dissatisfaction with initial response. In a database query setting, the user who has received an inadequate response typically tries to reformulate the query. She could relax certain constraints and add others. Choosing a different set of terms is often a matter of guesswork. The requester can try to explain why the initial response was unsatisfactory: "This is not what I wanted. What I need is X, whereas what you gave me is Y." With a better understanding of what the requester wants, the server can try again. The server may or may not request further clarification. If not, the requester's needs may still not be adequately understood, and the process will repeat itself.

Stage 3: Enlist additional help. The improved understanding that the server now has may lead it to contact another server. If the second server returns any information to the original server, the original server will package it together with his own data, perhaps filtered according to the requester's previous expression of dissatisfaction.

Stage 4: Negotiate ontologies to construct integrated story. The first and second servers must now negotiate to construct an integrated response to the query. They may discover that their ontologies are quite different, despite their use of many of the same terms. They may discover, in particular, that there are differences in the conceptual structures used to organize their respective data.

Stage 5: Evolve ontologies. A successful negotiation between the two servers results in a shared understanding of concepts that were previously not held in common, or were understood differently. One way to do this is to explore the compositional history of a concept being negotiated. If the concept itself is not understood by the other agent, or not understood properly, an agent can try to define the concept in more basic terms, which stand a better chance of being shared. This process can be iterated until the agents find fine-grained concepts in common. The second agent can then use the common concepts in one of the following ways:

- Add the new concept to its ontology
- Find the term in its ontology that represents the same, or a similar, concept
- Respond with a different understanding of the same term

3 Ontology Negotiation Protocol (ONP)

The ONP is an attempt to translate the above-descried interaction patterns into a machine-processable protocol. The message types are: query, query result, acknowledgement, declination, rejection, capability statement, confirmation of interpretation, clarification, explanation of relevance, and explanation of fact. For example, *confirmation of interpretation* is an agent A's validation of an agent B's tentative understanding of a previous message from A. *Clarification* is the means by which an agent makes explicit the meaning of a previous message it sent.

The core of the protocol is the process of receiving a message, attempting to interpret it, requesting either clarification or a confirmation of the tentative interpretation, and resuming interpretation and further processing when the clarification or confirmation has been received. The recursive nature of this pattern allows this simple model to serve as a faithful rendering of the processes described in Section 3. In the remainder of the paper we focus on explaining (clarifying!) the forms that interpretation and clarification assume in the protocol.

Interpretation. Interpretation is the process of determining whether a message just received is properly understood. In the current implementation, messages are sequences of keywords, and the recipient agent tries to interpret each keyword in turn. First, the agent checks its own ontology to see whether the keyword occurs there. If not, the agent queries the WordNet lexical database to find synonyms of the keyword (Fellbaum, 1998). Then it checks the ontology for any of these synonyms. If a synonym is located in the ontology, it represents an interpretation of the keyword. Since WordNet may identify distinct meanings for the keyword (homonyms), each synonym is only a possible interpretation, which must be confirmed by the source of the message. The recipient agent therefore requests a *confirmation of interpretation* from the source agent.

WordNet plays a role in our protocol that could be played by any number of common upper ontologies. Obviously, if such common ontologies sufficed to mediate all agent interactions, the ONP would be unnecessary. The existence of the ONP is predicated on the judgment that for specialized concepts, such an assumption is unwarranted.

Clarification. If an agent is not able to interpret some of the keywords of a message it has received, it can decide to proceed anyway (if enough other keywords are understood), or it can request a *clarification* from the source of the message. Given the richness of its database, it would be tempting to invoke WordNet as the primary means of clarifying a keyword. This would be pointless, however, since the recipient agent has already gone to WordNet during the interpretation process. Instead, the message source draws on the following methods of clarification:

- Locating synonyms in the source agent's ontology
- Providing a complete set of specializations (keyword as the union of its subclasses) from the source's ontology
- Providing a weak generalization from the source's ontology
- Providing a definition in formal logic—in particular, defining the keyword as the conjunction of other keywords

The interfaces to these clarification methods are formulated abstractly in an application program interface (API) that we have defined along with the protocol. It is up to the ontology software to decide how to implement the interface. The API also specifies a set of functions for ontology evolution. Numerous issues arise in the process of integrating a new concept or interpretation with an existing ontology. For example, to what extent should a new meaning be qualified with the context in which it was acquired (e.g., identification of the other agent)? The API is intended to draw a clear boundary between the exploration of these issues, which are admittedly important, and the ONP itself.

4 Conclusion

We have argued the need for communication and cooperation between agents with different ontologies. Such situations arise in human cooperative problem solving, and we have identified certain patterns that humans draw on to navigate them. We have developed a protocol that models these patterns; in doing so, we have layered the underlying reasoning process so that an ontology need only support a well-defined API in order to operate within the framework. We have demonstrated the utility of the approach in a test bed that includes ontology proxies for NASA's Global Change Master Directory and NOAA's Wind and Sea Index.

References

[1] Bailin, S. and Truszkowski, W. Ontology Negotiation between Agents Supporting Intelligent Information Management. *Workshop on Ontology in Agent Systems* (OAS-2001), to be held in conjunction with the *Fifth Annual Conference on Autonomous Agents,* Montreal, Canada, May 28 – June 1, 2001.

[2] Dieng R. and Hug S. Comparison of "Personal Ontologies" Represented through Conceptual Graphs. In H.Prade ed, *Proc. of the 13th European Conference on Artificial Intelligence* (ECAI'98), Wiley & Sons, pp. 341-345, Brighton, UK, 1998.

[3] Fellbaum, C. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[4] Finin, T., Labrou, Y., and Mayfield, J. KQML as an Agent Communication Language, in Jeff Bradshaw (ed.), *Software Agents*, MIT Press, Cambridge, Available at http://www.cs.umbc.edu/kqml/papers. 1997.

[5] Heflin, J., Hendler, J., and Luke, S. Coping with Changing Ontologies in a Distributed Environment. In *AAAI-99 Workshop on Ontology Management*, AAAI/MIT Press, 1999.

[6] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. Building Domain-Specific Search Engines with Machine Learning Techniques. *Intelligent Agents in Cyberspace: Papers from the 1999 AAAI Symposium*, March 22 - 24, Stanford, CA. Technical Report SS-99-03, AAAI Press, Menlo Park, CA, 1999. Pages 28 – 39

[7] Visser, P.R.S., Jones, D.M., Bench-Capon, T.J.M. and Shave, M.J.R. An Analysis of Ontology Mismatches; Heterogeneity versus Interoperability, *AAAI Spring Symposium on Ontological Engineering*, Stanford University, California, USA. 1997.

[8] Visser, P.R.S., Jones, D.M., Bench-Capon, T.J.M. and Shave, M.J.R. Assessing Heterogeneity by Classifying Ontology Mismatches, in *Formal Ontology, in Information Systems*, edited by N. Guarino (Proceedings FOIS'98, Trento, Italy), IOS Press, Amsterdam, The Netherlands, pp. 148-162, 1998.

[9] Visser, P.R.S. and Cui, Z. Heterogeneous Ontology Structures for Distributed Architectures, *Workshop on Applications of Ontologies and Problem-solving Methods*, 13th European Conference on Artificial Intelligence (ECAI-98), Brighton, United Kingdom, 1998.

[10] Williams, A. and Ren, Z. Agents Teaching Agents to Share Meaning. Fifth International Conference on Autonomous Agents, Montreal, Canada, May 28 – June 1, ACM Press, pp. 465-472, 2001.

[11] Zang, D. and Sycara, K. Benefits of Learning in Negotiation. *AAAI 97*, pages 36 - 41.

A Mechanism for Temporal Reasoning by Collaborative Agents*

Meirav Hadad[1] and Sarit Kraus[1,2]

[1] Department of Mathematics and Computer Science
Bar Ilan University Ramat Gan 52900, Israel
[2] Institute for Advanced Computer Studies, University of Maryland
College Park, MD 20742

1 Introduction

This paper considers the problem of temporal scheduling of actions in a cooperative activity under time constraints. In order to carry out their cooperative activity the agents perform collaborative planning that includes processes that are responsible for identifying recipes, assigning actions to agents and determining the time of the actions. A recipe for an action consists of subactions which may be either *basic* actions or *complex* actions.

Basic actions are executable at will if appropriate situational and temporal conditions hold. A complex action, can be either a single-agent action or a multi-agent action. In order to perform a complex action the agents have to identify a recipe for it and it might know several recipes for an action. A recipe may include temporal constraints and precedence relations between its subactions. For the agents to have achieved a complete plan, the values of the time parameters of the actions that constitute their joint activity must be identified in such a way that all the appropriate constraints are satisfied. Determining the time of the actions in a collaborative environment that we consider is complex because of the need to coordinate actions of different agents, the partiality of the plans, the partial knowledge of other agents' activities and the environment, temporal and resource constraints. Furthermore, the temporal constraints requires interleaving of planning and execution. In this paper we present briefly algorithms for cooperative agents to determine the time of the actions that are required to perform their joint activity.

2 An Algorithm for Temporal Reasoning

Building a collaborative agent that can flexibly achieve its goals in changing environments requires a blending of Real-Time (RT) computing and Artificial Intelligence (AI) technologies. Accordingly, our single-agent system consists of an AI subsystem and an RT subsystem. The goal of the AI subsystems when the agents attempt to perform α is to identify a set of basic actions and a set

* This work is supported in part by NSF under Grant No. IIS-9907482.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 229–234, 2001.

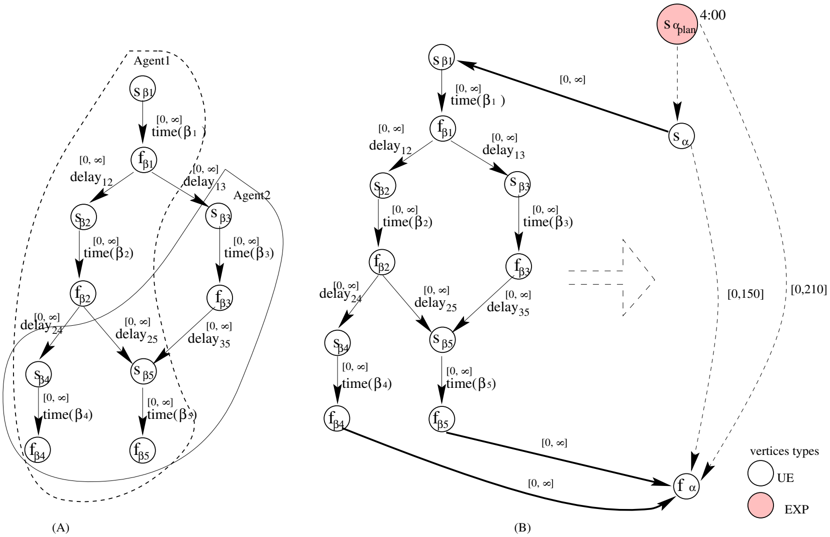


Fig. 1. (A) An example of $Gr^p_{R_\alpha}$. (B) An example of adding a precedence graph to Gr^i_α.

of the temporal constraints associated with the basic actions such that performing the basic actions under these temporal constraints constitutes performing α without conflict. As soon as it is identified, each basic action β is sent to the RT subsystem, along with its temporal requirements $\langle D_\beta, d_\beta, r_\beta, p_\beta \rangle$ where D_β is the **Duration time**, i.e., the time necessary for the agent to execute β without interruption; d_β denotes the **deadline**, i.e., the time before the β should be completed; r_β refers to the **release time**, i.e., the time at which β is ready for execution; p_β denotes the **predecessor actions**, i.e., the set $\{\beta_j|(1 \le j \le n)\}$ of basic actions whose execution must end before the beginning of β. In our system, the agents collaboratively develop the plan of their joint activity. Section 2.2 presents the major constituents of the time reasoning mechanism, which is based on the mechanism for an individual agent presented in [2]. However, expansion of collaborative plans differs from the individual case [1]. When an agent acts alone, it determines how to do an action (i.e., find a recipe) and then carries out the necessary subactions. When agents work together, both the formation or selection of the recipe and the carrying out of the subactions is distributed. The group must reach a consensus on how they are going to perform the action (i.e., on the recipe they will use) and on who is going to do the subactions. When each agent (or subgroup) decides on when to perform a subaction the temporal constraints of the group members must be taken into consideration. Thus, recipe and agent selection in collaborative planning require inter-agent negotiation and communication as well as individual agent reconciliation. In the mechanism presented, for simplicity, we assume that each action is associated with the agent that performed it. Thus, there is no need for decision making concerning the

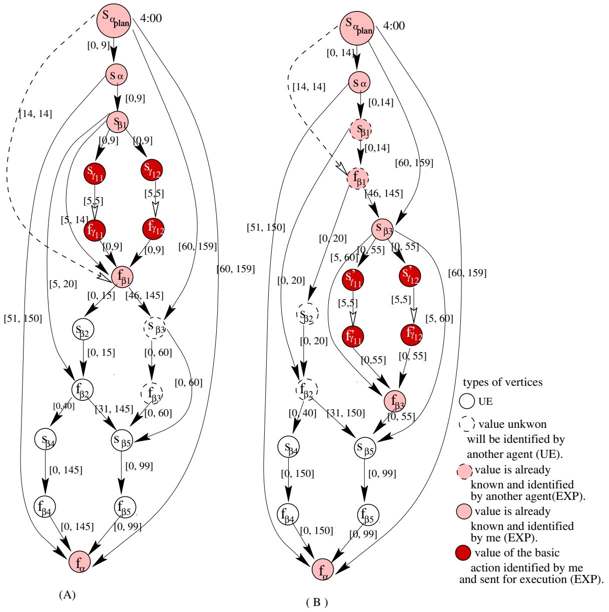


Fig. 2. An example of temporal graphs Gr^i_α which are maintained by A_1 (graph A) and A_2 (graph B) during the planning of the joint activity α.

performer of the action. We focus on the time scheduling problem and briefly describe some of its components. More details about the single-agent aspects of the system can be found in [2].

2.1 The Temporal Constraints Graph

The main two structures that are used when the agents identify the time parameters of their joint activity α is a precedence graph and a temporal constraints graph. A precedence graph $Gr^p_{R_\alpha}$ with respect to recipe R_α represents the precedence constraints of R_α (see Figure 1(A)).

The temporal constraints graph associated with a multi-agent action α maintains information about the temporal constraints of the actions that constitute the performance of α and the precedence relations among them. Formally, a temporal constraints graph, $Gr^i_\alpha = (V, E)$, is $V = \{s_\alpha, s_{\beta_1}, \cdots, s_{\beta_n}, f_\alpha, f_{\beta_1}, \cdots, f_{\beta_n}\}$ $\cup \{s_{\alpha_{plan}}\}$ and $\{\alpha, \beta_1, \cdots, \beta_n\}$, represents complex and basic actions that the agent intends to perform in order to execute the highest level action α (see Figure 2). The variables s_y and f_y represent the time points at which the execution of an action $y \in \{\alpha, \beta_1, \cdots, \beta_n\}$ can start and must finish, respectively. Some of the vertices may be fixed, i.e., these vertices denote a known time which cannot be modified. The vertex $s_{\alpha_{plan}}$ represents the time point that an agent A starts to plan the action α; it is a fixed vertex. Other fixed vertices may be initiated, for example, by a request from a collaborator. The activity of action y is represented by a directed edge between s_y and f_y, that is labeled by the time duration required for the execution of action y. A directed edge between the finish time point of action y, to the start time point of another action $z \in \{\alpha, \beta_1, \cdots, \beta_n\}$ denotes that the execution of z cannot start until the execution of y is completed. The interval associated with this edge indicates the possible delay between these actions. A metric constraint $a_{i,j} \leq (v_i - v_j) \leq b_{i,j}$ between two different time points $v_i, v_j \in V$ is represented by the metric edge (v_i, v_j) that is labeled $[a_{i,j}, b_{i,j}]$. One of the main purposes of the algorithm is to determine the values of s_y and f_y for each task.

In addition to the temporal information, a temporal constraints graph maintains additional information on each of the graph's actions: (a) whether the action is basic or complex; (b) whether the complex action is a multi-agent or a single agent action; (c) whether a plan for the action has been completed; (d) the agent(s) that is(are) assigned to perform the action, and (e) whether the action has been already executed by the agent(s).

This information is determined incrementally by the algorithm which also expands the graph. Each agent A_i, in the system runs the algorithm in order to build its temporal constraints graph Gr^i_α. The graphs of individual agents may be different with respect to individual actions, but similar with respect to the first level of multi-agent actions.

2.2 The Algorithm in General

During the agents' planning process, each agent A_i selects an action β from its Gr^i_α. For this action, A_i is the sole performer or A_i participates in its performance and the agents agree that A_i will develop the plan for β. Then, A_i attempts to complete β's plan. Action β is selected by A_i only if the planning of all the actions which precede β have been completed. If β is a basic action that has to be performed by A_i, it is sent to A_i's RT subsystem for execution. Otherwise, A_i performs the following major activities: (1) Finds an appropriate recipe, R_β, for the complex action, β. (2) Constructs the precedence graph $Gr^p_{R_\beta}$ for β and incorporates it along with the other temporal constraints of R_β in Gr^i_α. (3) It checks the consistency of the updated temporal graph. If Gr^i_α is not consistent it searches for a different recipe for β, R'_β, and replaces the information it added in 2 with the information associated with R'_β and checks consistency again. If such R_β does not exist it backtracks. (4) Determines the new values of the vertices in Gr^i_α by using an algorithm such as the Floyd-Warshall algorithm for solving a temporal constraints problem (see [2]). (5) Obtains messages from the RT about the execution of the basic level actions and updates the timing of the appropriate actions. (6) Obtains messages from the other team members and updates the graph with respect to the information that is listed in the next section.

The development of a shared plan between the agents requires information exchange. Consider the case of two agents, A_1 and A_2, that intend to perform a joint activity α. They will exchange information in the following cases: (1) When they identify a recipe for their joint action or for any joint subaction in their plan. This exchange of messages may be about possible recipes and also may be part of their group decision making. (2) Agent A_1 may inform agent A_2 about the time values that it identified for its individual actions $\beta_1 \dots \beta_k$ when $\beta_1 \dots \beta_k$ *delay* the planning of action γ and γ has to be performed by A_2. We assert that $\beta_1 \dots \beta_k$ *delay* the planning of γ if they directly precede γ. (3) When agent A_1 completes the plan of a subaction in a recipe of a joint action with A_2, it informs A_2 that the plan for this action has been completed (but does not send the details of the plan). (4) If A_1 already sent information to A_2 about some action β, but it failed to perform β, then A_1 backtracks and informs A_2 about the failure of β or about new properties of their plan that were determined as a result of its backtracking.

2.3 Example

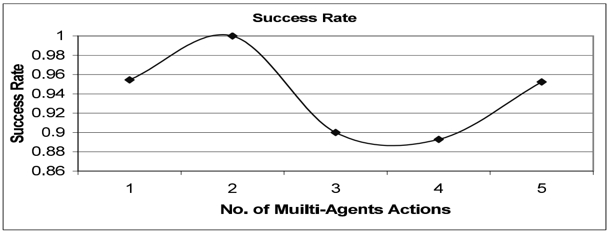
Suppose that agents A_1 and A_2 intend to perform a joint action α which is associated with the following constraints: { (finish-time $T_\alpha -$ start-time $T_\alpha \leq 150$), (start-time T_α after 4:00), (finish-time T_α before 7:30) }, where T_α represents the time interval of the execution of action α and we assume that initially the length of this interval is unknown. We also assume that both agents start the collaborative plan for α together at 4pm, then $V_{init} = \{s_{\alpha_{plan}}, s_\alpha, f_\alpha\}$ and $E_{init} = \{(s_{\alpha_{plan}}, s_\alpha, [0, \infty]), (s_\alpha, f_\alpha, [0, 150]), (s_{\alpha_{plan}}, f_\alpha, [0, 210])\}$. After the determination of the new values of the vertices, the new intervals of the edges in the example will be $E_{init} = \{(s_{\alpha_{plan}}, s_\alpha, [0, 210]), (s_\alpha, f_\alpha, [0, 150]), (s_{\alpha_{plan}}, f_\alpha, [0, 210])\}$ (see the dashed edges in Figure 1). Next, they need to agree on a recipe for α. Suppose that the agents agreed on R_α which consists of the subactions $\{\beta_1 \dots \beta_5\}$ where β_1 and β_2 can be performed only by A_1, β_3 can be performed only by A_2 and β_4 and β_5 are multi-agent actions. In addition, the recipe is associated with the precedence relations $\{\beta_1$ before β_2; β_1 before β_3; β_2 before β_4; β_2 before β_5; β_3 before $\beta_5\}$ and the temporal constraints: {(finish-time $T_{\beta_2} -$ start-time $T_{\beta_1} \leq 20$), (start-time $T_{\beta_4} -$ finish-time $T_{\beta_2} \leq 40$), (finish-time $T_{\beta_5} -$ start-time $T_{\beta_3} \leq 60$), (start-time T_{β_3} after 5:00) }. Each agent builds the $Gr^p_{R_\alpha}$=(V^p, E^p) and incorporates it into Gr^i_α by adding directed edges from s_α to each vertex $v_{b_1}, \dots v_{b_m} \subseteq V^p$ with an indegree of 0 and by adding edges from each vertex $v_{e_1}, \dots v_{e_k} \subseteq V^p$ with an outdegree of 0 to f_α. A pictorial illustration of adding this $Gr^p_{R_\alpha}$ to Gr^i_α is given in Figure 1. $Gr^p_{R_\alpha}$ is incorporated into Gr^i_α by adding the edges $(s_\alpha, s_{\beta_1}), (f_{\beta_4}, f_\alpha), (f_{\beta_5}, f_\alpha)$. After this, the agents also update the temporal constraints that are associated with R_α.

Then, each agent A_i tries to continue the development of its Gr^i_α by selecting an action to be developed. In this stage, Gr^1_α is identical to Gr^2_α but only the action β_1 may be selected since it does not depend on any other action. Since A_1 is the single agent involved with β_1, A_2 has to wait until it will receive the values of the temporal parameters of β_1 from A_1. Suppose that the recipe that A_1 selected for β_1 consists of the basic actions γ_{11} and γ_{12} and the execution time of each of them is exactly 5 minutes. Thus, A_1 should identify $\langle D_{\gamma_{11}}, r_{\gamma_{11}}, d_{\gamma_{11}}, p_{\gamma_{11}} \rangle$ and $\langle D_{\gamma_{12}}, r_{\gamma_{12}}, d_{\gamma_{12}}, p_{\gamma_{12}} \rangle$ and send it to its RT subsystem. The decision regarding the exact time in which β_1 will be executed is determined by the RT subsystem.

In this example, we assume that the RT subsystem of A_1 decides to execute γ_{11} at 4:02 and γ_{12} at 4:09. Thus A_1 will inform A_2 that it intends to terminate the execution of β_1 at 4:14. Following this announcement, A_2 can start developing β_3. Similarly, A_1 can develop β_2. A_2 will use this information to develop the plan for β_3. In addition, both of them will add the edge $(s_{\alpha_{plan}}, f_{\beta_1})$ and will label this constraint on it (i.e., the weight of this edge will be $[14, 14]$). The temporal graph Gr^i_α which is maintained by each agent in this stage of their collaborative plan is given in Figure 2. As shown in this figure each agent maintains a different graph according to its plan. Graph (A) is the graph which is built by A_1 and graph (B) is built by A_2. The identical vertices represents the subactions which appear in the recipe of their collaborative action.

2.4 Experimental Results

We developed a simulation environment to test our algorithm. In our simulations we ran the algorithm on several different recipe libraries which were created randomly. Each recipe library includes at least one possible solution to the joint activity. We tested the effects of the number of multi-agents' actions in the recipe tree of α on the success of the system in an environment comprising 2 agents. We ran 150 experiments with randomly drawn parameters from ranges with high success rates of one agent. We revealed that the number of multi-agent actions that are explored when developing the plan for α does not influence the success rate of the agents which is between $90\% - 100\%$ (see the graph below). We can conclude that for the environments that we checked, the multi-agent activity does not make the temporal reasoning problem significantly more difficult.



References

1. B. Grosz and S. Kraus. Collaborative plans for complex group action. *AIJ*, 86, 1996.
2. M. Hadad, S. Kraus, Y. Gal, and R. Lin. Time reasoning for a collaborative planning agent in a dynamic environment. *Annals of Math. and AI*, 2001. www.cs.biu.ac.il/ sarit/articles.html.

Agent Coordination Infrastructures for Virtual Enterprises and Workflow Management

Alessandro Ricci[1], Enrico Denti[2], and Andrea Omicini[3]

[1] DEIS - Università di Bologna
Via Rasi e Spinelli, 176 47023 – Cesena (FC), Italy
aricci@deis.unibo.it
[2] DEIS - Università di Bologna
Viale Risorgimento, 2 40136 – Bologna, Italy
edenti@deis.unibo.it
[3] DEIS - Università di Bologna
Via Rasi e Spinelli, 176 47023 – Cesena (FC), Italy
emailaomicini@deis.unibo.it

Abstract. Today's B2B application scenarios call for the integration of heterogenous resources, services, and processes, as in the case of Virtual Enterprises (VE) and inter-organisational Workflow Management Systems (WfMS). Agent-based approaches have already proven to be suitable to deal with the complexity of such application environments. In this paper we first recognise VE and workflow management as agent coordination problems, then discuss how *objective coordination* – that is, coordination outside the agents – can help to model VE and WfMS. Finally, we show how an agent coordination infrastructure like TuCSoN can impact on the engineering of highly dynamic VE and WfMS, by discussing a simple case study.

1 VE, WfMS, and Agent Coordination

The agent paradigm [24] and multi-agent systems (MAS henceforth) are widely recognised as suitable abstractions to deal with complex application environments, especially when the openness and unpredictable dynamics of the environment make traditional approaches less effective. In particular, agent organisations seem to be the most promising candidates for the development of open digital markets, business process management [9,5] and virtual enterprise management [5]. In this context, a Virtual Enterprise (VE henceforth) is a temporary aggregation of autonomous and possibly heterogeneous enterprises, meant to provide the flexibility and adaptability to frequent changes that characterise the openness of the business scenarios. Correspondingly, technologies for VE development have to face strong requirements resulting from the need to integrate and coordinate distributed activities, which should communicate and cooperate despite their heterogeneity and the unpredictability of the environment.

When building the support for a VE, two main issues arise. The first one is the integration of the existing services and infrastructures of the VE's component

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 235–246, 2001.

enterprises – from passive information sources to services and business processes. A key problem, here, is heterogeneity, which comes out at different levels – mainly the technology, information, and process levels. The second issue concerns the development of the VE's distributed workflow management system (WfMS henceforth), which should both integrate the different business processes featured by the VE's component enterprises, and enable the specification, execution, and monitoring of the VE's specific business processes.

Since there are dependencies among autonomous and heterogeneous activities to be properly managed and governed in order to reach global system goals, the above issues can be understood as agent *coordination* problems [11]. In fact, activities can be easily conceived as agents, featuring tasks to be pursued autonomously and pro-actively, typically involving the interaction with other agents and services. Dependencies concern sharing and exchange of heterogeneous information resources, task assignments for VE business processes, as well as temporal and prerequisite constraints among heterogeneous activities.

Modelling workflow management in terms of *workflow engines* (workflow servers or coordinators, where coordination takes place) and *workflow participants*, as suggested by the standard approach, induces a clear separation between coordinating and coordinated activities. Accordingly, the interpretation of workflow processes as coordination processes suggests the adoption of *objective coordination* [20], which promotes, too, a clean separation between coordinated and coordinating activities. These approaches typically provide explicit abstractions where to embed the coordination laws, both to enable the interaction among agents and enforce the required coordination policies. Correspondingly, in principle, coordination abstractions can be exploited as workflow engines, embodying the workflow rules in terms of coordination laws: in this way, workflow rules can be encapsulated outside agents and superimposed on them (*prescriptive* coordination), so that agents need not to be aware of the global workflow. This uncoupling property is peculiar to objective coordination models, and does not apply to *subjective* coordination models, where the workflow rules are embedded inside agents, mixing coordinated and coordinating activities.

2 Infrastructures for VE and WfMS

Electronic Markets and Virtual Organisations are among the most relevant economical structures emerged from e-commerce development. A VE is a temporary aggregation of autonomous and independent enterprises connected through a network, brought together to deliver a product or service in response to the customer needs [18]. So, VEs typically mean to combine the core competences of several autonomous and heterogeneous enterprises in a new agile and flexible enterprise, addressing specific and original industrial business targets [18].

An infrastructure supporting VE management has to address two main issues: *(i)* the integration of selected heterogeneous resources provided by single participants, and *(ii)* the execution of VE specific business processes featuring their own dynamics, while exploiting at the same time as many existing resources, services, and processes from component enterprises as possible. Since such aspects involve a multiplicity of autonomous interacting entities and require the management of their interactions and inter-dependencies, they can both be seen as coordination issues [11].

Moreover, heterogeneity comes out at the technology, the information, and the process levels. At the technology level, different kind of software environments might be used – from the operating system to the middleware used as infrastructure for the distributed computing environment [12]. At the information level, information to be shared can easily differ for syntax, semantics, and ontology. At the process level, heterogeneous activities and business processes must be able to interact despite the differences in the nature of involved entities (humans, services, software agents of different computational paradigms) and in protocols and languages used for agent interaction.

So, the first requirement for a VE infrastructure is to promote the conceptual homogenisation of VE's resources, information, and activities. Acting as an organisational glue, a VE infrastructure should support consistent interactions with shared services as well as among processes, while minimising VE's impact on participants infrastructure [25]. In this context, a VE infrastructure should also provide abstractions and mechanisms for security, safety and privacy, so as to protect the VE from the outside world, while taking into account the mutual requirements of the VE's individual components. Therefore, an infrastructure for VE management should help to govern the dependencies characterising the execution of VE business processes, such as task assignments, prerequisite and temporal constraints [11], which require the development of a WfMS [8].

Workflow systems are among the most important models currently used by organisations to automate business processes while supporting their specification, execution, and monitoring. A business process is precisely a workflow, that is, a set of activities to be executed in some order, involving multiple collaborating entities in a distributed environment to accomplish a given task [8]. A WfMS is meant to improve the process throughput, promote a better use of resources, and enable efficient process tracking [19]. Moreover, since workflow applications are subject to frequent changes due to the business environments [4], flexibility and adaptability are key features to face the necessary dynamic evolution of coordination policies, and to suitably react to unpredicted situations [10]. In the VE case, WfMS also have to face the issue of distribution, having to coordinate (heterogeneous) activities spread over the network.

Furthermore, the requirement definition for a WfMS standard [19] explicitly states that a distributed workflow infrastructure should be based upon the principle of loose-coupling between workflow engines and workflow participants. As a result, an infrastructure for WfMS should provide support for [19]:

– communication among participants and engines, with no need to agree on point-to-point protocols;
– flexible and scalable workflow participation, so that participants be able to participate using traditional devices (such as desktop PC) as well as non-

traditional devices (such as thin clients, or PDA), requiring no *a priori* information on engines, nor any dedicated connections to the engines themlselves;
- disconnected operation, since workflow participants may be mobile and non-frequently connected to the network, as well as location-independent;
- traceability of the business process state [12].

To face such requirements, the VE infrastructure should enable a clean separation between the conceptual places where the workflow rules are specified (and enforced) and the workflow participants, uncoupling the coordination activity from the activities to be coordinated. Workflow rules should also be explicitly represented, as well as (possibly) dynamically inspectable and modifiable.

3 VE, WfMS, and Coordination in TuCSoN

TuCSoN is both a model and an infrastructure for the (objective) coordination of Internet agents, particularly suitable to mobile information agents [16]. In the following, we overview the TuCSoN model, then discuss how such a coordination-based approach could be fruitfully exploited in the context of VE's WfMS.

3.1 TuCSoN Overview

The TuCSoN coordination model is based on the notion of (logic) *tuple centre* [15], a Linda-like tuple space [6] empowered with the ability to define its behaviour in response to communication events according to the specific coordination needs.

It has been argued [1,17] that the openness and the wideness of the Internet naturally lead to conceive it as a multiplicity of independent environments (e.g. Internet nodes or administrative domain of nodes), and to design applications in terms of agents that explicitly locate and access resources in this environment. TuCSoN supports such an approach to application design by providing multiple, independent interaction spaces, called *tuple centres*, that abstract the role of the environment. (Mobile) agents access tuple centres by name, either locally in a transparent way, or globally on the Internet in a network-aware fashion [16].

A local interaction space can be used by agents to access the local resources of an environment and as an *agora* where to *meet* other agents and coordinate activities with them. For these reasons, tuple centres, as fully distributed interaction media, can be understood as *social abstractions* [7], which allow to constrain agent interactions explicitly and to enforce the coordination and cooperation activities that define the agent aggregation as a society. Interaction protocols can then be designed and distributed among agents and media, adopting the most adequate balance for the specific application goals. This implies:

- the enhancement of agent autonomy – i.e., agents can be designed focusing only to their own goals and tasks, disregarding dependencies with respect to other agents, and without having to track (open) environment evolution;

– the enactment of prescriptive coordination, constraining agent interactions so as to reflect sound behaviours according to the defined social goals;
– the enhancement of the decentralised nature of the multi-agent system – tuple centres are spread over the network, living in infrastructure nodes visited by rambling agents migrating from node to node. The topological nature of the TuCSoN global interaction space [16] makes it possible to deal with the typical issues of distributed systems, thus enforcing flexible security policies, workload allocation policies, fault tolerance policies.

3.2 TuCSoN as an Infrastructure for VE and WfMS

The problem of heterogeneity is addressed in TuCSoN both at the model and infrastructure levels. From the technology viewpoint, TuCSoN is developed in Java, thus guaranteeing portability among platforms. Its architecture is based on a light-weight core, designed to support different sorts of middleware – from sockets, Java RMI, and CORBA to proprietary solutions – thus minimising the impact on the hosting environments and enabling a wide range of hardware devices (desktops, PDA, etc) to exploit the infrastructure.

TuCSoN directly supports the integration between heterogeneous information sources, too: in fact, its tuple-based coordination model naturally supports uncoupled interactions, thanks to *generative communication* (communication data that survive communication acts) [6], which provides *agent uncoupling*, from both the *space* and *time* viewpoints. In this way, agents can interact needing neither to know each other nor where they are (space uncoupling), and independently of their contemporaneous existence (time uncoupling). These features are key aspects for a peer-to-peer interaction model between WfMS engines and participants, since they allow to go beyond point-to-point communication protocols, supporting location-independent, disconnected workflow participation.

Since tuple centres can be programmed so as to react to incoming/outgoing communication events, thus defining the coordination laws to rule agent interactions, TuCSoN tuple centres are more powerful coordination abstractions than tuple spaces. As discussed in [14], this allows logic tuple centres to act as intelligent information mediators, mapping knowledge to knowledge: VE participants are therefore no longer required to change their model of knowledge representation. Moreover, new participants can be easily integrated dynamically.

Tuple centres' dynamic behaviour specification is also the key property that allows TuCSoN to fully support heterogeneity at the process level and, more generally, to face the coordination of dynamic aspects inside the VE. In fact, this is how TuCSoN supports a form of *uncoupled*, [22], *objective* coordination [20], storing coordination rules outside the interacting entities, and thus effectively governing their interaction in a predictable way. This is a key aspect both to provide the clean separation between workflow participants and workflow engines, and to enforce prescriptive coordination – i.e., the two issues uneasy to be achieved by an infrastructure that supports subjective coordination only.

Coordination rules in TuCSoN are expressed as programs in the ReSpecT specification language [3], which is embodied within tuple centres: as a Turing equivalent language, ReSpecT ensures that any computable coordination rule can be expressed (and enforced). As a consequence, workflow rules can be defined as TuCSoN coordination laws, decoupled from the interacting agents, and embedded into the coordination abstractions, keeping workflow coordinators and workflow participants distinct at the design, development, and run times.

ReSpecT can be considered an assembly language for interaction, with strong formal basis, yet general and powerful enough to support the development of higher-level specification language, tuned for specific applications. For these reasons, future investigations are planned to design and implement classic workflow languages, including the visual ones, on top of ReSpecT.

Finally, coordination laws in tuple centres are dynamically inspectable. Since a ReSpecT behaviour specification is structured as a multi-set of logic tuples, an intelligent agent can in principle retrieve it and reason about the rules currently governing interaction – in our case, about the workflow rules. Moreover, since a tuple centre's behaviour specification can be changed dynamically, an intelligent agent could monitor the evolution of a VE, and pro-actively adapt the workflow rules so as to improve the VE behaviour and performance. This is how a TuCSoN-based WfMS can provide the flexibility and adaptability required to face rapid changes in business environments, as well as to react to unpredicted situations.

4 A Case Study

To clarify some of the ideas discussed in the previous Sections, we sketch a simple case study, where the TuCSoN coordination infrastructure is exploited to support a *virtual bookshop*, here taken as an example of virtual organisation.

The *virtual bookshop* is a Virtual Enterprise (VE) that aggregates several companies of different sorts to sell books through the Internet. In this scenario, we identify four basic roles: the *book seller* (who provides the books), the *carrier* (who delivers books from sellers to customers), the *interbank service* (who executes payment transactions), and the *Internet service provider* (the Web portal for customers). We consider two distinct cases: first, the purchase of a single book from a specific book seller, then, the purchase of a set of books from (possibly) different book sellers.

These two examples imply the specification and execution of classic workflow rules, managing a sequence of activities, an AND-splitting process (coordinating more activities executed in parallel, started at the same time) and an AND-joining process (requiring synchronisation of parallel activities). The above workflow scenarios involve the following agents:

– *interface agents*, responsible for collecting information about customers and orders. These agents also interact with customers during order execution, informing them about the order status and possible problems.
– *buyer agents*, responsible for ordering books from the booksellers and getting them ready for delivery.
– *delivery agents*, responsible for delivering the books to the customers, provided that the books are ready at the booksellers: to this end, they inform

Table 1. Tuple centre programming in the case of the purchase of a single book

```
% a new order is placed
reaction(out(new_order(Customer, [(Book,Seller)], Carrier)), (
    in_r(new_order(Customer, [(Book,Seller)], Carrier)),
    % generate a new transaction ID
    in_r(trans_id_counter(ID)), NextID is ID + 1, out_r(trans_id_counter(NextID)),
    out_r(order_info(ID, Customer, [(Book,Seller)], Carrier)),
    out_r(order_state(ID, ordering)),
    % update buyer activity working list
    out_r(working_list(buyer, ID, Book, Seller)) )).
% a book is ready at the seller's: execute dispatching
reaction(out(book_ready(ID)),(
    in_r(book_ready(ID)),
    in_r(order_state(ID, ordering)),
    out_r(order_state(ID, ready)),
    % update carrier activity working list
    rd(order_info(ID, Customer, [(Book,Seller)], Carrier)),
    out_r(working_list(carrier, ID, Carrier, [(Book, Seller, Customer)] )) )).
% the book has been collected from sender (seller)
reaction(out(book_dispatched(ID)),(
    in_r(book_dispatched(ID)),
    in_r(order_state(ID, ready)),
    out_r(order_state(ID, dispatching)) )).
% book delivered to customer: execute payment
reaction(out(book_delivered(ID)),(
    in_r(book_delivered(ID)),
    in_r(order_state(ID, dispatching)),
    out_r(order_state(ID, delivered)),
    % update payment activity working list
    out_r(working_list(payment, ID)) )).
```

the carrier of each new dispatching to do, and monitor the delivering activity until the book is in the customer's hands.
- *payment agents*, responsible for getting the payments done: these agents interact with an interbank service for transferring money from customers to the VE.

Generally, agents synchronise on the working lists related to their role. Working lists are frequently used in the workflow context, as the place where information about new activities are placed and observed by agents that work on them. In our examples, working lists are represented as set of tuples, managed by coordination laws. New tuples are created for each new activity: authorised agents consume those tuples and execute the corresponding activities.

Coordination is performed via the `vbs` tuple centre, where the tuple representing a book order is placed by the interface agent: Table 1 and Table 2 show the coordination laws involved in the case of the purchase of a single book and of a set of books, respectively.

First workflow: purchase of a single book. The business process starts when an interface agent places a tuple describing a new order in the `vbs` tuple centre at the Internet portal node, programmed as illustrated in Table 1. This action triggers the first reaction of Table 1: a new transaction ID is generated, the information about the order is saved (tuple `order_info`) and the state of the order is registered (tuple `order_state`). The order state may be either `ordering` (the book is not yet available at the book seller), `ready` (the book is ready at the book seller but not yet collected by the carrier), `dispatching` (the book has been collected by the carrier, but not yet delivered to the customer), `delivered` (the book has been delivered to the customer and the payment activity is going on), or `closed` (the payment activity has been successfully brought to end). The buyer working list is then updated, storing a new tuple carrying information about the new order: one of the available buyer agents that is currently observing the working list gets the tuple and execute the corresponding activity.

When the book is ready at the bookseller's, the buyer mobile agent comes back home and outputs a new tuple indicating the success of the activity (tuple `book_ready`). This event triggers the second reaction of Table 1, which updates the order state and the deliver working list with the next activity to do. Then, one of the delivery agents observing the working list gets the tuple with the information about the new delivering activity and executes it, going to the selected carrier's node and suitably interacting with the local services.

From the carrier's node, the delivery agent indicates the status of the delivering process by producing tuples in the `vbs` tuple centre of the node from which it has migrated: more precisely, it places the `book_dispatched(ID)` tuple when the book is picked up from the bookseller's, and the `book_delivered(ID)` tuple when the book is successfully delivered to the customer. These interactions trigger the third and fourth reactions, which respectively update the order status from `ready` to `dispatching` (third reaction), and from `dispatching` to `delivered`, also updating the payment working list (fourth reaction).

Second workflow: purchase of a set of books. Again, the business process starts when a new order tuple is placed: unlike the previous case, the tuple describes the purchase of a set of books, each from a specified bookseller (see Table 2). A set of orders are then placed in the buyer's working list (first three reactions in Table 2), which means that several buyer mobile agents will be working concurrently on the same order (one for each book), reflecting a branch of activities. For each concurrent activity, a `suborder_state` tuple describes the state of the specific sub-order.

The computation creating the sub-orders and updating the buyer activity working list is performed by the second and third reactions: both are triggered the first time by the first reaction. While there are sub-orders to be created, only the second reaction is successful, updates the working list, and triggers the two reactions again, by means of the `out_r(order_all(ID,L))`. Instead, when no more sub-orders have to be created, the second reaction fails, the third is successful and computation ends.

When a buyer agent communicates the success of a sub-order activity by placing a `book_ready` tuple, the fourth reaction is triggered, changing the sub-order state from `ordering` to `ready`, and placing a `check_order_complete` tuple. This triggers the fifth and sixth reaction in Table 2, which check the sub-order

Table 2. Tuple centre programming in the case of the purchase of a set of books

```
% new order is placed
reaction(out(new_order(Customer, BookList, Carrier)),(
    in_r(new_order(Customer, BookList, Carrier)),
    in_r(trans_id_counter(ID)), NextID is ID + 1, out_r(trans_id_counter(NextID)),
    out_r(order_info(ID, Customer, BookList, Carrier)),
    out_r(order_state(ID, ordering)),
    % executing parallel buying activities
    out_r(order_all(ID, BookList)), out_r(suborder_counter(ID, 0)) )).
% executing buying activity for each sub order
reaction(out_r(order_all(ID, [(Book,Seller)|L] )), (
    in_r(order_all(ID, _)),
    % execute sub-order activity with C as sub ID
    in_r(suborder_counter(ID, C)),
    out_r(suborder_state(ID, C, ordering)),
    out_r(working_list(buyer, ID, Book, Seller)),
    % iterate for next sub-order
    C1 is C+1, out_r(suborder_counter(ID, C1)),
    out_r(order_all(ID, L)) )).
% no more sub-order to process
reaction(out_r(order_all(ID,[])),(
    in_r(order_all(ID, [])), in_r(suborder_counter(ID, _)) )).

% a book is ready at a seller's
reaction(out(book_ready(ID, SubCode)),(
    in_r(book_ready(ID, SubCode)),
    % update sub-order state
    in_r(suborder_state(ID, SubCode, ordering)),
    out_r(suborder_state(ID, SubCode, ready)),
    out_r(check_order_complete(ID)) )).
% not all sub-orders have been completed
reaction(out_r(check_order_complete(ID)),(
    in_r(check_order_complete(ID)), rd_r(suborder_state(ID, _, ordering)) )).
% all sub-orders processed: prepare for dispatching activity
reaction(out_r(check_order_complete(ID)),(
    in_r(check_order_complete(ID)), no_r(suborder_state(ID, _, ordering)),
      ...
      <clean information about no-longer-needed sub-orders and
      prepare the order list for the dispatching activity:
      the structure of OrderList below is [(Book,Seller,Customer)|L]>
      ...
    out_r(working_list(carrier, ID, Carrier, OrderList)) )).
```

completion. When all the sub-orders have eventually been processed successfully (AND-join workflow rule), the sixth reaction is successful, order details are collected in a list used to update the deliver working list, and everything goes on as in the single book case.

4.1 Discussion

The above examples have shown how objective coordination abstractions can be effectively used to model and engineer the workflow engines, thus pointing out some of the benefits of the TuCSoN approach. First, it should be noted how a workflow engine may be naturally mapped onto a coordination abstraction able to encapsulate the workflow rules as coordination laws, as a tuple centre.

Table 1 and Table 2 show two examples of tuple centre behaviour specification, represented as multi-set of logic tuples (reactions) that can be easily inspected, elaborated and possibly changed dynamically by (intelligent, human or artificial)) agents. In the first example, for instance, swapping delivery and payment activities, so that books are paid before being delivered, could be obtained by simply reformulating the specification of Table 1 by adapting the first and last reactions, and adding an extra reaction to take care of delivery after payment (tuple `payment_ok(ID)`):

```
reaction(out(new_order(Customer, [(Book,Seller)], Carrier)), (
    ...
    out_r(order_info(ID, Customer, [(Book,Seller)], Carrier)),
    out_r(order_state(ID,payment)),
    out_r(working_list(payment,ID)) )).

reaction(out(payment_ok(ID)),(
    in_r(payment_ok(ID)),
    rd_r(order_info(ID,_,[(Book,Seller)],_)),
    out_r(working_list(buyer,ID,Book,Seller)) )).

reaction(out(book_delivered(ID)),(
    in_r(book_delivered(ID)),
    % change order status to closed
    in_r(order_state(ID,_)), out_r(order_state(ID,closed)) )).
```

Instead, in this paper we did not mean to address the issues of effectiveness of information representation (as tuples), optimisation of resource management, effectiveness of coordination algorithms and interaction protocols used, neither meant we to propose ReSpecT as a workflow language. In fact, despite its Turing-equivalence, ReSpecT can be considered an assembly language for ruling interactions – complete, general-purpose, but low-level. So, a perspective worth investigation is to express coordination laws in a higher-level language (tailored to the specific application domain, such as WPDL [23] or the language of WebWork [13] in the case of workflow) built on top of ReSpecT.

Moreover, it should be noted that the notion of tuple centre is not bound to any specific behaviour specification language [15]: so, in principle, the behaviour of tuple centres could be programmed in a high level workflow language, using workflow rules directly as coordination laws.

Examples also put in evidence the uncoupling between agents and the workflows where they are called to operate: agents retrieve and provide only the information concerning their tasks, in term of logic tuples, with no need to manage and know other information concerning workflow execution. So, changes to the logic of the workflow policies can be done without affecting agents. Moreover, the tuple centre coordination model allows prescriptive coordination to be enforced locally, without compromising agent pro-activeness: indeed, the model does promote agents autonomy, since agents need not to be aware of workflow details (which are outside the scope of their specific goals) in order to accomplish their tasks. Finally, examples emphasise that *cooperation* among agents can be obtained without the need of *negotiation*, since agents coordination does not require agents to be aware of other agents' activity.

5 Related Works and Conclusions

Virtual Enterprises and Workflow Management Systems call for easily deployable and flexible infrastructures, enabling the integration of heterogeneous resources and services, and promoting the development of VE business processes in terms of workflow rules involving the VE's component enterprises. A suitable agent coordination infrastructure, providing the abstractions and the run-time support to address heterogeneity of different sorts and to represent workflow rules as agent coordination laws, may well fit the needs of a VE management in a highly dynamic and unpredictable environment.

In this paper we suggested that VE and workflow management can be conceived as coordination problems, and discussed the requirements for VE infrastructures. Then, we introduced the TuCSoN model and technology for the coordination of Internet agents, and discussed how a coordination infrastructure like TuCSoN may support the design and development of VE's WfMS, both in principle and in a simple case study.

In [25], [5], and [21], mobile and intelligent agent technologies are used for VE management, yet without providing an explicit model for WfMS. In [10], AOODBMS technologies are exploited for WfMS development: however, this approach seems to be particularly suited to the context of a single enterprise and to close environments, with low degrees of heterogeneity.

In [2] and [12], VEs are represented using multi-agent systems, with an explicit model of WfMS. Basically, these approaches support business processes execution with subjective coordination, encapsulating social rules directly within agents. As discussed in this paper, we found objective coordination, as provided by TuCSoN and by many other coordination models [22], more suitable to the context of VE than subjective coordination, since objective coordination seems to provide a more effective support for dynamics, flexibility, heterogeneity, and process traceability.

References

1. G. Cabri, L. Leonardi, and F. Zambonelli. Mobile-agent coordination models for internet applications. *IEEE Computer*, 33(2):82–89, Feb. 2000.
2. P. K. Chrysanthis, T. Znati, S. Banerjee, and S.-K. Chang. Establishing virtual enterprises by means of mobile agents. In *Workshop on Research Issues in Data Engineering (RIDE 1999)*, pages 116–125. IEEE CS, Mar. 1999.
3. E. Denti, A. Natali, and A. Omicini. On the expressive power of a language for programming coordination media. In *1998 ACM Symposium on Applied Computing (SAC'98)*, pages 169–177, Atlanta (GA), 27 Feb. – 1 Mar. 1998. ACM.
4. C. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In *ACM Conference on Organizational Computing Systems*, pages 10–21, 1995.
5. K. Fischer, J. Muller, I. Heimig, and A.-W. Scheer. Intelligent agents in virtual enterprises. In *Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96)*, Apr. 1996.
6. D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, 1985.

7. M. Huhns and M. Singh. Social abstractions for information agents. In M. Klusch, editor, *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet.* Springer-Verlag, 1999.
8. S. Jablonski and C. Bussler. *Workflow-Management: Modelling Concepts, Architecture and Implementation.* International Thomson Computer Press, 1996.
9. N. R. Jennings, P. Farantin, P. Johnson, M.J. O'Brien, and M. Wiegand. Using intelligent agents to manage business processes. In *Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 96)*, Apr. 1996.
10. G. Kappel, S. Rausch-Scott, and W. Retschitzegger. A framework for workflow management systems based on objects, rules and roles. *ACM Computing Surveys*, 32(1es), 2000. Article no. 27.
11. T. Malone and K. Crowstone. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.
12. M. Merz, B. Liberman, and W. Lamersdorf. Using mobile agents to support interorganizational workflow-management. *Applied Artificial Intelligence*, 11(6):551–572, Sept. 1997.
13. J. A. Miller, D. Palaniswami, K. J. Sheth, Amit P. Kochut, and H. Singh. Webwork: METEOR$_2$'s web-based workflow management system. *Journal of Intelligent Information Systems*, 10(2):185–215, 1998.
14. A. Omicini. Hybrid coordination models for handling information exchange among Internet agents. In A. Bonarini, M. Colombetti, and P. L. Lanzi, editors, *AI*IA 2000 Workshop "Agenti intelligenti e Internet: teorie, strumenti e applicazioni"*, pages 1–4, Milano (Italy), 13 Sept. 2000.
15. A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 2001.
16. A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, Sept. 1999. Special Issue: Coordination Mechanisms for Web Agents.
17. G. Picco, A. Fuggetta, and G. Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5), May 1998.
18. A. P. Rocha and E. Oliveira. An electronic market architecture for the formation of virtual enterprises. In *PRO-VE 99 IFIP/PRODNET Conference on Infrastructures for Industrial Virtual Enterprises*, Oct. 1999.
19. P. Santanu, P. Edwin, and C. Jarir. Essential requirements for a workflow standard. In *Proceeding of OOPSLA 1997 – Business Object Workshop III.* ACM, Oct. 1997.
20. M. Schumacher. *Objective Coordination in Multi-Agent System Engineering – Design and Implementation*, volume 2039 of *LNAI.* Springer-Verlag, Apr. 2001.
21. W. Shen and D. H. Norrie. Implementing internet enabled virtual enterprises using collaborative agents. In *Infrastructures for Virtual Enterprises*, pages 343–352. Kluwer AcademicPublisher, 1999.
22. R. Tolksdorf. Models of coordination. In A. Omicini, R. Tolksdorf, and F. Zambonelli, editors, *Engineering Societies in the Agents World*, volume 1972 of *LNAI*, pages 78–92. Springer-Verlag, Dec. 2000.
23. WfMC. Interface 1: Process definition interchange. Technical Report WFMC-TC-1016-P, Workflow Management Coalition, Oct. 1999.
24. M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.
25. H. Wortmann, A. Aerts, N. Szirbik, D. Hammer, and J. Goossenaerts. On the design of a mobile agent web for supporting virtual enterprises. In *WET ICE 2000 - Web-based Infrastructures and Coordination Architectures for Collaborative Enterprises*, 2000.

Mobile Agents:
State of the Art and Research Opportunities

Gian Pietro Picco

Dipartimento di Elettronica e Informazione—Politecnico di Milano
Piazza Leonardo da Vinci 32, Milano, I-20133, Italy
picco@elet.polimi.it, http://www.elet.polimi.it/~picco

The notion of mobile agent, a software component that is able to migrate autonomously among the hosts of a distributed system, is enjoying wide popularity as a novel abstraction for structuring distributed applications.

A number of advantages are usually claimed for mobile agents, including the potential for a more efficient use of communication resources, improved fault tolerance, support for disconnected operations, and in general enhanced flexibility.

Mobile agent enthusiasts often wonder why, despite all these advantages, this concept has not yet made it into the mainstream of distributed systems. Their explanations often put the blame on the immaturity of the technology, especially as far as security is concerned. On the other hand, mobile agent critics often complain about the lack of evidence supporting the claimed advantages. Indeed, scientifically sound evaluations providing strong quantitative or qualitative support for mobile agents are extremely rare.

The advent of Java, and the ease by which a mobile agent system can be crafted in such language, paradoxically makes matters even worse, giving another argument to detractors of mobile agents. A recent census reports the existence of more than 70 mobile agent systems—a surprising anomaly for such a young research field. A closer look reveals how very few among these systems bear innovative ideas, while the rest are essentially stereotypical in nature. Even worse, many are designed in a way that disregards or even hampers many of the aforementioned advantages.

It is true that the research area concerned with mobile agents is still in its early stage, and hence a lot of issues still need to be tackled. By the same token, however, this research area is no longer in its infancy: a lot of work has been disseminated since the appearance of the term in 1994. Hence, we are at a point where it is meaningful to ask ourselves what we have accomplished, what opportunities we have missed, and what are meaningful directions in the future of this research area.

The goal of this talk is to contribute some reflections around these questions, with the ultimate intent to help reshape the agenda for mobile agent research.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, p. 247, 2001.

Enabling FIPA Agents on Small Devices

Giovanni Adorni[1], Federico Bergenti[2], Agostino Poggi[2], and Giovanni Rimassa[2]

[1] Dipartimento di Informatica, Sistemistica e Telematica, Università degli Studi di Genova,
via all'Opera Pia 13, 16145 Genova, Italy
Adorni@DIST.UniGE.IT

[2] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma,
Parco Area delle Scienze 181A, 43100 Parma, Italy
{Bergenti,Poggi,Rimassa}@CE.UniPR.IT

Abstract. The foreseen progress of the market of portable devices is demanding the realisation of increasingly complex applications for increasingly smaller and ubiquitous devices. This trend promotes the migration of technologies that were originally developed for desktop computers to mobile devices. This paper describes the current results of the LEAP project, a European-scale effort that aims at enabling FIPA agents on small devices. The LEAP project achieves its goal providing a FIPA-compliant agent platform that runs seamlessly on any Java-enabled device, from cellular phones to enterprise servers. This platform is naturally ubiquitous as it can be configured to run on any mobile or fixed device. The LEAP project has already released a first version of the platform to registered users for testing purposes and the second, and final, version will be released in open source by the end of this year. We tested the platform in a logistics scenario intended to provide mobile workers with "just-in-time information."

1 Introduction

Internet-based applications are becoming ubiquitous, extending their reach to the telephony world and to its base of several hundred million users. We believe that in the near future people will want to access the information they need from handy devices. The importance of information agents running seamlessly on any kind of device exploiting any kind of network will increase dramatically because they will become the means for accessing "just-in-time information." As such, our research on agent-based services takes into account the scenario of deploying agents on small and ubiquitous devices. The following is a list of requirements that such a scenario imposes to agents and multi-agent systems:

1. They will have to conform to international standards;
2. They will have to run seamlessly on devices ranging from cellular phones to enterprise servers, adapting their characteristics on the capabilities of the device;
3. They will have to integrate and exploit the services provided by both the fixed and the mobile networks.

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 248–257, 2001.

This paper describes the effort we spent in the last year in developing the basic tech nology needed to enable FIPA agents running seamlessly on mobile and fixed devices. This effort is framed in the *LEAP – Lightweight and Extensible Agent Platform* – project. This project is a European-scale attempt to provide a FIPA platform, called LEAP, to allow agents running seamlessly on any Java-enabled, connected device. The LEAP project is funded for 2.5 years by the European Commission and it involves research centres spread across Europe in: France (Motorola), Germany (Siemens and ADAC), Ireland (Broadcom), England (BT) and Italy (TILAB and Università degli Studi di Parma). LEAP will be developed in two releases: the first release is already available to registered users for testing purposes. It was tested in joint laboratory sessions and it provides a solid base for deploying agents on small devices like PDAs. The second version will be released in open source by the end of this year.

In order to meet its goal, the LEAP project decided to start the development of its platform from JADE [1], an existing FIPA platform sufficiently modular and scalable to support the downsizing process toward small devices. LEAP is a new kernel for JADE that allows running legacy JADE agents on small devices without any modification, provided that the device offers sufficient resources and processing power. When running on a device with no severe constraints on resources, LEAP provides the same functionality as the original kernel of JADE. The result of this development approach is a first release of LEAP that:

1. Runs seamlessly on desktop PCs and on handy devices with limited resources, such as Palm Pilots and emulated Java-enabled phones;
2. Adapts its functionality to the available resources in terms of memory, processing power, screen, etc.;
3. Exploits wired networks and guarantees connectivity to handy devices via wireless networks, like TCP/IP over GSM and IEEE 802.11 wireless LANs.

The rest of this paper describes LEAP and the LEAP project and it is organised as follows: in section 2 we describe in greater detail the aims and scope of the LEAP project. Section 3 shows the architectural design of the platform and gives some hints on its implementation. In section 4 we describe a logistics scenario that we used to try the current implementation of LEAP. Finally, section 5 outlines some conclusions and describes some future work.

2 Aims and Scope of the LEAP Project

Agents need resources to act and to communicate. In FIPA specifications, the run-time support providing such resources is the agent platform [platform]. Agents can run only in the scope of an agent platform providing the basic services to support interoperability: a means for sending and receiving messages and a means for finding agents, i.e., white pages and yellow pages. We do not request the platform to provide any support for concepts from agent-oriented software engineering [8] such as autonomy or service-level interoperability [2]. Basically, the platform is only meant to support the typed-message agent model [12].

Agents communicate explicitly sending messages and such messages may reach either agents within the same platform or agents on different platforms. This difference must be transparent to the developer and a fundamental characteristic of agent platforms is enabling this to support open societies where agents running on different platforms can join and leave dynamically. Platforms can be distributed across the nodes of a network and the transparency of communication allows using agents to deploy flexible distributed systems. In addition, one single agent platform can be physically distributed across the nodes of a network in terms of locations. The platform is still seen as a whole from agents, but it is composed of a set of locations distributed across the network. One privileged location, called front-end location, provides the interface between the platform and other platforms.

The distribution and cooperation of agents residing on different platforms implies the conformance to a standard. At the moment, only FIPA is producing specifications for agent platforms. These specifications provide an abstract architecture describing the services of an agent platform [4] as well as a specification for all aspects related to the lifecycle of agents and platforms [5]. FIPA has just released FIPA 2000 specifications [3] and the aspects of FIPA 2000 specifications relevant to the LEAP project are:

1. Agent management and agent communication: support for a variety of encoding and naming schemes, including support of agents in mobile contexts;
2. Nomadic computing: agent management and communication frameworks geared towards support of mobile communication networks.

At the moment, a number of FIPA platforms are available [1, 7, 9, 12, 13], but none of them is capable of running on small devices like cellular phones even if FIPA introduced the specification of how nomadic agents might interoperate with agents running on the fixed network [6]. The LEAP project is developing the enabling technology, i.e., the platform, for allowing the seamless deployment of agents to all Java-enabled devices from mobile phones to enterprise servers.

In addition to the platform, the LEAP project will develop three generic agent services designed to show the possibilities of agent technology in assisting people irrespective of physical location, time of day, day of week, etc. In particular, we address the needs for open infrastructures and services supporting mobile teams. The characteristics of these services are that they are able to:

1. Anticipate a user's information needs by accessing and customising information on the basis of the user's profile and location;
2. Provide access to collective information assets in the team by networking individuals with each other, based on their current needs;
3. Empower individuals to collectively coordinate activities, e.g., by trading jobs, automatically negotiating for work, and expressing personal preferences;
4. Anticipate a user's travel needs, providing guidance and time estimation so as to synchronise the movements of virtual teams working over vast geographic areas.

A specification and a design of such services based on the MESSAGE [10] methodology have already been released.

Figure 1 shows a coarse-grained diagram of the components involved in an application built on top of LEAP. The Java platform hides the differences among devices concerning the operating system and the network. LEAP provides the services for agents to execute and to communicate. The generic agent services implement a framework for the integration of the logics intended to realise the functionality of the application. Such services will be implemented during this year and they will be used to implement the applications for the field trials. Finally, the GUIs allow the user interacting with the agents and with the platform.

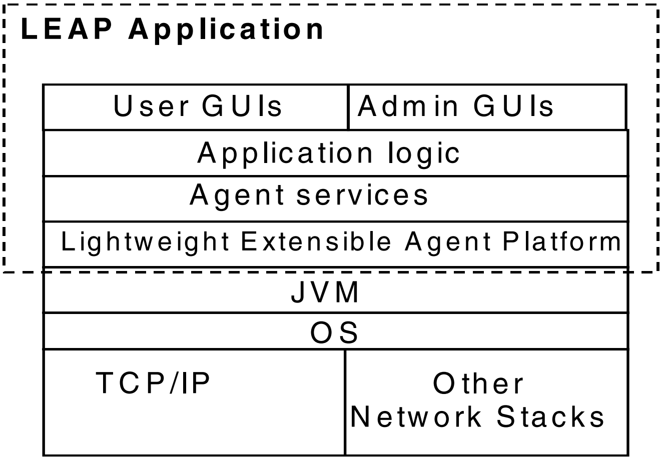


Fig. 1. Coarse-grained components of a typical application built on top of LEAP

3 Architectural Design of LEAP

The main goal of the LEAP project is to develop a FIPA-compliant agent platform sufficiently lightweight to execute on a mobile device, such as a cellular phone or a PDA, but also sufficiently powerful and open to be a first-class choice for enterprise servers. To meet this goal, LEAP can be deployed according to a set of profiles identifying the functionality available on each particular device. The basic profile supports only the functionality that FIPA requires and it suits the smallest device that we address, i.e., a cellular phone. The full-featured profile provides the functionality of an agent platform designed to run on desktop computers and it copes with any device with sufficient memory and processing power. The choice of implementing LEAP as a lightweight and extensible kernel for JADE allows using the services that JADE offers across all profiles. In the current implementation of LEAP, the basic profile provides the subset of the APIs of JADE that do not deal with the behaviour abstraction [1], and it does not integrate any run-time tool, such as the RMA or the Sniffer [1]. This allows saving memory and running LEAP on the current implementation of the KVM for Palm Pilots, which reserves only 200Kbytes of heap memory for Java applications. On the contrary, the full-featured profile integrates all tools that JADE provides and, from the developer point of view, it offers the same functionality and the same APIs of JADE. All profiles are instantiations of the FIPA abstract architecture and agents running on platforms configured with different profiles can interoperate. LEAP allows different locations of the same platform to be deployed according to different profiles.

The design of LEAP shares the basic principles of the design of JADE. This allows preserving the APIs and permits to any LEAP platform configured in the full-featured profile to run agents developed for JADE. Nevertheless, the implementation of LEAP is basically different from JADE one because the latter was not implemented taking into account the limitations of small devices. The introduction of the concept of profile in JADE required a substantial redesign and re-implementation of its internals but we succeeded in implementing it without changing its APIs and therefore agents developed for JADE can still run on LEAP. As a consequence, the community of JADE users will be able to run existing applications on small and wireless devices without any modification, provided that the device offers sufficient resources. Reasonably, such applications will need to spread the computation effort across the fixed infrastructure and the wireless devices. As applications are already decomposed into agents, the load balancing of the tasks is as simple as to allocating agents to network nodes on the fixed and the mobile network. These deployment issues are transparent to the developer as the agent platform is seen as a whole.

In order to make LEAP operating system and communication-protocol agnostic, we decided to base our development on the Java 2 platform. This allows accessing a common layer of platform-independent functionality available on mobile phones, PDA and PCs running all sorts of operating systems. Even if Java provides a solid foundation to build an application running seamlessly on different target devices, some device-specific customisations are necessary, as small devices do not provide a full-featured Java 2 platform. Considering the diversity of existing devices, Sun decided [14] to group Java platforms into three editions, each having its proper Java virtual machine, as shown in figure 2:

1. Java 2 Enterprise Edition (J2EE), intended and optimised for enterprise servers and mission-critical applications;
2. Java 2 Standard Edition (J2SE), intended for PCs and workstations and optimised for ordinary applications;
3. Java 2 Micro Edition (J2ME), intended for devices with limited resources, such as cellular phones and PDAs.

Moreover, J2ME introduces the notion of configuration. A configuration of J2ME specifies a subset of its Java virtual machine features and an associated subset of Java programming language features. As shown on figure 2, there are currently two configurations within J2ME:

1. Connected Device Configuration (CDC), for devices with memory and processing power comparable to a small PC;
2. Connected, Limited Device Configuration (CLDC), for connected devices with strict restrictions.

Most of the functionality in CLDC and CDC has been inherited from J2SE. In addition, CLDC and CDC introduce a number a features, not drawn from the J2SE, designed specifically to fit the needs of small devices. Such features refer mainly to display and communication.

LEAP is not only meant for small devices, it is also intended to support enterprise servers and we cannot impose constraints on its functionality when running on devices with no limitations on resources. To achieve this, we decided to go for the worst case, i.e., the CLDC, and we implemented an extensible architecture over a layer of adaptation capable of matching the classes available on this platform with the ones available on the other Java 2 platforms. This allows using the classes with maximum functionality where available and other classes with restricted functionality otherwise, without changing the source code of the agents.

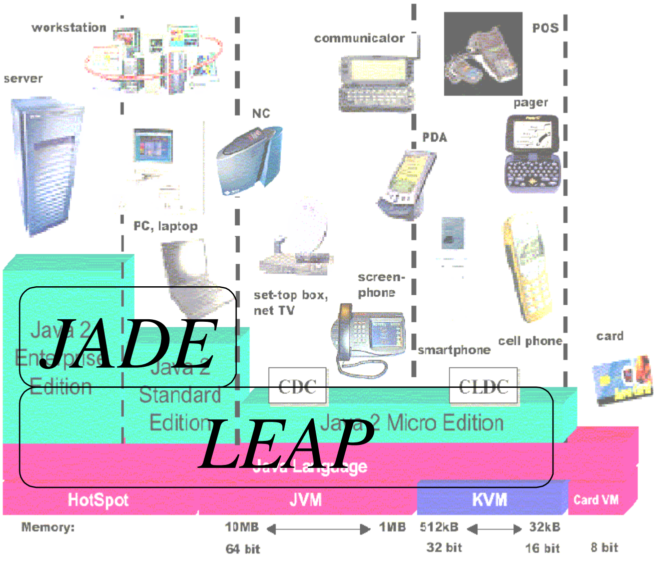


Fig. 2. Java editions and configurations and their relations with JADE and LEAP

LEAP is naturally distributed as it is composed of a number of locations that provide the run-time resources that agents need to execute. Mimicking the nomenclature of JADE, we say that locations contain agents and therefore we call them agent containers, or simply containers. We impose no restrictions on the way containers can be deployed across the nodes of the network, but the best way of deploying the platform is having one container running in one Java virtual machine for every node. It is worth noting that splitting the platform into several containers distributed on different devices is not mandatory. According to the context and the requirements of a given application, it is possible to concentrate the whole platform into a single container or to distribute it across the nodes of a network. As an example, if the application consists simply of a personal assistant supporting the user in managing the information on her PDA, probably the best deployment choice is to have a single-container platform running on the PDA. On the contrary, given an application where each user in a team is assisted by an agent running on her mobile phone and all such agents interoperate between each other and with a coordination centre, the choice of a distributed platform composed of one container running on an host connected to the Internet and several peripheral containers on the users' phones can be the best solution.

The nodes of the network can be of any kind, from mobile phones to enterprise servers, and they can access indifferently any transport mechanism for which a handler in the platform is available. At the moment, only the handler for TCP/IP is available, but this does not restrict to have only wired networks. This handler has been designed to support TCP/IP both over wired and wireless connections, such as GSM and IEEE 802.11, exploiting the possibility of using two-way connections.

The choice of spreading the platform across the network poses some problems for agents running on other platforms to communicate with agents running on LEAP. Following the design of JADE, we solved this problem introducing a privileged container, called main container, to allow agents on other platforms seeing the platform as a whole. The main container is unique and it acts as a front-end for the platform. It maintains platform-wide information and provides platform-wide services. Therefore, this container must be always reachable by all active containers in the platform. A LEAP platform is composed of a single main container and a set of peripheral containers, allowing a high modularity by running lightweight peripheral containers on small devices. The main container provides the FIPA mandatory services, i.e., the AMS and the DF, and it is mandatory to preserve FIPA compliancy. The amount of resources needed by the AMS and by the DF in the current implementation of LEAP suggests that the main container should run on a fixed network node with sufficient memory and processing power.

FIPA specifies a set of communication protocols that agents may use to exchange messages and it specifies how a gateway for matching such protocols must be implemented. Nevertheless, the first FIPA specifications did not allow many communication protocols, but imposed that all FIPA agents were addressed through a CORBA interface. Therefore, the first FIPA platforms relied on IIOP for communication between agents and, where applicable, this choice is still adopted. The main container of LEAP is supposed to run on a full-featured computer and we can reasonably suppose that we could find an implementation of IIOP there. This assumption is not valid for containers running on mobile devices and it may not be the best solution also for full-featured computers. For example, a container may run behind a firewall and direct IIOP communication could be impossible or it may require sending Java objects to the main container and Java RMI could be a better solution. Therefore, we need to support the communication between the main container and the rest of the platform by means of a flexible mechanism allowing the integration of different communication protocols. We call such protocols ITP – Internal Transport Protocol – and currently LEAP provides the following ITPs:

1. IIOP;
2. Java RMI;
3. Proprietary protocol over TCP/IP to support mobile devices equipped with a wireless connection.

FIPA introduces the AMS as the authority governing the agent platform and requires agents consulting it to be granted in lifecycle transitions. FIPA does not specify how agents should interact with the AMS for management operations because these activities are private to the platform and each platform is allowed to use its optimised techniques. The LEAP platform exploits the availability of many ITPs to support internal communication with the AMS. In particular, we use distributed-object technology where available, while we rely on our proprietary protocol where more sophisticated techniques cannot be applied.

4 Deploying LEAP in a Logistics Scenario

The possibility of running agents on mobile devices and allowing them to exchange messages through the public wireless network with other agents running on the Internet offers the possibility of using agents for “on-the-road” applications. This offers the opportunity of assisting users providing information only when and where they need it and it allows improving many location-aware business processes. Nomadic agents can provide location-aware services in direct contact with the user and they can also integrate such services with the services provided by the agents on the fixed network. This results in the possibility of dynamically creating services shaped on actual users’ need just when and where the need appears. A concrete example of an application scenario where this approach can show its benefits is logistics. In a typical logistics application, we have mobile workers in charge of moving goods in the territory and a service centre coordinating mobile workers. If we provide workers with a handy device running a personal-assistant agent capable of connecting to the service-centre agent, then the mobile worker can exploit the benefits of agents for its everyday job. We deployed this scenario in a demonstrator for an Italian project. This demonstrator is about mobile workers in charge of delivering Parmesan cheese across the neighbouring cities of Parma and Reggio Emilia. Each worker is equipped with a Compaq iPAQ H3000 palmtop running LEAP on PersonalJava. Such a device is linked to the Internet exploiting a Nokia Card Phone 2.0 to guarantee connectivity between the personal-assistant agents and the service-centre agent. The personal-assistant agent is in charge of keeping the information available to the worker synchronized with that of the service centre. To achieve this, we implemented the service-centre agent wrapping the service-centre database with an agent running on a LEAP platform deployed on the central server of the delivery system. The communication between the nomadic agents and the service-centre agent occurs to implement the following use cases:

1. The mobile worker has successfully delivered some Parmesan cheese to its final destination and he is about to go to the next location;
2. A change in the schedule of deliveries occurs. The central system must inform the mobile worker;
3. A delay in the schedule occurs because of a problem, e.g., a traffic jam or a flat tire. The mobile worker must inform the central system for a possible re-schedule.

The first use case allows keeping the information on the state of the delivery updated in real-time. The second and third use case allows recovering from possible faults in the business process thus saving time and money to the delivery company. The added value of the agent approach over more traditional technologies is mainly due to:

1. The personalisation that agents provide to the service;
2. The possibility of proactively querying the mobile worker for information if, for same reason, the state of the database seems inconsistent;
3. The possibility of negotiating a change in the schedule with the service centre on the basis of the current location of the mobile worker and of his preferences.

The first characteristic is important because the mobile worker can access a device with a tiny display and filtering the information from the service centre according to his actual needs is helpful. The second characteristic allows reducing the time lost before reacting to a fault in the schedule. If the personal-assistant agent or the service-centre agent believes that a delivery is late, it can proactively query the mobile worker. If the mobile worker simply forgot to update the state of the database, the agent can do it immediately. On the contrary, if some fault occurred and the delivery is still pending, the service-centre agent may decide to re-schedule all the deliveries. Finally, the third characteristic allows exploiting the expertise and the knowledge of the mobile workers to schedule a good plan for the deliveries. The service-centre agent schedules a plan and it proposes this plan to the workers. The workers are free to disagree to this plan if they believe their schedule could be improved on the basis of their current location and of their knowledge of the territory.

The use cases that we implemented clearly shows the centralised architecture that we used for this service, against its more natural peer-to-peer architecture of having only the nomadic agents negotiating possible changes in the schedule of the delivery. The choice of implementing a centralised service is mainly caused by the constraint of keeping the business processes of the delivery company untouched. This will not be the case of the scenarios that will be implemented for the field trials of the LEAP project. An analysis and a design of such trials based on the MESSAGE methodology are already available and they deal, for example, with peer-to-peer scheduling of meetings and overtimes.

5 Conclusions and Future Work

The impressive progress of the market of portable devices allows delegating more and more complex tasks to small, wireless and ubiquitous devices. This trend promotes the porting of technologies originally developed for PCs and workstations to mobile devices. This paper follows this trend describing the current results of the LEAP project. This project is intended to migrate agent technology to mobile devices through the development of a FIPA platform running seamlessly on any Java-enabled device, from cellular phones to enterprise servers. The project has released the first version of the platform and the second, and final, version will be released in open source by the end of the year. The work toward version two of the platform is intended to:

1. Optimise the current implementation with particular regards of the transport mechanism for wireless networks;
2. Improve the support for wireless connections exploiting the availability of protocols such as GPRS and Bluetooth;
3. Continue the downsizing of the platform to meet the requirements of real cellular phones;
4. Handle disconnections from the network more efficiently.

The products of the LEAP project will be validated in two field trials targeted at mobile engineers working in the telecommunications and roadside assistance industries. This will provide a unique insight into the practicalities and issues related to the industrial deployment and management of agent services on mobile devices. Additionally, the field trials will provide a means for investigating the social and usability aspects of agent services in an operational environment. These trials should clarify the importance of the results obtained by the project and should also show how agent technology could play a central role in the development of customer-oriented services integrating the possibilities offered by the fixed and the mobile networks.

Acknowledgement. The authors acknowledges the contribution made by all members of the Project LEAP consortium, without which the work described here would not have been possible. The European Commission funds this work through the grant IST-1999-10211. A.S. s.c.r.l. funds this work through the grant "Interventi di Ricerca & Sviluppo nel Settore della Logistica".

Disclaimer. The views expressed in this paper are the personal views of the authors. They should not be construed as representing the official view of the Project LEAP consortium.

References

1 F. Bellifemine, A. Poggi, and G. Rimassa. "Developing Multi-agent Systems with a FIPA-compliant Agent Framework", in Software – Practice and Experience, Vol. 31, pp. 103-128, 2001.

2 F. Bergenti and A. Poggi, "A Development Toolkit to Realize Autonomous and Interoperable Agents", in Proceedings of Autonomous Agents 2001, 2001.

3 FIPA "FIPA 2000 Specifications", available at http://www.fipa.org.

4 FIPA "FIPA Abstract Architecture Specification", available at http://www.fipa.org.

5 FIPA "FIPA Agent Management Specification", available at http://www.fipa.org.

6 FIPA "FIPA Wireless Message Transport Protocol Specification", available at http://www.fipa.org.

7 J. Heecheol, C. Petrie and M. R. Cutkosky. "JATLite: A Java Agent Infrastructure with Message Routing", IEEE Internet Computing, Mar./Apr., 2000.

8 C. A. Iglesias, M. Garijo and J. C. A. González, "Survey of Agent-Oriented Methodologies", in Proceedings of ATAL'98, 1998.

9 F. G. McCabe, "April – An Agent Programming Language for the Internet", available at http://www.nar.fujitsulabs.com.

10 MESSAGE Consortium, "Deliverable 1: Initial Methodology", deliverable of the EURESCOM Project P907-GI, 2000.

11 H. S. Nwana, D. T. Ndumu and L. C. Lee, "ZEUS: An advanced Toolkit for Engineering Distributed Multi-Agent Systems", in Proceedings of PAAM'98, London, 1998.

12 C. Petrie. "Agent-based Engineering, the Web, and Intelligence", IEEE Expert, Vol. 11, No. 6, 1996.

13 S. Poslad, P. Buckle and R. Hadingham, "The FIPA-OS Agent Platform: Open Source for Open Standards", available at http://fipa-os.sourceforge.net.

14 Sun Microsystems, "Java 2 Platform Micro Edition (J2ME) Technology for Creating Mobile Devices", available at http://www.java.sun.com.

Towards Efficient and Reliable Agent Communication in Wireless Environments

Heikki Helin and Mikko Laukkanen

Sonera Corporation
P.O.Box 970, FIN-00051 Sonera, Finland
{Heikki.J.Helin,Mikko.T.Laukkanen}@sonera.com

Abstract. We discuss agent communication in wireless environment according to FIPA's communication model, which can be seen as a communication stack. From the stack, we focus on the message transport protocols, message envelope layer and agent communication language (ACL) layer. The message transport protocol should provide both efficient and reliable service for the upper layers. We analyze the performance of bit-efficient FIPA-ACL in terms of the output size of an encoded message and the time needed to construct and parse an encoded message. The results show that this encoding is not only the most space-efficient but also proved to be efficient to deal with for instance in message parsing.

1 Introduction

Two interesting and important research fields are converging: nomadic computing and software agent technology. Nomadic computing enables accessing the fixed network services from virtually anywhere and at any time. Nomadic users are not tied to any particular location, time or terminal device, but they can use these services whenever needed and by whatever terminal device available. Software agent technology on the other hand has been considered suitable methodology for designing and implementing software for complex and unpredictable environments.

In wireless environments, the agent communication should be tailored in order to provide an efficient usage of scarce and fluctuating data communication resources. In some cases, efficiency is not so important aspect as for example reliability. Nevertheless, the communication solutions used in the modern distributed systems or the agent-based systems do not typically fulfill these requirements. In this paper we analyze the FIPA communication model (see Fig. 1) when employed in the wireless environment. Especially we concentrate on an efficient ACL communication and analyze the various options for concrete encoding of ACL messages that FIPA has specified. It is important to notice that low-level issues related to data transfer over the network is not itself an agent-related problem. However, without efficient and reliable delivery of data, implementing efficient and reliable communication—especially in wireless environments—is impossible.

The rest of this paper is organized as follows. Section 2 discusses message transport issues concentrating on reliability. In Section 3 we analyze FIPA-ACL

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 258–263, 2001.

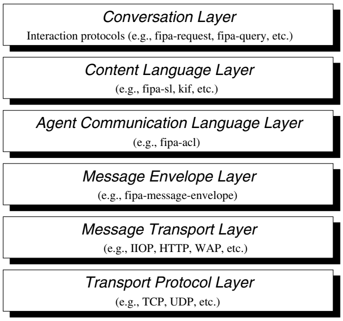

Conversation Layer
Interaction protocols (e.g., fipa-request, fipa-query, etc.)
Content Language Layer
(e.g., fipa-sl, kif, etc.)
Agent Communication Language Layer
(e.g., fipa-acl)
Message Envelope Layer
(e.g., fipa-message-envelope)
Message Transport Layer
(e.g., IIOP, HTTP, WAP, etc.)
Transport Protocol Layer
(e.g., TCP, UDP, etc.)

Fig. 1. Layered model of FIPA agent communication

messaging and provide results of extensive study on different concrete FIPA-ACL encodings. Finally, Section 4 concludes the paper.

2 Message Transport and Envelope Layers

TCP is the most widely used transport protocol in the Internet. In most cases, TCP is used indirectly. For example, the IIOP and HTTP protocols used in FIPA architecture use TCP as the transport protocol. It is well known that TCP performs poorly in slow wireless networks [6], and thus gives a poor basis for message transport protocol (MTP) employing it.

TCP is not the only option for transport protocol. FIPA has specified that WAP can be used as a baseline MTP in wireless environments. Should the WAP be used, it is up to the implementation and the environment, which protocol is used as the transport protocol. Further, depending on the MTP, proprietary transport protocols may be possible. For example, there are several transport protocols for replacing TCP in the wireless environment (e.g., [4]). While these protocols typically provide efficient and reliable data transfer, using proprietary protocols may deteriorate the interoperability.

An important function of the MTP is to provide reliable message delivery. Neither IIOP nor HTTP provide sufficient reliability, although at-least-once message delivery semantics can be implemented fairly easily. However, this functionality is not specified by FIPA, thus not making it a generic solution. WAP is the only FIPA-specified MTP that provides the system with sufficient reliability.

The purpose of the envelope layer is to enable transport protocol independent message handling, for example routing of messages without touching the ACL. FIPA has defined three concrete message envelope syntaxes: IIOP-IDL, XML, and bit-efficient [3]. The last one is meant for wireless environments.

We compared the number of bytes needed for different encodings with an envelope we think would be a typical one. In IIOP envelope, the number of bytes is about one kilobyte, whereas with XML envelope it is as large as two kilobytes. This is due to the fact that XML syntax is based on ASCII strings.

Using bit-efficient envelope encoding the output size is reduced to about 700 bytes. However, the dominating factor is the actual content of the message envelope, that is, the strings used as agent-identifiers, addresses, etc. Even the bit-efficient message envelope encoding handles these inefficiently, as it decodes them as strings. Here we do not analyze other aspects of concrete message envelope syntaxes, such as construction or parsing time of a message envelope.

3 ACL Communication

The agent communication language layer defines the outer language used in communication. Here we concentrate on one particular ACL: FIPA-ACL. FIPA-ACL three standard encoding schemes have been specified [2]. Here we compare bit-efficient FIPA-ACL to other standard encoding schemes as well as to some non-standard solutions.

In the bit-efficient FIPA-ACL there are two primary ways to reduce the transfer volume over the wireless link: tokenized syntax and the use of dynamic code table. Although the tokenized syntax gives a significant improvement compared to string-based encodings, the true power of bit-efficient FIPA-ACL lies in the use of dynamic code table. In this encoding scheme similar parts of subsequent messages are not transmitted multiple times over the communication channel, but subsequent occurrences are replaced by short codes.

For the analysis, we selected three test cases in which we analyze the size of the output and both the time it takes to construct a message and the time it takes to parse a message. In the first case, we use fipa-request interaction protocol in which the initiator sends one message (REQUEST) and the participant two (AGREE and INFORM). In the second case, fipa-subscribe interaction protocol is used. The initiator sends a SUBSCRIBE message and the participant replies with 10 INFORM messages. The last case is a combination of the other cases.

Four alternative methods—string-based, XML, serialized Java object, and zipped string—is selected to encode FIPA-ACL messages which are compared against the bit-efficient encoding. The last two schemes are not standard methods to encode messages. However, serialized Java objects are used, for example, when Java RMI is used. Further, as string-based messages are text information, using text compression is perhaps simplest way to encode them efficiently, and therefore we wanted to analyze it as well.

All the experiments are conducted in the Linux environment using FIPA-compliant Jade agent platform [1] (JDK1.3), hardware platform being Intel 366Mhz processor and 128 Mb of memory. Our bit-efficient FIPA-ACL encoding implementation used in the measurements is an implementation of FIPA's standard in Java, and can be used both with Jade and FIPA-OS [7].

Table 1 shows the results of the output size measurement in bytes. In this case, we use the bit-efficient encoding without a dynamic code table. As can be seen, the bit-efficient encoding gives the smallest output in all cases, as was expected. However, the difference between zipped string encoding and bit-efficient encoding is insignificant. The XML encoding output size is about twice as big as

Table 1. The output size in bytes

	String		XML		Bit-efficient		String (cmpr)		ACL object	
	send	recv	send	recv	send	recv	send	recv	send	recv
Case 1	351	720	638	1294	175	371	204	422	1408	2854
Case 2	345	3550	632	6420	167	1800	211	2165	1392	14144
Case 3	696	4270	1270	7714	342	2171	415	2587	2800	16998

the string-based encoding. This also was expected. The serialized `ACLMessage` output size is notably big. This is due to the fact that the Java serialization outputs the class descriptions to each `ObjectOutputStream` to which the serialized objects are written. This could be optimized by using one stream for several objects. However, this is not how for example Java RMI uses streams.

In the second test, we analyze how long does it take to construct the output for different encodings[1]. Table 2 provides the results of these measurements. Each test is repeated 50 times and results (averages) are given in milliseconds. In these tests we first create a Jade `ACLMessage` object of a FIPA-ACL message and then generate the encoded output of the message from this object. The time to create the `ACLMessage` object is not calculated in the final results. The bit-efficient encoding is the fastest in all cases, but the difference to string-based encoding is insignificant. This was expected, since creating string-based message is basically just outputting strings; there is not much to optimize. Surprising in these results is the low performance of the zip algorithm. The algorithm we employ in our test is implemented using native code instead of Java, and therefore it should be faster. Creating serialized objects also is surprisingly slow. The reason is that creating a new `ObjectOutputStream` is a slow operation.

In the third test, we measure the parsing time of an encoded message, that is, how long does it take to create a Jade `ACLMessage` object from encoded stream. In all cases, the data is first read into memory buffer, so the actual reading time is not included in the results. Table 2 gives the results of this test. Again, the bit-efficient encoding is the fastest, and this time it is much faster than any other encoding scheme we measured. The main reasons for this are that (1) very few string comparisons are needed in order to parse the message and that (2) the bit-efficient FIPA-ACL implementation instead of allocating new memory tries to reuse already allocated memory whenever possible.

In all the tests analyzed before we used the bit-efficient encoding without the dynamic code table. Before the tests, we believed that using the dynamic code table should give better compression ratio, but the code table management might slow down both constructing output and parsing input as the code table is implemented in Java. However, as the result will show, the code table management slows down neither message constructing time nor parsing time.

[1] The XML encoding is left out in these tests, as the current version of Jade does not support it.

Table 2. Time to construct and parse the messages (milliseconds)

	String		Bit-efficient		String (cmpr)		ACL object	
	send	recv	send	recv	send	recv	send	recv
	Message construction							
Case 1	4.24	6.24	3.40	4.40	9.74	12.42	105.48	110.48
Case 2	4.24	21.18	3.28	11.78	9.34	37.72	105.44	148.38
Case 3	6.16	25.12	4.16	13.76	12.44	44.32	110.26	157.62
	Message parsing							
Case 1	24.64	30.58	13.78	15.14	27.36	40.48	143.98	152.44
Case 2	24.84	93.20	13.80	37.20	27.34	185.42	144.36	210.50
Case 3	30.56	114.36	15.36	43.64	39.96	231.52	157.66	226.43

Table 3. Number of bytes using different cache sizes

	No cache		2^{8}		2^{9}		2^{10}		2^{15}	
	send	recv	send	recv	send	recv	send	recv	send	recv
Case 1	175	371	175	249	175	257	175	257	175	257
Case 2	167	1800	167	792	167	864	167	864	167	864
Case 3	342	2171	273	972	277	1056	277	1056	277	1056

First, we analyze the size of an encoded message. As can be seen from Table 3, using code table provides a more compact output, but only if there are enough messages to encode. Next we analyze how long does it take to construct the encoded output using different cache sizes. Table 4 shows the results of this test. Coding scheme without a code table is the fastest when there is only one or at most a few messages. This was expected, since when the code table is used, the encoder tries to find every string from the code table, which takes some time. However, when there are several messages and the encoder finds something from the code table, the process of constructing messages becomes faster, as the encoder does not have to copy whole string to the encoded message, but only the corresponding index. Similar results are also achieved when the parsing time is measured (see Table 4). The difference is less significant than in constructing messages. The reason for this is that the code table lookups are much faster when decoding the message.

4 Conclusions and Future Work

We presented the layered model of the FIPA agent communication, and analyzed thoroughly the bit-efficient encoding of FIPA-ACL messages, and showed that this encoding is not only more space-efficient but also more efficient to deal with. Space-efficiency is an important feature especially in wireless environments, but

Table 4. Time to create and parse messages using different cache sizes (milliseconds)

	No cache		2^8		2^9		2^{10}		2^{15}	
	send	recv	send	recv	send	recv	send	recv	send	recv
	Message construction									
Case 1	3.40	4.40	4.52	4.98	4.32	4.98	4.12	5.00	4.22	5.04
Case 2	3.28	11.78	4.20	9.94	4.20	9.96	4.18	9.98	4.20	9.98
Case 3	4.16	13.76	5.16	11.24	5.20	11.42	5.28	11.48	5.16	11.52
	Message parsing									
Case 1	13.78	15.14	15.68	18.04	15.78	18.04	15.56	18.06	15.76	18.10
Case 2	13.80	37.20	15.54	35.58	15.52	35.94	15.68	35.94	15.56	35.92
Case 3	15.36	43.64	18.24	40.20	18.24	40.72	18.14	40.50	18.12	40.66

faster handling of messages becomes important when either processing power is limited or a great deal of messages should be handled. Former is true in today's low-end mobile devices [5] and the latter can be expected in the future when agent technology is employed on a large scale. Additionally, we briefly analyzed message transport issues in wireless environments.

We are currently investigating an efficient encoding for various content languages and models for efficient agent interaction protocols in wireless environments. For the content languages binary-XML might be the best encoding scheme. The interaction protocol on the other hand could be selected based on the current situation. This selection involves careful analysis of the protocol; how many round-trips are needed and how much of data is needed.

References

1. F. Bellifemine, A. Poggi, and G. Rimassa. JADE — A FIPA-compliant agent framework. In *Proceedings of the 4th International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-99)*, pages 97–108, 1999.
2. Foundation for Intelligent Physical Agents. *FIPA-ACL Message Representations*, 2000. Specification numbers 00069, 00070, and 00071.
3. Foundation for Intelligent Physical Agents. *FIPA Agent Message Transport Envelope Representation Specifications*, 2000. Specification numbers 00073, 00085, and 00088.
4. M. Kojo, K. Raatikainen, M. Liljeberg, J. Kiiskinen, and T. Alanko. An efficient transport service for slow wireless telephone links. *IEEE Journal on Selected Areas of Communication*, 15(7):1337–1348, September 1997.
5. M. Laukkanen, S. Tarkoma, and J. Leinonen. FIPA-OS agent platform for small-footprint devices. In *Proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL'01)*, August 2001. To appear.
6. G. Montenegro, S. Dawkins, M. Kojo, V. Magret, and N. Vaidya. RFC 2757: Long Thin Networks, January 2000. (Informational).
7. S. Poslad, P. Buckle, and R. Hadingham. FIPA-OS: The FIPA agent platform available as open source. In *Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000)*, pages 355–368, April 2000.

Information Agents for Mobile and Embedded Devices

Tim Finin, Anupam Joshi, Lalana Kagal, Olga Ratsimore, Vlad Korolev, and Harry Chen

University of Maryland Baltimore County, Baltimore MD 21250 USA

Abstract. The *pervasive computing environments* of the near future will involve the interactions, coordination and cooperation of numerous, casually accessible, and often invisible computing devices. These devices, whether carried on our person or embedded in our homes, businesses and classrooms, will connect via wireless and wired links to one another and to the global networking infrastructure. The result will be a networking milieu with a new level of openness. The localized and dynamic nature of their interactions raises many new issues that draw on and challenge the disciplines of agents, distributed systems, and security. This paper describes recent work by the UMBC Ebiquity research group which addresses some of these issues.

This research was supported in part by the NIST Advanced Technology Program, DARPA contract F30602-97-1-0215, and NSF grants CCR0070802 and IIS9875433.

1 Introduction

In the past few years, the research community has seen plenty of hype associated with wirelesss / pervasive / mobile / ubiquitous computing. Mobile Commerce (M-Commerce) in particular was declared as the “killer app” driving the wireless revolution. In what is undoubtedly testimony to the speed at which internet time moves, the last year has also seen pundits declaring, with equal certainty, that M-Commerce is either a non-starter or that it is dead. In large part, the blame for both the initially hype and the more recent disappointments must rest with the rather narrow vision of m-commerce that some segments of the industry were promoting. In this vision, cell phones (or wirelessly connected PDAs) essentially became mobile storefronts for e-tailers – essentially an incremental change in the present e-tailing idea. We are all familiar with the ads of people buying products and services from their cell phones from the beach. The drawbacks of this idea are not hard to identify, as an increasing number of recent critical commentaries show. This approach essentially treats palmtop devices as consumers (or clients) of goods or information. The information or goods come from servers on the wired side.

This approach (typically based on a client–proxy–server model) has been developed by the academia over the last five or six years in contexts such as web

M. Klusch and F. Zambonelli (Eds): CIA 2001, LNAI 2182, pp. 264–286, 2001.

access from mobile platforms (for instance [23,27,5,30,34,3,22]) or transaction support for database access [11]. Variants of this approach are now emerging from several commercial companies in the form of systems that allow phone based "microbrowsers" or PDAs to access domain specific internet content (headline news, stocks, sports etc.) from designated portals. The WAP consortium (http://www.wap.com/) is leading efforts amongst telecommunication companies and service providers to evolve a standard mechanism and markup language to support this. There have also been efforts, mostly from startups and e-tailers, to allow people to buy goods using the phone microbrowsers or PDAs. In some sense, one can think of this as a *supermarket approach*, where a few identified service providers exist and the traffic in services is one–way.

Yet viewed in a broader perspective, M-Commerce in particular, and M-Services in general, have yet to be fully articulated or explored, especially in the context of emerging mobile ad-hoc networks. Staying with the M-Commerce idea, consider a move away from the prevailing mobile store front vision. In the future, instead of just interacting with the "buy–it–yourself" web storefronts, consumers will be able to interact with service providers in a personalized way via automated service-oriented eMarket models (e.g. [19]). The selling of tangible goods will simply be one such service. Other services would include data, information, software components or even processing time/storage on networked machines. There will be no explicit clients and servers – but peers which can be both consumers and providers of different services. M-Commerce will thus be transformed into what we refer to as *Me-Commerce*, or more generally, into *Me-Services*.

We have been working on a number of project utilizing this alternative *bazaar approach* that involves the cooperation of autonomous ("active"), self-describing ("articulate"), highly interactive ("social"), and adaptive ("intelligent") components which are located in "vicinity" of one another. Such hardware and software components will automatically become aware of each other, establish basic (wireless) communication, exchange information about their basic capabilities (e.g. services they can offer) and requirements (e.g payments they need), discover and exchange APIs, and learn to cooperate effectively to accomplish their individual and collective goals.

This idea of "ad-hoc" teams of entities that are dynamically formed to pursue individual and collective goals can be used to create the software infrastructure needed by the next generation of mobile applications. These will use the emerging third and fourth generation broadband wireless systems, as well as short range narrowband systems such as Bluetooth. Heretofore, the software component of mobile computing has lagged behind its hardware (communication, computing and networking) aspects. Much of the research in the software area is often limited to allowing applications built for the wired world (web, databases etc) to run in the wireless domain using proxy based approaches. Our work has sought to move beyond this and provide a framework which uses the power of mobility and ad-hoc wireless connectivity to enable novel applications. The results will point us toward a physical environment in which devices all around us and even on our body are in constant communication and organize themselves to cooperate to do our bidding.

1.1 A Fanciful Scenario

It is 5:30 in the evening, and Jane's panel meeting at NSF in Arlington is just ending. As she steps out of the building to walk the two blocks to her hotel, she decides that after resting for a bit, she'll get ready and have dinner in one of the nearby restaurants. She tells her palmtop of this decision, and asks it to find out nearby restaurants that serve cuisines that she would like to try but can't find in her home town. The palmtop goes into discovery mode and connects to a nearby broker provided by the Arlington Restauranters Association. It sends it Jane's cuisine preferences and price ranges, and asks for a recommendation for a restaurant where she can eat in about 45 minutes from now. The broker has some static information about its local restaurants such as location and menus. Managers in the restaurants are also feeding it dynamic information such as wait times or any discounts/specials that they may have, based on their current situations. Based on both the static and dynamic information, the broker comes up with a list of possibilities. Meanwhile, as Jane is walking back, her PDA asks the PDAs of others in the area for their opinions of good local restaurants and stores them. When Jane is ready to head out, the Palm PDA pulls the recommendations from the broker and presents them to Jane. Her choices include Vietnamese, Malaysian, Mongolian and Cambodian restaurants, since these cuisines are similar to the Chinese foods she likes. She selects the Vietnamese restaurant, since it is offering a 20% off coupon and had a couple of good mentions in the information her PDA had gathered from other people. Her PDA communicates this choice to the broker, asks that a reservation be made. It then discovers the local map server that the city of Arlington runs, and gets directions from Jane's hotel to her restaurant.

Other scenarios can be drawn in and around meeting rooms, shopping malls, airport terminals, train stations, and highway exits. A commuter train passenger getting off her destination station may need to send an urgent fax but her PDA lacks dial-up capability. Or perhaps, during the trip, she may have received an email attachment that contains a new audio or graphics format that requires a player she does not have on her PDA. As the train starts to slow down entering the station, the passenger starts up an agent that can seek and retrieve the fax or the player software services as she walks out of the station. Another possible scenario that could arise in airport terminals and gateways is one in which arriving and departing passengers may seek each other to exchange leftover currency directly short of bank's buy/sell exchange rates. Similarly, crossing tourists may exchange or sell goodies such as e-coupons, unused museum tickets, and personal tips on their way out of a country they have visited.

1.2 Background

The basic assumption behind our work is that at the level of computing and networking hardware, we will see dramatic changes in the next few years. Computing will become pervasive – a large number of devices (e.g. phones, PDAs, household appliances) will become computationally enabled, and micro/nano sensors (the so called smart dust) will be widely embedded in most engineered artifacts. All of these devices will be (wirelessly) networked. More specifically, we will assume the emergence of (i) palmtop and wearable/embeddable computers, (ii) Bluetooth like systems which will provide short range, moderate bandwidth connections at extremely low costs, and (iii) widely deployed, easily accessible wireless LANs and satellite WANs. We assume that these will be a mixture of traditional low bandwidth systems and the next generation high speed ones. These developments will lead to wireless networks that will scale all the way from ad hoc body area networks to satellite WANs, and link together supercomputers, "palmstations" and embedded sensors & controllers. There is ongoing research in industry and academia in creating the hardware and low level networking protocols that are needed to realize this vision. Some recent efforts have also started in creating smart spaces and integrated sensor networks and addressing the data management challenges that will need to be solved in order to enable new applications.

The scenario we have described earlier serves to illustrate the technical challenges that we and others are trying to address. In particular, as computing becomes pervasive, an increasing number of entities will be both sources and consumers of data and information. This is a significant change from the present, where mobile devices essentially remain consumers of information, and the research challenge has been to get them the right information in a format suited to the bandwidth and resource constraints of the device [23]. In the future we envision, besides the traditional Mobile Support Station based wireless access, many of the devices will communicate using Mobile Ad-hoc Networks (MANET) [18] formed by bluetooth type devices. Further, most devices will have (and use) multiple wireless network interfaces (bluetooth, LAN, cellular etc.) [31]. Besides obtaining information from "canonical and centralized" sources (such as a yellow pages server for restaurants), a device may obtain information (recommendation about restaurants) or service offers (a discount coupon for dinner) from other devices around it – its present *dynamic community.* The scenario demands that the system have some sense of vicinity. The entities in the system must be able to describe themselves as well as discover others and figure how (and whether) to interoperate with them, both at syntactic and semantic levels. Finally, the entities should be able to communicate abstract ideas, (e.g. goals such as what information is it looking for) and be able to negotiate with others for services (I want to know what the traffic conditions are at I 95 Springfield Interchange, in return I can provide traffic conditions in downtown DC.). Of course, the devices participating will be resource constrained and heterogeneous, which poses further problems.

Note that the challenges here go over and beyond those found in heterogeneous and distributed data management, and in ways more subtle than simply handling reduced bandwidth or disconnection, which of course remain important issues. To express this in traditional data management parlance [10], we could say that unlike heterogeneous data access where "schemas" and "catalogs" at least are known in advance, we are talking of a situation where both are highly variable and not known up front. Thus a "query" will return results dependent on where it originates, what location it refers to, and who is around at that time.

1.3 The Remainder of This Paper

In the remainder of this paper we describe on three recent efforts we have made to explore some of the issues mentioned above. We will first presents our work in developing the Centaurus system as a relatively low-level framework on which to build intelligent services in a mobile environment. We then describe research aimed at using a distributed trust model to provide security and access control in such environments. Finally, we will describe an example application, agents2Go, which uses location awareness to provide a simple restaurant recommendation service.

2 Centaurus

The system described in this section provides an infrastructure and communication protocol for providing 'smart' services to these mobile devices. This flexible framework allows any medium to be used for communication between the system and the portable device, including infra-red, and Bluetooth. Using XML for information passing, gives the system a uniform and easily adaptable interface. We explain our trade-offs in implementation and through experiments we show that the design is feasible and that it indeed provides a flexible structure for providing services. Centaurus provides a uniform infrastructure for heterogeneous services, both hardware and software services, to be made available to the users everywhere where they are needed.

Our goal is to provide an infrastructure and communication protocol for wireless services, that minimizes the load on the portable device. While within a confined space, the Client can access the services provided by the nearest Centaurus System (CS) via some short-range communication. The CS is responsible for maintaining a list of services available, and executing them on behalf of any Client that requests them. This minimizes the resource consumption on the Client and also avoids having the services installed on each Client that wishes to use them, which is a blessing for most resource-poor mobile clients.

We also expect all Services to communicate via XML or XML-based languages such as RDF and DAML which are suitable for defining ontologies and describing properties and interfaces of Services. As this is already being widely used, we think that it will help in integrating Centaurus with already existing systems. The information flowing in the system is strictly in the form of CCML (Centaurus Capability Markup Language) which is built on top of XML.

To verify the feasibility of our infrastructure, we have used IR for communication between the Client and the CS in our first stage of the development. One of the main drawbacks is the limitation of the infrared architecture. However, we believe that the simplicity and the affordability of the infrared devices can overcome these limitations. We would like to emphasize that any other medium could be used for communication including Bluetooth; all we provide is the framework.

In the last few of years, a number of technologies have emerged that deal with 'Smart' Homes and Offices. Among them are the Berkeley Ninja Project [8], the Portolano project [13] from the University of Washington, Stanford's Interactive Workspaces Project [16], and Berkeley's Document-based Framework for Internet Application Control [20]. In the remainder of this section we will present the design and modeling issues of our approach and touch on ongoing work. Complete details on the communication protocols, implementation, experimental results and comparisons to other systems are available in [25,26].

2.1 Design

A 'SmartRoom' is equipped with a Centaurus Communication Manager, which continuously broadcasts, through some medium, a client application. A person with a portable device who enters the room for the first time is given the option to install the software. Once the application is installed, it continuously reads the updated list of services. The person is able to choose a service, select a function, fill in the related options and execute the function. These services may be provided by Centaurus systems other than the one the portable device is connected to.

Centaurus Communication Protocol. The Centaurus Communication Protocol (CCOMM) is used to communicate with mobile clients and services and consists of two levels. CCOMM Level1 is used as a glue between some existing communication architecture such as IrDA stack, Bluetooth, or TCP/IP and the generic Level2 protocol. The Centaurus Level1 protocol handles connection and disconnection issues, identification and authentication of the clients and interaction with architecture specific protocols such as IrLAP and IrLMP or Bluetooth. The CCOMM Level2 protocol handles transmission of the XML messages, time synchronization, message fragmentation and re-assembly. The Centaurus Level2 protocol is designed to be insensitive to disconnections, handle multiple clients, provide minimal turnaround times and be easily portable. In fact in the current implementation all communication managers and client communication modules use the exactly the same code base.

Components. There are four main components in a Centaurus System; the Service Managers, the Services, the Communication Managers, and the Clients. The Communication Managers handle all the communication with the Centaurus Client. The Communication Manager could implementing a number of different protocols by having different CComm modules, for example, one that handles IR, another that works with Bluetooth, one that works with HTTP to provide a web interface etc. The Service Managers are the controllers of the system that coordinate the message passing protocol between Clients and Services. The Services are objects that offer certain services to the Centaurus Client. At the present moment the Services contain information to enable them to locate the closet Service Manager and register themselves with it. Once registered, the Services can be requested by any Client talking to any Communication Manager. The Centaurus Client provides a user interface for accessing and executing Services. Figure 1 shows the different components and the relationships between them.

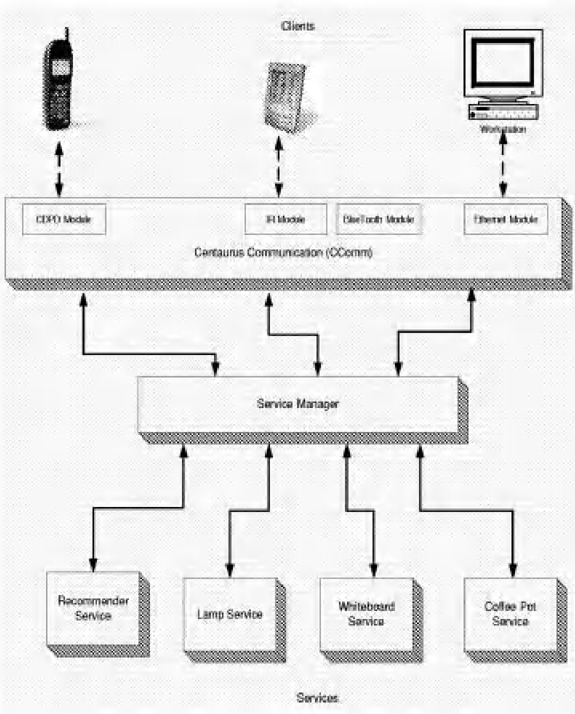


Fig. 1. Centaurus Components

Service Manager. The Service Manager(SM) acts as a mediator between the Services and the Client. This is disregarding the fact that the Client sends the information to its Communication Manager that forwards it to the Service Manager. When a Service starts up, it has to register with the Service Manager, sending its CCML file. This file contains its name, id and the interfaces it implements. When a new Client comes along, the Service Manager sends it a ServiceList Object. This ServiceList object changes dynamically, according to the services registered with the Service Manager. So the Client always has the updated list of services. The Client can select a service, which causes the Service Manager to send the CCML file for the service. The Service Manager then updates its database to reflect that the specific Client is interested in the requested Service. Whenever the Service Manager gets a status update of the Service, it will send it to all interested Clients. The Client will continue to receive status reports from the Service, until it de-registers itself. The Client sends the new CCML file to the Service Manager, after invoking the interfaces of the Service. On receiving this CCML, the Service Manager validates the Client and the CCML. If the Service is still available, the Service Manager sends the CCML to it, otherwise it is queued for a for sometime. Once this timeout expires, an error is returned to the Client. The SM is also responsible for service discovery and leasing. It allows Services to register for a certain amount of time. If it does not receive any status update within that time, the registration is deleted. The SM implements an intelligent lookup for Services, enabling the Clients to search for Services that provide a certain kind of or related function.

Communication Manager. This is responsible for the communication between the Client and the Centaurus system. As mentioned earlier, the system can have a Communication Manager containing different CCOMM modules, one for every type of communication it wishes to implement. The Communication Manager talks via a certain socket to the Service Manager, allowing them to be on different system. When the Communication Manager receives information from a Client, it sends this information directly to the Service Manager through the socket. When it receives data from a Service Manager, it validates the data and looks at the header to decide which Client to send it to.

Services. A Service performs a certain action on behalf of the Client. These Services could range from controlling a light switch or a coffee pot to controlling a printer or even a memo pad service, where Clients can leave messages for each other. Each Service registers with a Service Manager by sending its CCML file, along with its name, id and a brief description of its functionality. Every time its status changes, it informs the Service Manager. It accepts requests only from the Service Manager that it is registered with.

Clients. A Client is a special kind of Service and is treated as a Service. It has to respond to commands and regularly send status updates. A Client talks to the Communication Manager and registers itself with a Service Manager. This registration is similar to the registration of Services. On registration, it receives the ServiceList, which contains the current list of Services. The ServiceList is a Service itself, and causes the Service Manager to send the list of Services, every time a new Service registers, or a Service de-registers.

By choosing a Service, the Client expresses interest in it. The Service Manager sends it the CCML file describing the Service. The Client can invoke the specified functions on the Service, by choosing one of its interfaces. After changing values of certain variables, specified in the CCML for the particular Service, it sends the file to the Service Manager to perform that action. It will receive status updates from all Services that it expresses interest in through the Service Manager, until it specifically informs the Service Manager that it no longer wants to receive these messages. Every time it wants to perform a certain action on a Service, it retrieves the current CCML file from its list, changes the appropriate values and returns the updated CCML to the Service Manager, which forwards it to the selected Service.

The way Centaurus is setup, Clients and the Service Managers only exchange XML messages. By providing XSL transforms, the XML messages can be rendered to fit the Client device. For example, if the Client wants to access a Lamp Control Service, the lamp can be turned on and off. This can be displayed as a true/false button on a PDA but on a cell phone, the the command could be spoken.

Centaurus Capability Markup Language. The CCML is divided into 'system' , 'data', 'addons', 'interfaces', and 'info'. The 'system' portion contains the header information, the id, timestamp, origin, etc. There are two variables, 'update' or 'command'. An 'update' variable is used to inform other Centaurus components about status updates of Services and Clients, whereas the 'command' is only used by Clients to send a command to a certain Service. The system also contains the listening section for a Service or Client. It specifies all the Services that a Service or Client is interested in. Using the 'addons' section, we can add a related Service to another Service, for example, add an Alarm Clock Service to a Lamp-Control Service. We are not currently using this section. All information regarding the variables and their types are contained in the 'data' section. The CCML for a Client always has one or more 'actions' in its data section that a Service Manager can invoke on it. This is used by the SM to change the state of the device.

Two actions can be conveyed in the CCML:

– *AddService*: When this action is set, the Client adds the value of this variable to its InterestList; i.e. the list of services that it is interested in.
– *RemoveService*: This is set by the Service Manager, if the Service that the Client is interested in, is no longer available. It causes the Client to stop listening or using the Service and remove the Service from its InterestList.

The 'interface' section contains information about the interfaces that the object (Service/Client) implements. Other details like the description, and icon for representation are in the 'info' section.

2.2 Conclusion

We have successfully developed the first version of Centaurus. We believe that by providing a uniform infrastructure using XML-encoded data exchange we have shown that it is appropriate and effective for deploying services in an indoor environment. The first stage development, including the Service Manager, IR Communication Manager, MP3 player services, Lamp services etc. has verified that our vision is definitely feasible.

We are also working on a Recommender Service. Instead of returning a list of all possible services that are available to a Client, this service recommends a list of services that might be in the interest of the Client based on the existing environment context. For example, the system returns a coffee-maker control service during the morning to the user, and in the evening it returns a light control service to the user. It may also notice that the user generally wants to listen to to the same list of songs and provide the list as soon as the user steps into the room. We would like to arrange the Service Managers into a hierarchy so that the Services could connect to the closest Service Manager, and the location of a Service Manager need not be coded into the Services. This will also allow the Services to be shared across the Service Managers, so a user could enter one room and use the printer in another room by using the printer Service on the Service Manager in the other room.

Although, our project is far from complete, we believe that now that the framework is in place, adding attractive interfaces for the portable devices, creating new services, and enabling more intelligent brokering of Services will follow easily. We believe that we have crossed all the major hurdles, and completing the remaining portion will be pretty straightforward.

3 Distributed Trust

Traditionally, security for stand alone computers and small networks was handled by physical security and logging into computers and domains. With open networks like the *Internet* and pervasive environments, issues concerning security and trust become crucial. There is no longer the physical aspect of security due to the distributed nature of the networks and the concept of user authentication to a domain is not possible. Imagine a scenario where a user, with a portable device, walking through a building, switches on the lights in the corridor and lowers the temperature of the room that he/she enters. This is an example of pervasive/ubiquitous environments that will soon be a reality. In these *ubiquitous computing* environments users expect to access resources and services anytime and anywhere, leading to serious security risks and problems with access control as these resources can now be accessed by almost anyone with a mobile device. Adding security to such open models is extremely difficult with problems at many levels. We can not assume an architecture with a central authority and access control is required for foreign users. The portable hand-held and embedded devices involved have severe limitations in their processing capabilities, memory capacities, software support and bandwidth characteristics. Moreover, there is currently a great deal of heterogeneity in the hardware and software environments and this is likely to continue for the foreseeable future. Finally, in such an open, heterogeneous, distributed environment there is a great likelihood that inconsistent interpretations will be made of the security information in different domains.

We encountered several problems with security for Centaurus. Firstly, it is not possible to have a central authority for a single building, or even a group of rooms. So we have to use a distributed model, with the *service managers* arranged in a hierarchy. It is also not sufficient to authenticate users because most users are foreign to the system, i.e. they are not known. So there is no means of providing access control. Consider a *Centaurus Smartroom* in an office, equipped with an MP3 player, fax machine, several lights, a coffee maker and a printer. If a user, John, walks, how does the room decide which services John has the right to access. Just authenticating John's certificate gives no information on access control because John is an unknown user. Unless it is known in advance which users are going to access the room and their access rights are also known, simple authentication and access control is not going work. Assume John does not work in the office, but in one of its partner firms. How will the system decide whether to allow him to use certain services?

We suggest enhancing security by the addition of trust, which is similar to the way security is handled in human societies. Some authorized person in the office can *delegate* the use of the services in the room to John, for the period during which he is in the office. Trust management can be viewed as developing of security policies, the assignment of credentials to entities, verifying if the credentials fulfill the policy and the delegation of trust to third parties [32,4]. We propose a lightweight solution for trust management that is applicable for the *Internet*, which we are tailoring for pervasive computing environments.

3.1 Distributed Trust

The distributed trust approach involves articulating policies for authentication, access control and delegation, assigning credentials to individuals, allowing entities to delegate or defer their rights to third parties and providing access control by checking if the initiators credentials fulfill the policies. If an individual has credentials allowing it to access a certain service, the individual is said to have the *right* to access the service. If an individual defers a right he/she has to another individual, it is called a *delegation*, the former is called delegator and the latter delegatee.

Blaze, who coined the term *distributed trust management*, tries to solve the trust problem by binding public keys to access control without authentication [32,4]. His PolicyMaker, given a *policy*, answers queries about trust. Though powerful, the policy definition is complicated and not easy to understand for non-programmers who are probably going to develop the policy. The Simple Public Key Infrastructure (SPKI) was the first proposed standard for distributed trust management [12]. This solution, though simple and elegant, only included a rudimentary notion of delegation. Pretty Good Privacy or PGP [40] was developed to enable the sending of secure email without a secure key exchange or a central authority. In PGP, a keyholder (an individual associated with a public/private key pair) learns about the public keys of others through introductions from trusted friends. Whether or not a user accepts the information about new keys depends on the number of introduces and the degree to which they are trusted (quantized in PGP to three levels: fully trusted, partly trusted and untrusted). Some of the main problems with PGP are with key distribution and management. The Use-Condition Centered Approach [21] uses certificates for use-conditions that are created by those responsible for the resources. This can only be used when the resource is simple enough to be described by use-conditions, but in large systems there could be many types of access like read, write, execute etc. Delegation logics [29,17] is similar to our approach, however it is not able to capture adequately the constraints associated with rights and delegations.

3.2 Trust Architecture

A *security policy* is a set of rules for authorization, access control and trust in a certain domain. All devices/users of the domain must enforce its policy and can impose a *local policy* as well. A service being accessed by a foreign user should verify that the user conforms to both its policies. Unification of the information provided by the user and the policy of the resource is difficult because of the domains could be using different ontologies.

In our system, we model permissions as the rights of a user or agent. We associate rights with actions, so possession of a right permits the corresponding agent to perform or request a certain action. We are currently exploring the extension of our model to also include obligations and the repercussions of failing to fulfill them, as well as other normative or deontic concepts such as entitlements, prohibitions, duty, responsibility, etc.

Rights or privileges can be given to trusted agents who are responsible for the actions of the agents to whom they subsequently delegate the privileges. So the agents will only delegate to agents that they trust. This forms a delegation chain. If any agent along this chain fails to meet the requirements associated with a delegated right, the chain is broken and all agents following the failure are not permitted to perform the action associated with the right [24].

We have implemented a Distributed Trust Mechanism for a Supply Chain Management (SCM) system for the CIIMPLEX EECOMS project [38,7]. An SCM system consists of groups of agents that are either vendors or clients. These agents need to access resources in each others domains. For example, a software consultant may need to access the database of its client. Each group of agents that are part of the same company form a policy domain, and follow the same security policy. The policy in each domain is enforced by special agents called *security agents*. Agents are identified by X.509 [15] authentication certificates and all communication is via signed messages. Security agents are able to reason about these signed messages and policies to provide authorization.

Agents can make requests, either for certain actions or to ask for permission to perform an action, and while doing so they attach all their credentials, i.e. ID certificate, authorization certificates etc., to the request. The *security agents* generate authorization certificates, that can be used as 'tickets' to access a certain resource. The authorization certificates are generated as the result of a request for permission, if the request is valid. Policies consist of rules regarding authorization, delegation and some basic knowledge about the agents. This knowledge could be the role that the agent fills in the organization (e.g. system administrator), and permissions associated with the agents or the roles. An agent is allowed to execute any action that it has the permission to execute, or if the ability has been delegated to it by an agent with the right to delegate. In our system we view 'delegation' as a permission itself. Only an agent with the right to delegate a certain action can actually delegate that action, and the ability to delegate, itself can be delegated.

We have developed a representation of trust information in Prolog, that allows flexibility in describing requests and delegations. Delegations can be constrained by specifying whether the delegatee has the permission to delegate and to whom it can re-delegate.

Consider the previous example of John entering a *SmartRoom*. John is an employee of one of the office's partners. He approaches one of the managers, Susan, and asks for permission to use the services in the *SmartRoom*. According to the policy, Susan has the right to delegate those rights. Susan delegates to John the right to use the lights, the coffee maker and the printer but not the fax machine. Susan sends a message to the *Centaurus service manager* of the office, associating John's identity certificate with the rights. When John enters the room, the service manager of the room reads his identity certificate but can't locate any access rights. So it asks the service manager above it in the hierarchy, this continues till the request reaches the root. This service manager has the delegation made by Susan. The service manager sends a list of rights back down the chain. On receiving the list of rights, the SmartRoom's service manager creates a *delegation certificate* and returns it to John. Now, while the delegation certificate is valid, John can access the services. Once the certificate expires, the service manager checks with the root if the delegation is still valid and creates another certificate. By having very short periods for the certificate, the system handles *revocation* of rights easily. While requesting for a service, the client on John's hand-held device will send his ID certificate and the delegation certificate to the service manager. If the delegation certificate is still valid, the service manager will allow John to access the printer, coffee maker and lights. This scenario demonstrates the importance of trust over security.

3.3 Ongoing Work

We are working on integrating trust into the security infrastructure for *Centaurus*. We believe that trust will add a new dimension to pervasive computing, allowing more flexibility and control over access control.

At the same time, we are improving on our trust architecture. The system is being extended to include entitlements, prohibitions and obligations and the ability to delegate them. We are not sure if these delegations can be interpreted correctly. In many contexts, an agent may want to delegate some obligation to another agent who is willing and able to assume it. If the obligation is not fulfilled, then hopefully the former agents reputation will not suffer as much as the agent which took on the obligation. Entitlements are stronger than permissions and prohibitions are negative permissions, both of which can be delegated in the same way as permissions.

Our approach is to treat delegation as a "speech act" and to associate with it appropriate conversational protocols along the lines of those used in agent communication languages [28]. In some cases it may be necessary to allow or even require the delagatee to accept or reject the delegated object. Although we typically view privileges as extending our capabilities and thus being purely beneficial it is not always the case. If we allow privileges to entail corresponding obligations when the privilege is exercised, an agent may not want to accept the delegation of the privilege. Similarly, an agent may want to reject the delegation of an obligation or a prohibition, if it is allowed to.

Another important issue with distributed networks is that of privacy. Users do not want their names and actions to be logged, so we are trying to do away with with X.509 certificates and replace them with XML signatures [39] from a *trusted authority* and does not include the identity of the bearer, but only a role or designation.

Our past work on distributed trust represented actions, privileges, delegations and security policy as horn clauses encoded in Prolog. In order to develop a approach that is better suited to sharing information in an open environment, we are recasting this work in DAML [9], the DARPA Agents Markup Language. DAML is built on XML and RDF and provides a description logic language for defining and using ontologies on the web. Using DAML, we are defining the basic ontologies for actions, agents, roles, privileges, obligations, security policies and other key classes and properties. In applying our framework, one must extend the initial ontology (http://daml.umbc.edu/dt.daml) by defining domain specific class of actions, roles, privileges, etc. and creating appropriate individuals.

3.4 Conclusion

According to the current paradigm of pervasive computing, soon we will be able to access information and services virtually anywhere and at any time via any device, whether it is our phones, PDAs, laptops or even watches. We are at the threshold of 'SmartSpaces' where the environment is intelligent enough to pick up subtle user inputs like user movement, proximity or temperature and supply any required service and/or information to the user through a mobile device or by using voice/sound/lights or other media for communication. In environments like this, security is very important. But just security itself is insufficient, and *trust management* is required. Trust management is the development of security policies and credentials, the checking of presented credentials against the policy and the delegation of permissions. To help make the vision of ubiquitous computing a reality, *trust* should be included in the security infrastructure.

4 Agents2Go

In this section we describe the Agents2Go system that addresses some problems related to location dependence that arise in an M-commerce environment. Agents2Go is a distributed system that provides mobile users with the ability to obtain location dependent services and information. Our system also automatically obtains a user's current geographical location in CDPD (Cellular Digital Packet Data) based systems without relying on external aids such as GPS (Global Positioning System).

One of the most critical requirements for M-Commerce is the ability to discover services in a given context. Context aware computing [35,2] is a challenging goal and involves understanding a persons location, focus of attention, mood, immediate objectives and even ultimate underlying goals. Our objective in Agents2Go is relatively modest – learning the user's location using a simple CDPD hardware environment so that, for example, a user arriving at a location that he/she has never visited before should be able to find a local cab service. Current mobile devices have well known inherent limitations [22] like limited power supply, smaller user interface, limited computing power, limited bandwidth and storage space. These limitations necessitate the development of systems that provide mobile users with high quality, precise and context relevant information. It is important that these systems be highly scalable since the demand for service searches will increase in the future.

A location dependent search utilizes a user's current geographical location to refine their search and provide access to locally available services. One of the challenges of location-based searches is determining the user's current location. Users are often uncertain, or even completely unaware, of their current geographical location making location based searching more difficult. An automated detection of the user's current location would be very helpful in eliminating this problem.

Location dependent systems are naturally described and implemented as distributed systems. This also improves their fault tolerance and scalability. For instance service information can be grouped by location and managed by a server responsible for the specified geographical region. In such a decentralized scheme user requests are processed at the local server and do not burden the rest of the system. This makes the system more efficient, responsive and scalable.

4.1 Related Work

There are a number of platforms that provide multi-agent infrastructures to allow inter-agent communication and collaboration. These platforms can be used to create collaborative intelligent agent environments that could provide location based information services. One such infrastructure is the Lightweight Extensible Agent Platform (LEAP) [1]. LEAP provides a lightweight platform that is executable on small devices such as PDAs and phones. It is FIPA [14], compliant and also supports WAP and TCP/IP. The YellowStone Project [36] from Reticular Systems, Inc. also deals with location dependent services. Essentially, there are communities of software agents called agencies, which provide information services and e-commerce support for a particular geographical area. However, a participating user has to specify his current geographical location. Very recently researchers at AT&T Research Labs have described a project with similar goals [33], which also locates the user in a CDPD [37] environment using cell tower IDs.

Our Agents2Go system has several distinguishing features that provide advantages over the existing location dependent service search systems. First, it is a platform to deploy any location dependent service, not just to provide location

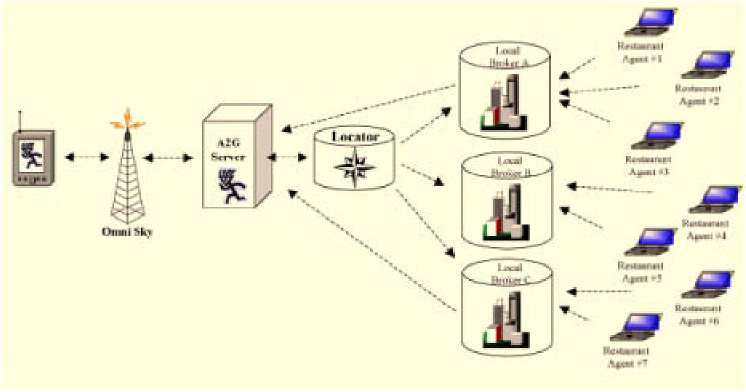


Fig. 2. Agents2Go is a simple prototype application exploring which makes use of location aware services.

dependent information. Second, our system automatically discovers a user's location and uses this information to refine its searches to provide the most relevant information to the user. Also the Agents2Go system allows service providers to actively participate in the system through dynamic information updates. This improves the quality of information or services presented to the user and ensures that the service providers are able to send their latest promotions/updates to their users.

4.2 Components of the Agents2Go System

The Agents2Go System, illustrated in Figure 2, is composed of several components: the PalmApp, the Agents2Go Server, the Locator, the Agents2Go Information Repository, the restaurant Brokers and participating Restaurant Agents.

PalmApp. The PalmApp is the end user interface to the Agents2Go System. This component runs on the user's PDA equipped with a CDPD modem. Essentially, it is a generic "form visualizer" that is independent of the system functionality. To reduce complexity, in-house markup tags are used to specify the layout and components of the form. In our system, the PalmApp captures a user request, converts it to an appropriate format, and then forwards that request to the Agents2Go Server. The PalmApp also handles the responses from The Agents2Go Server and presents them to the user.

Communication and Location Detection. All the messages that are exchanged between the Agents2Go Server and the PalmApp are sent using the CCOMM described in section two. Our current Agents2Go System uses CDPD Level1 Module, which is an extension of the UDP Level1 module. CDPD provides an infrastructure that allows the transmission of data over idle capacity of already existing cellular voice networks. Cellular networks consist of cell towers with a unique ID and a specific geographical cell over which it provides services. Our system employs these tower IDs to identify a user location. We have developed a library that allows us to control and interact with Novatel wireless modems through MSCP (Minstrel Status and Configuration Protocol). The PDA's CDPD module employs this library to obtain periodic status reports which contain information like cell tower signal strength (which can be used to minimize packet loss) and the tower ID.

Messages. All the messages that are exchanged between the PalmApp and the Agents2Go Server are encapsulated in a generic message format which specifies a sender ID, a message type and message content. Messages sent from a PDA to a Server use that PDA's ID as the Sender ID and messages sent from a Server to a PDA use the Server's ID as the Sender ID. The message type field can contain one of three message types: "response form message", "form request message", or "form data message". The Message Content field contains the message itself. The PalmApp uses a generic "form request message" to request forms from the Agents2Go Server that are displayed to the user. This message specifies the form name that the PalmApp is requesting and the cell tower, with which it is currently communicating.

When it is started, PalmApp requests an "initial query form" from the Agents2Go Server through which the user can issue requests. Requests are converted into a "form data message" and sent to the Agents2Go Server. The response is a form that the PalmApp displays to the user that may contain a "home" button that causes the PalmApp to generate the "initial query form" message to allow the user to enter a new query. Unlike a "form request message" or a "form data message", the "response form message" is initiated at the Agents2Go Server and destined for the PalmApp. This message contains a form that the PalmApp is required to display to the user. This message may contain a response to the user's query, an error message, etc.

Agents2Go Server. The Agents2Go Server is the component that handles messages to and from a PalmApp. User queries are forwarded to the Locator, and the corresponding responses are forwarded back to the PalmApp. Upon receiving a "form request message", the Agents2Go Server reads the requested form from a file. If the desired form cannot be located, a suitable error form is sent back to the PalmApp. Once the desired form is located, the Agents2Go Server uses a lookup table to map the specified cell tower ID (obtained from the request message) to its neighborhood name. This neighborhood name is inserted into the location field of the form that needs to be displayed to the user. This neighborhood name, which can be changed by the user, is inserted into location field of the form displayed to the user. Thus a user, regardless of his current location, can find information about any participating region. This is encapsulated in a "response form message" and sent back to the PalmApp. When the Server receives a "form data message" from a PalmApp, it forwards it to the Locator. Other alternative designs could be used, a "form data message" could be forwarded to the corresponding Broker. Once the Agents2Go Server receives a response from either the Locator or a Broker, it generates the corresponding "response form message" and sends it to back the PalmApp.

Locator. The Locator is the component that receives requests from the Agents2Go Server, determines which Broker is responsible for the area from which the request originated and then forwards the request to that Broker. The Locator maintains a dynamic table that maps geographical areas to Brokers. The Locator listens on a well defined port for registration messages from Brokers that specify a port on which the Broker will accept requests forwarded by the Locator, and the geographical area for which it is responsible. Upon receiving a request form the Agents2Go Server, the Locator looks inside the request string and extracts the point of origin information. This information is used to determine the designated Broker. The Locator then forwards the request to that Broker. If the Locator is unable to locate a suitable Broker for the given request, the Locator sends a "broker not found" message back to the Agents2Go Server. A reliable communication channel is maintained between the Agents2Go Server and the Locator for all message transfers.

Broker. The Broker maintains information about restaurants in its designated geographical region, processes requests from users and generates suitable responses. These requests are forwarded to the Broker from the Locator and the generated responses are sent to the Server for forwarding to the requesting PalmApp. Agents2Go partitions participating restaurants into sets based on the geographical region in which these restaurants are located. Each coverage region is assigned a unique name and is serviced by a designated Broker which is responsible for generation of replies for requests pertaining to its coverage region. Our current implementation allows grouping of several geographical regions or partitioning a single geographical region to construct a coverage region.

Every Broker in the Agents2Go System is also associated with a specific Agents2Go Information Repository – a set of databases that contain information about participating restaurants in a Broker's coverage region. Restaurant information like name, address, cuisine etc. of all participating restaurants in that coverage region is distributed among these databases. This information can be classified as static, since it rarely changes. The Broker is also responsible for frequently changing restaurant information like waiting times and promotions. This kind of information can be classified as dynamic. This dynamic information is maintained within the Broker itself. This separation of dynamic and static information reduces the number of messages that is exchanged between the Agents2Go System components.

It is common for wireless cells to overlap. If a user is in a cell overlap region, then that user's PDA connection can hop from one overlapping cell to another. So, the cell tower ID that the user's PDA picks up can change quite frequently. If the cell overlap is contained within a single coverage region, then cell hopping is not an issue because any cell ID that is picked up in that cell overlap will map to the same coverage region name and is managed by the same Broker. However, if the cell overlap is on the border of two or more coverage regions, the user's PDA may pick up IDs that belong to cell towers that service neighboring coverage regions. This could create a scenario where a user's request query could be routed to a neighboring Broker that has absolutely no information about the current location of that user.

The Agents2Go System solves this issue by imposing a policy that prohibits coverage regions from overlapping unless there are some cell overlaps falling on their borders. We also require a special configuration for the Information Repositories that are associated with these overlapping coverage regions. Restaurants in overlapping regions are partitioned into two disjoint sets: shared and native type. A shared set contains restaurants that are located in the areas of cell overlap (that fall on the borders of coverage regions) and a native set contains the remaining restaurants that are in the Broker's coverage region but not in the cell overlap. During his interaction with Agents2Go a user might move to a different coverage region, while the request is being processed. In this scenario, the reply to the request contains the information relevant to the region from where the request originated. This information could still be of relevance to the user since he/she is not far form the initial coverage region.

On initialization, a Broker establishes connection with its Agents2Go Information Repository. The Broker queries its repository to obtain IDs of restaurants for which it will broker information. If the Broker is unable to establish required connection(s) with its repository, the initialization fails and the Broker exits gracefully. Upon successful connection establishment, the Broker builds a "waiting time" table. The "waiting time" table is a data structure that the Broker uses to maintain the dynamically changing restaurant information. The restaurant IDs and the location identifiers for the restaurants are used as keys of the table. The values of the table are the waiting time information, promotion information and time stamps of updates. Once the table is built the Broker registers itself with the Locator component. The registration contains geographical regions that this Broker administers and the port on which the Broker will listen for forwarded requests from the Locator. If the registration with the Locator fails, the Broker exits gracefully.

Once initialized, the Broker starts to listen for updates sent by the local Restaurant Agents and requests forwarded by the Locator. When a Broker receives a wait time update and some promotion information from a Restaurant Agent, it timestamps it and then caches this information into the "waiting time" table. When the Broker receives a request from the Locator, it first checks for the validity of the request. If the request string does not match the expected format, further processing of that request is terminated and an error message is sent back to the user. For valid requests, corresponding database queries are dynamically generated. Request parameters are dynamically incorporated into a database query.

If the request contains a waiting time limit, then the Broker searches its "waiting time" table for the restaurants that have their waiting time below the requested time limit. This search returns IDs of the restaurants that have suitable waiting time. Returned IDs are also incorporated into the database search query. Once the query is constructed, it is executed on the Broker's Agents2Go Information Repository. If no records are found, then a "No record found" message is returned to the user. If matching records are found, a timestamp for each result record is evaluated. This timestamp can belong to one of three age groups: “fresh” age group, “aged” age group, or “trashed” age group. A record will be treated differently depending on its age group, and on whether the user is interested in dynamic information.

To identify the age group of a timestamp, the difference between the value of that timestamp and current system time is calculated. This difference is the age of the timestamp. The timestamp age is compared against two threshold values. The first, lower, threshold denotes the limit between the “fresh “age group and the “aged” age group. The second, higher, threshold denotes the limit between the “aged” age group and the “trashed” age group. Hence, if a timestamp is “fresh”, the record that has been selected is sent to the user, and the dynamic information is displayed in its regular format. Else, if a timestamp is “aged”, the record is sent to the user along with a warning that the record is not up to date. And finally, if a timestamp is “trashed”, the user request is analyzed to determine if the user is interested in dynamic information. If the user’s request specifies a waiting time limit, the record is dropped from the result set. On the other hand, if there is no time limit specified, the record is sent to the user, but the dynamic, outdated portion of that record is replaced with an “Information is unavailable” message. This classification of the record’s timestamps gives users some flexibility. Users themselves can determine if the dynamic information is useful.

Once the response to the query is formed, it is converted into an appropriate format and sent to the Agents2Go server. So there are two types of Restaurant Information messages that could be sent to the Agents2Go Server.

The Restaurant Agent. The Restaurant Agent resides and runs at the location of the participating restaurant and allows the restaurant manager to update dynamic information such as waiting times, promotion information, etc. Updates from the restaurant are sent to the Broker that is responsible for the geographical area in which the restaurant resides. If the restaurant is located in a cell overlap region, which is managed by several Brokers, then the update message is sent to every Broker that manages the overlap region. The update message contains the restaurant id, and other relevant information like the value of the wait time for table for two, the wait time for table for four, the wait time for table for six, etc. Each update message also contains a timestamp specifying the creation time. The Broker, upon receiving an update message, extracts the relevant values from the message and inserts these values into the appropriate row of its “waiting time” table.

4.3 Conclusion

We have implemented a working prototype of the Agents2Go System as a location aware, distributed system that allows mobile users to request and receive various services information that is of most relevance to their current geographical location. Thus, the mobile users will not be burdened with extraneous information for services in remote locations. Also, the Agents2Go System allows service providers to supply dynamic service updates. This dramatically improves the value of the service for the providers and gives users more refined service information. Our implementation currently deals with restaurants, but it could be easily updated to work with other location specific services. All of the above mentioned features make our system well suited for various M-Commerce experiments using the existing CDPD infrastructure in use in the United States.

5 Conclusions

The *pervasive computing environments* of the near future will involve the interactions, coordination and cooperation of numerous, casually accessible, and often invisible computing devices. These devices, whether carried on our person or embedded in our homes, businesses and classrooms, will connect via wireless and wired links to one another and to the global networking infrastructure. The result will be a networking milieu with a new level of dynamism and openness. The localized and dynamic nature of their interactions raises many new issues that draw on and challenge the disciplines of agents, distributed systems, and security. This paper describes recent work by the UMBC Ebiquity research group which addresses some of these issues ranging from the need to develop a common communication infrastructure, to requirements for security and access control to techniques for location recognition.

References

1. A FIPA platform for handheld and mobile devices. In *Proceedings of the 2001 Workshop on Agent Theories, Architectures, and Languages*, 2001.
2. G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *ACM Wireless Networks*, 3:421–433.
3. Harini Bharadvaj, A. Joshi, and Sansanee Auephanwiriyakyl. An active transcoding proxy to support mobile web access. In *Proc. IEEE Sumposium on Reliable Distributed Systems*, October 1998.
4. Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, pages 185–210, 1999.
5. E.A. Brewer, R.H. Katz, Y. Chawathe, A. Fox, S.D. Gribble, T. Hodes, G. Nguyen, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V. Padmanabhan, and S. Seshan. A network architecture for heterogeneous mobile computing. *IEEE Personal Communications Magazine*, 5(5):8–24, 1998.
6. Harry Chen, Anupam Joshi, Tim Finin, and Dipanjan Chakraborty. Dynamic service discovery for mobile computing: Intelligent agents meet jini in the aether. *Baltzer Science Journal on Cluster Computing, Special Issue on Advances in Distributed and Mobile Systems and Communications*, 2001.
7. R. Scott Cost, Tim Finin, Yannis Labrou, Xiaocheng Luan, Ian Soboroff Yun Peng, James Mayfield, and Akram Boughannam. An agent-based infrastructure for enterprise integration. In *First International Symposium on Agent Systems and Applications*, October 1999.

8. Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, and Randy H. Katz. An architecture for a secure service discovery service. In *Fifth Annual International Conference on Mobile Computing and Networks (MobiCom '99)*, pages 24–35, Seattle, 1999.
9. DAML. Darpa agent markup language specification, http://www.daml.org/.
10. M. Dunham, P. Chrysanthis, A. Joshi, V. Kumar, K. Ramamritham, O. Wolfson, and S. Zdonik. Issues in wireless data management. Discussion of the wireless data management panel at NSF IDM PIs Meeting, March 2000.
11. M. Dunham, A. Helal, and S. Balakrishnan. A mobile transaction model that captures both the data and movement behavior. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2(2):149–162, 1997.
12. Carl M. Ellison, Bill Frantz, and Brian M. Thomas. Simple public key certificate. Internet document, 1996.
13. Mike Esler, Jeffrey Hightower, Tom Anderson, and Gaetano Borriello. Next century challenges: Data-centric networking for invisible computing. In *Mobile Computing and Networking*, pages 256–262, 1999.
14. FIPA. FIPA 97 specification part 2: Agent communication language. Technical report, FIPA - Foundation for Intelligent Physical Agents, october 1997.
15. Internet Engineering Task Force. Public-key infrastructure (x.509), http://www.ietf.org/html.charters/pkix-charter.html.
16. Armando Fox, Brad Johanson, Pat Hanrahan, and Terry Winograd. Integrating information appliances into an interactive workspace. *IEEE Computer Graphics and Applications*, 20(3), 2000.
17. Benjamin Grosof and Yannis Labrou. An approach to using xml and a rule-based content language with an agent communication language, 1999.
18. Z. J. Has. Panel report on ad hoc networks - MILCOM'97. *Mobile Computing and Communications Review*, 2(1), January 1998.
19. A. Helal, M. Wang, A. Jagatheesan, and R. Krithivasan. Brokering based selforganizing e-service communities. In *Fifth International Symposium on Autonomous Decentralized Systems (ISADS)*, Dallas, Texas, March 2001.
20. T. Hodes and R. H. Katz. A document-based framework for internet application control. In *Proceedings of the Second USENIX Symposium on Internet Technologies and Systems*, October 1999.
21. W. Johnston and C. Larsen. A use-condition centered approach to authenticated global capabilities: Security architectures for large-scale distributed collaboratory environments. Technical Report Technical Report 3885, Lawrence Berkeley National Laboratory, 1996.
22. A. Joshi, S. Weerawarana, and E. N. Houstis. Disconnected Browsing of Distributed Information. In *Proc. Seventh IEEE Intl. Workshop on Research Issues in Data Engineering*, pages 101–108. IEEE, April 1997.
23. Anupam Joshi. On proxy agents, mobility and web access. *ACM/Baltzer Journal of Mobile Networks and Applications*, 2000. (accepted for publication, also availbe as UMBC CS TR 99-02).
24. Lalana Kagal, Tim Finin, and Yun Peng. A framework for distributed trust management. In *To appear in proceedings of IJCAI-01 Workshop on Autonomy, Delegation and Control*, 2001.
25. Lalana Kagal, Vlad Korolev, Harry Chen, Anupam Joshi, and Tim Finin. Centaurus: A framework for intelligent services in a mobile environment. In *Proceedings of International Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, The 21st International Conference on Distributed Computing Systems (ICDCS-21) April 16-19, 2001.

26. Lalana Kagal, Vladimir Korolev, Sasikanth Avancha, Anupam Joshi, Timothy Finin, and Yelena Yesha. A highly adaptable infrastructure for service discovery and management in ubiquitous computing. Technical Report TR CS-01-06, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 2001.
27. R. H. Katz, E. A. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. Hodes, D. Jiang, G. T. Nguyen, V. Padmanabhan, and M. Stemm. The bay area research wireless access network (barwan). In *Proceedings Spring COMPCON Conference*, 1996.
28. Yannis Labrou, Tim Finin, and Yun Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems*, 14(2):45–52, / 1999.
29. Li, Feigenbaum, and Grosof. A logic-based knowledge representation for authorization with delegation. In *PCSFW: Proceedings of The 12th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1999.
30. M. Liljeberg, M. Kojo, and K. Raatikainen. Enhanced services for world-wide web in mobile wan environment. http://www.cs.Helsinki.FI/research/mowgli/mowgli-papers.html, 1996.
31. D. Maltz and P. Bhagwat. Msocks: An architecture for transport layer mobility. In *Proc. IEEE Infocom 98, San Francisco*, pages 1037–1045, April 1998.
32. M.Blaze, J.Feigenbaum, and J.Lacy. Decentralized trust management. *IEEE Proceedings of the 17th Symposium*, 1996.
33. S. Muthukrishnan, Rittwik Jana, Theodore Johnson, and Andrea Vitaletti. Location based services in a wireless wan using cellular digital packet data (cdpd). In *Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE01)*, May 2001.
34. B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R.Walker. Agile application-aware adaptation for mobility. In *Proceedings of the 16th ACM Symposium on Operating System Principles.*
35. Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.
36. Reticular Systems. The yellowstone project.
37. Mark Taylor, William Waung, and Mohsen Banan. *Internetwork Mobility: The CDPD Approach.* Professional Technical Reference. Prentice Hall, 1996.
38. W. J. Tolone, B. Chu, J. Long, T. Finin, and Y. Peng. Supporting human interactions within integrated manufacturing systems. *International Journal of Agile Manufacturing*, 1998. To appear.
39. W3C. Xml signature http://www.w3.org/signature/.
40. Philip R. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, MA, USA, 1995.

Author Index