Lecture Notes in Artificial Intelligence 2112

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

Ryszard Kowalczyk Seng Wai Loke
Nancy E. Reed Graham Williams (Eds.)

Advances in Artificial Intelligence

PRICAI 2000 Workshop Reader

Four Workshops held at PRICAI 2000
Melbourne, Australia, August 28 - September 1, 2000
Revised Papers

Springer

Volume Editors

Ryszard Kowalczyk
CSIRO Mathematical and Information Sciences
723 Swanston Street, Carlton, VIC 3053, Australia
E-mail: Ryszard.Kowalczyk@cmis.csiro.au

Seng Wai Loke
RMIT University, School of Computer Science and Information Technology
GPO Box 2476V, Melbourne, VIC 3001, Australia
E-mail: swloke@cs.rmit.edu.au

Nancy E. Reed
Linköping University, Department of Computer and Information Science
581 83 Linköping, Sweden
E-mail: nanre@ida.liu.se

Graham J. Williams
CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra, ACT 2601, Australia
E-mail: Graham.Williams@cmis.csiro.au

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Advances in artificial intelligence : PRICAI 2000 workshop reader ; four workshops held at PRICAI 2000, Melbourne, Australia, August 28 - September 1, 2000 ; revised papers / Ryszard Kowalszyk ... ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2112 : Lecture notes in artificial intelligence)
ISBN 3-540-42597-7

CR Subject Classification (1998): I.2

ISBN 3-540-42597-7 Springer-Verlag Berlin Heidelberg New York

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

http://www.springer.de

Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna
Printed on acid-free paper SPIN: 10839817 06/3142 5 4 3 2 1 0

Preface

The Pacific Rim International Conference on Artificial Intelligence (PRICAI) is the leading conference in the Pacific Rim region for the presentation of research and the latest developments in Artificial Intelligence, including the application of AI to problems of social and economic importance. PRICAI 2000 was the sixth in the biennial series of conferences and was held in Melbourne, Australia, August 28–September 1, 2000.

The PRICAI 2000 workshops were designed to provide a forum for researchers and practitioners to present and exchange the latest developments at the AI frontier. Workshops are more specialized and on a smaller scale than conferences to facilitate active interaction among the attendees. They encourage AI theories to meet reality, AI researchers to work with practitioners, and vice versa. Through workshops, both sides get exposure to evolving research and tools, and to the practical problems to which AI can be applied. As an excellent indicator of the current level of active research and development activities, PRICAI 2000 included a total of seven workshops: *AI and the Internet*, *Intelligent Information Agents*, *Applications of AI in Industry*, *Multi-Agent Systems*, *Text and Web Mining*, *AI in E-Commerce*, and *Teams with Adjustable Autonomy*.

The work from four of the seven workshops has formed this unique collection of four parts. *Part I* reports on Applications of AI in Industry; *Part II* covers AI in E-Commerce; *Part III* details Intelligent Information Agents, and *Part IV* discusses the issues of Teamwork and Adjustable Autonomy in Agents. Each workshop paper was accepted after review by at least two experts. Further improvements were included in many papers in preparation for this collection. Readers can find diverse topics and careful discussions centered around the four important themes in our ever changing world. This collection plays an important role in bridging the gap between AI theory and practice, to emphasize the importance of AI in the research and development of Agents, E-Commerce, and in many real-world applications, and to publicize and extend AI technology to many domains in this fast moving information age.

The chairs of the workshops did an excellent job in bringing together many AI researchers and practitioners from the Pacific-Asia region and from all over the world. The well received workshops at PRICAI 2000 and the publication of this collection have convincingly shown the significance and practical impact of the work presented in this collection. Professor Nancy Reed's great effort in producing this special, fine collection will be applauded and appreciated by many. I am certain that this collection will stimulate more AI research and applications, influence many graduate students to conduct research and development in AI, and have a positive impact toward making our future better by creating an increasingly intelligent world.

June 2001 Huan Liu

Table of Contents

I Applications of Artificial Intelligence in Industry

II Artificial Intelligence in Electronic Commerce

III Intelligent Information Agents

Artificial Intelligence in Industry

Graham J. Williams[1] and Dickson Lukose[2]

[1] Enterprise Data Mining
CSIRO Mathematical and Information Sciences
GPO Box 664, Canberra 2601, ACT, Australia
Graham.Williams@cmis.csiro.au
[2] Mindbox Inc.
300 Drake's Landing, Suite 155
Greenbrae, CA 94904, USA
Dickson.Lukose@mindbox.com

Abstract. The Symposium on the Application of Artificial Intelligence in Industry was held in conjunction with the Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI-2000), Melbourne Australia, August 2000. It was the second of the Symposium series aiming to highlight actual applications of Artificial Intelligence in industry and to share and compare experiences in doing so. The Symposium brought together researchers and developers of applied Artificial Intelligence systems. The symposium is the leading forum in the Pacific Rim for the presentation of innovative applications of AI in industry.

1 Introduction

Artificial Intelligence techniques represent mature technology that have widespread application in many areas of industry. The application of Artificial Intelligence technology to industry has been happening, often quietly, for several decades. Success is often in making the techniques "disappear" into the functionality of the delivered system. Consequently, we do not always appreciate the wide application of AI.

In common with other computer systems, important issues related to building and deploying AI based production systems include: the development of domain-specific knowledge bases and ontologies; the development of domain specific tools, techniques and methods; systems architecture, systems and software engineering techniques and methodologies that are practical and effective; integration of AI systems with conventional database applications and legacy systems; the validation and verification of the functionalities and systems tuning for speed and efficiency. Other factors play a key role in determining the success of projects, including project, client, and management expectations, project planning, and social and cultural issues.

The objectives of the symposium were: to provide a forum for the discussion of innovative applications of AI in industry; to provide practising engineers exposure to and an evaluation of current research, tools, and practises in industry;

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 3–4, 2001.

to provide the research community a forum for exposing them to the problems of the practical applications of AI in industry; and to encourage the exchange of practical AI technologies and experience.

The selection of papers included in this volume brings together a collection that record varying degrees of application. Some propose techniques that are applicable to building industrial applications, while others report on industrial projects that are at various stages of implementation. Of particular interest are those that describe successful deployment in government and industry.

We trust you will find this selection of papers from the Symposium an interesting read and provide insights that you may be able to take advantage of in applying Artificial Intelligence solutions to real world problems.

2 Program Committee

We owe a considerable debt to the active members of the program committee for their efforts in carefully reviewing the papers and assisting the symposium chairs in their task of selecting papers that will be of interest to industry. The program committee consisted of:

- Leila Alem (CSIRO, Australia)
- Andrew Blair (BT Financial Group, Australia)
- Shu Heng Chen (National Chengchi University, Taiwan)
- Sung-Bae Cho (Yonsei University, Korea)
- Vladimir Estivill-Castro (Newcastle University, Australia)
- Brian Garner (Deakin University, Australia)
- Warwick Graco (Health Insurance Commission, Australia)
- Tim Hawes (Mindbox Inc, USA)
- Zhexue Huang (ETI, The University of Hong Kong)
- Philip Jackson (Intra - Team IT, UK)
- Ryszard Kowalczyk (CSIRO, Australia)
- Robert Kremer (University of Calgary, Canada)
- Chris Leckie (Telstra, Australia)
- Setsuo Ohsuga (Waseda University, Japan)
- William Richer (Mindbox Inc, USA)
- David Ripley (CSC/NASA, USA)
- Dharmendra Sharma (Univ South Pacific, Suva, Fiji)
- Robert Straughan (Inst for High Performance Computing, Singapore)
- Said Tabet (Mindbox Inc, USA)
- Ian Watson (University of Auckland, NZ)
- Wayne Wobcke (British Telecom, UK)

Applying Behavior-Oriented Robotics to a Mobile Security Device

Andreas Birk and Holger Kenn

Artificial Intelligence Laboratory
Vrije Universiteit Brussel
{cyrano, kenn}@arti.vub.ac.be

Abstract. The paper describes an industrial application of behavior-oriented robotics, the so-called RoboGuard. It is a mobile security device which is tightly integrated into the existing surveillance framework developed and marketed by Quadrox, a Belgian SME. RoboGuards are semi-autonomous mobile robots providing video streams via wireless Intranets to existing watchguard systems. RoboGuards fill several market-niches. Especially, they are a serious alternative to the standard approach of using Closed Circuit Television (CCTV) for surveillance. Low production cost and user friendliness were two important design targets for the actual mobile base of the RoboGuard, which were achieved by following the AI paradigm of behavior-oriented robotics.

1 Introduction

The RoboGuard allows remote monitoring through a mobile platform using onboard cameras and sensors. RoboGuards are supplements and often even alternatives to standard surveillance technology, namely Closed Circuit Television (CCTV) and sensor-triggered systems. The RoboGuard is a joint development between Quadrox [QUA], a Belgian security SME, and two academic partners, the AI-lab of the Flemish Free University of Brussels (VUB) and the Interuniversity Micro-Electronics Center (IMEC).

RoboGuards are tightly integrated into the existing range of products of Quadrox. This is an important aspect for the acceptance of any novel technology in well-established markets as customers are usually not willing to completely replace any existing infrastructure. For efficiency and security reasons, the RF-transmitted video-stream of the on-board cameras is compressed using a special wavelet-encoding [DC97]. The IMEC is the responsible partner for this feature of RoboGuard. The mobile base and its control, which are the main focus of this paper, are at the hands of the VUB AI-lab. The VUB AI-lab has a long tradition since the mid-1980s in basic research in the domain of behavior-oriented robotics. The expertise of the VUB AI-lab in this domain includes conceptual aspects as well as technological know-how, which are both incorporated in the RoboGuard system.

In accordance with recent interest in service robotics [Eng89], there has also been previous work on security robots. This work is widely scattered, ranging from unmanned gunned vehicles for military reconnaissance operations [AHE+90] to theoretical research on reasoning within decision-theoretic models of security [MF99]. Our approach deals

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 5–15, 2001.

with a system operating in semi-structured environments under human control and which is a product, i.e., it must be competitive to existing alternative solutions for the task.

The rest of this paper is structured as follows. In section two, a short introduction to the field of behavior-oriented AI is given and its relation to the RoboGuard project is explained. Section three presents the hardware aspects of a RoboGuard. In section four, the software side of the RoboGuards is introduced. Section five concludes the paper.

2 AI and Robotics

"Classical" robots as we find them today in industry rely very much on precise mechanics. This is necessary as these robots rely on exact models, for example to describe and compute their kinematics. The need for extreme precision puts high demands on the parts, the assembling, and the maintenance of these robots. The field of behavior-oriented robotics or the "artificial life road to AI" [Ste94a] on the other hand has come up with robotic systems which are much less demanding in this respect. The model-based control schemes are substitute by reactive ones [Bro86,Ste91], which establish close, dynamic couplings between sensors and motors. Instead of using complex models, sensor information is exploited as, as Rodney Brooks put it, "the world is its best own model" [Bro91]. In addition to the much lower need for precision, there is a lower need for computation power as well, as reactive control schemes are usually computationally much cheaper than model-based control schemes like for example inverse kinematics.

Behavior-oriented robotics profits mainly from recent advances in sensor technology as there are more and more mass-produced, inexpensive sensors available for a multitude of different applications. Especially, advances in vision related hardware allow for some various sensing tasks at low cost. In combination with the tremendous savings at the mechanical part of the device, a behavior-oriented robot can be produced much cheaper than its classical counterpart [Bir98]. This cost effectiveness is an important feature for RoboGuard.

In addition, RoboGuard needs several functionalities which are directly related to artificial life (Alife) [Lan89] and research on animats (= animal + robot) [Wil91]. For example, RoboGuard has to be self-sufficient, i.e., it has to be able to sustain itself over extended periods of time in respect to energy [Pfe96]. In this context, basic research in an artificial robotic ecosystem [Ste94b,McF94,Bir97,BB98] turned out to be very useful for this practical application.

A further advantage of employing behavior-oriented concepts in RoboGuard is the facilitation of semi-autonomy. In contrast to the naive intuition, including a human operator in the control loop of the base can make the task more complex. It is very difficult, if not impossible for a human teleoperator to efficiently steer a mobile base with video-streams from a on-board camera only. Operators do not take the current speed and momentum of the base into account, they neglect possible delays, they have difficulties to develop a feeling for the size of the base, and so on. In addition, the mobile base has to be protected from accidental or malicious misuse.

Shortly, the mobile base needs an advanced system for navigation and steering support including obstacle avoidance. The fusion of operator steering commands, autonomous drive and navigation functionality, as well as domain-specific plausibility and

safety checks is a non-trivial task. For this purpose, the modular approach of using behaviors is especially suited. It also turned out that we could strongly benefit in this respect from insights gathered in the domain of robot soccer [BKW00,BWBK99,BWB+98].

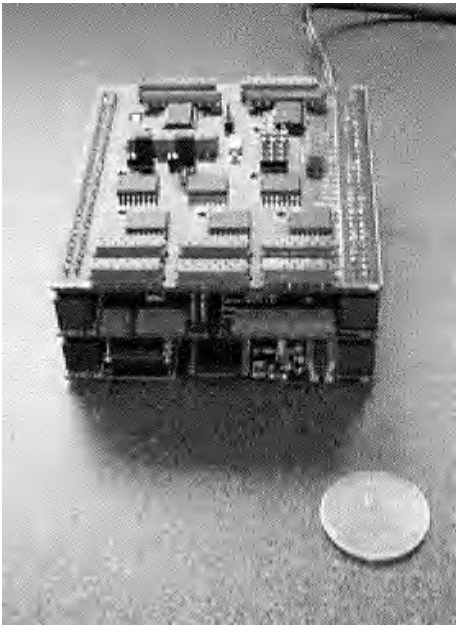


Fig. 1. A picture of the RoboCube, an extremely compact embedded computer for robot control.

3 The Implementation of the Mobile Base

3.1 The RoboCube as Controller-Core

The basic Sensor-Motor-Control of RoboGuard centers at the side of electronics around the RoboCube, a special robot control hardware developed at the VUB AI-lab. The VUB AI-lab has quite some tradition in developing flexible robot control hardware. Various experimental platforms have been build in the lab starting from the mid-eighties up until now. They were used in basic research on behavior-oriented architectures as well as for educational purposes. In all applications it was important that researcher or students could easily add, change, and replace components. Over the years, experiences with approaches based on embedded PCs and different micro-controllers were gathered and lead to the *Sensor-Motor-Brick II (SMBII)*[Ver96]. The SMBII is based on a commercial board manufactured by Vesta-technology providing the computational core with a Motorola MC68332, 256K RAM, and 128K EPROM. Stacked on top of the Vesta-core, a second board provides the hardware for sensor-, motor-, and communication-interfaces.

The so-called RoboCube (figure 1) is an enhanced successor of the SMBII. In RoboCube, the commercial computational core is replaced by an own design, also based on the MC68332, which saves significant costs and simplifies its architecture. In addition, the physical shape of RoboCube is quite different from the one of the SMBII. First, board-area is minimized by using SMD-components in RoboCube. Second, three boards are stacked on each other leading to a more cubic design compared to the flat but long shape of the SMBII, hence its name Robo*Cube*.

RoboCube has a open bus architecture which allows to add "infinitely" many sensor/motor-interfaces (at the price of bandwidth). But for most applications the standard set of interfaces should be more than enough. RoboCube's basic set of ports consists of

- 24 analog/digital (A/D) converter,
- 6 digital/analog (D/A) converter,
- 16 binary Input/Output (binI/O),
- 5 binary Inputs,
- 10 timer channels (TPC),
- 3 DC-motor controller with pulse-accumulation (PAC)

The RoboCube is described in more detail in [BKW00,BKW98].

3.2 The Developments of the Mobile Base

When developing and integrating the different hardware components of the mobile base, it was necessary to engineer specific aspects through several iterated test and developments. For example, it is necessary to exactly adapt the drive-units (with motors, gears, encoders, wheels, etc.) to achieve a maximal performance at minimal cost. The same holds for the power-system and all other sub-units of the base. The following two bases serve as an example of this process of constant adaption and improvement. At the moment, the second base is produced in a small series to be used as RoboGuards.

The basic hardware aspects are the mobile platform inculding the RoboCube controller, the motor drivers, the support frame, the power system and the energy management. All these factors are strongly interdependent. In addition, they are strongly affect by the type of main-computer supplementing the RoboCube as this main-computer strongly affects the power consumption. The main-computer is used for the high-level computations, especially the image acquisition, the image compression and the communication on board of the robot. Due to an adaptation to the developments of the computer market, the type of main-computer on the robot was changed and therefore there were significant changes within the base-design between the first and the second version.

3.3 The First Mobile Platform and Base

The most significant feature of the first version of the base (figure 2) is the usage of a network computer, namely the Corel Netwinder (figure 3). At the beginning of the project, network computers seemed to be a promising technology especially in respect

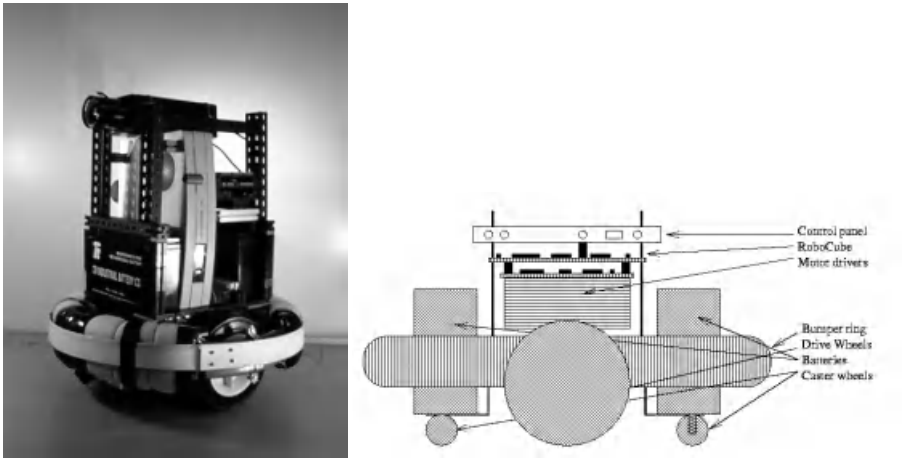


Fig. 2. The first base for RoboGuard (left) and a schematic drawing of its mobile platform (right).

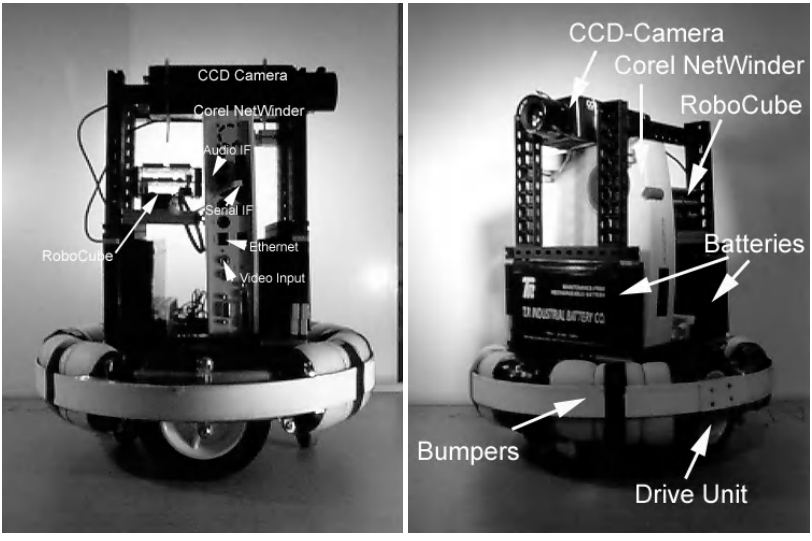


Fig. 3. The first version of the RoboGuard base includes a network-computer, the Corel Netwinder.

to this project. The Corel netwinder is very compact, offers many default interfaces, and it has a very low power-consumption.

But its computing power is not sufficient for the needs of this project. Furthermore, it is questionable if this trait of computers will survive the fast current developments in the market. To guarantee availability and increase in performance for the future, it was seen necessary to switch to a PC-based approach. This implied that the drive- and power-system of this first base were much too small. They had to be severely adapted for the

next version. But the general development of motor-drivers and the control-electronics were already successfully completed on this base.

3.4 The Second Mobile Platform and Base

As mentioned above, it was decided to move to a PC-based approach for future developments instead of using a low-power but also low performance network computer. This change had also many implications for the mobile base.

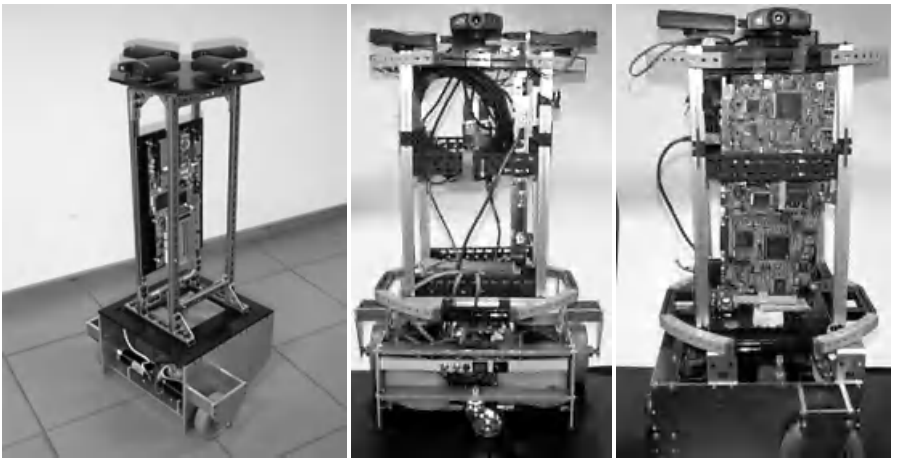


Fig. 4. The inside core of the second version of the RoboGuard base. It includes a mobile PC-board and four color-cameras allowing full 360 degrees surveillance.

The second version of the mobile platform (figure 4) and base was developed with several intermediate tests and changes. It is already a very matured version, i.e., there will be no or only minor changes to its low-level functionality for future versions. As mentioned above, a small series of these bases is produced at the moment to be used in RoboGuards.

4 The Software Side

4.1 Overview

The RoboGuard control software's task is the low-level control of the RoboGuard Base as well as several forms of support for the operator. Ideally, the operator has the impression that he or she is in full control while the system autonomously takes care of crucial tasks like obstacle avoidance, keeping on a trajectory, emergency stops, and so on. The software architecture is structured into several layers (figure 5), each allowing several modules or behaviors to run in (simulated) parallel.

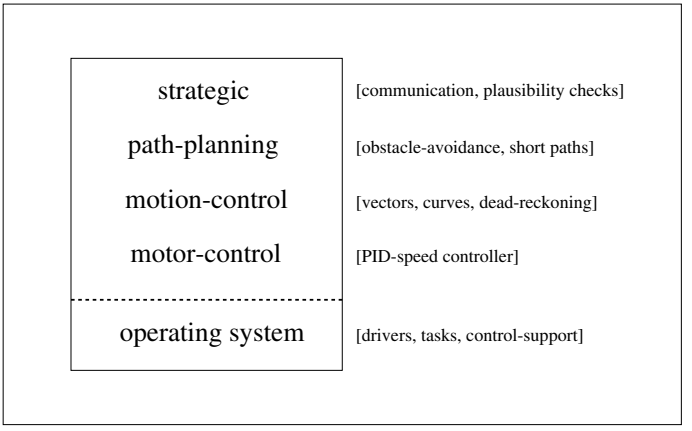


Fig. 5. The software architecture of RoboGuard's mobile base. It is a layered architecture where several modules or behaviors run in simulated parallel.

4.2 CubeOS and Control

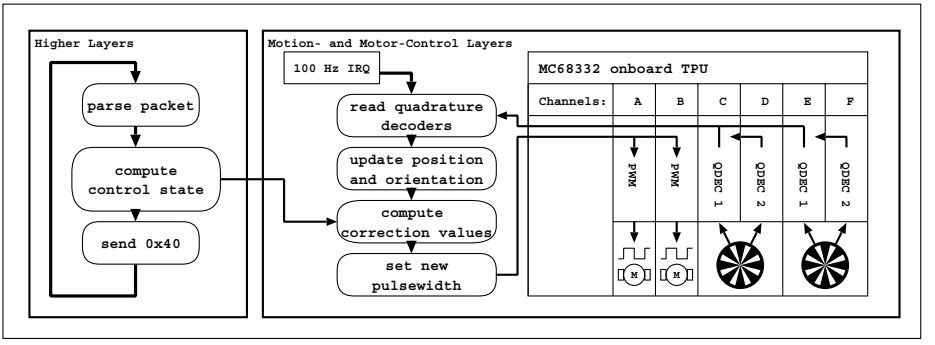
The RoboGuard control software relies on the RoboCube controller platform and on it's CubeOS operating system to implement the control application. The CubeOS nanokernel contains real-time multi-threading, abstract communication interfaces and thread control primitives. On top of the nanocore, a set of software drivers provides an application programming interface to the RoboCube's hardware.

The RoboGuard controller makes use of these drivers to control the MC68332 TPU on the RoboCube. The TPU is the main hardware interface for motor control and odometry. The TPU has 16 independent Channels which can be programmed to various functions. In the RoboGuard, four of these channels are used for odometry, two channels form a quadrature decoder. This quadrature decoder represents an up/down impulse counter that is controlled by the encoders on the motor axis. The CubeOS TPU driver configures two TP channels to form the decoder by linking them together in QDEC mode. Motor control is implemented by the TPU's pulse width modulation function. Again, the CubeOS TPU driver prepares one TPU channel per motor to generate a fixed-frequency square waveform with variable duty cycle that is controlled by the controller application.

The communication with the onboard PC makes use of the serial communication driver in CubeOS. It provides queued input and output to the application as well as platform-independent data encoding (XDR). Upon initialization, the controller application initializes the TPU driver which in turn initializes the TPU hardware and sets up the channel functions. The control information for the mobile base state and the odometry position is reset to orientation 0 degrees, position (0,0), speed 0.

Then, the control thread is configured to be restarted every 25 msec. This function is also executed by the TPU. Then the communication thread starts up and waits for incoming packets on the serial communication link that is connected to the onboard PC of the RoboGuard mobile base. Upon proper reception, the content of each packet is translated into control commands for the control task.

The control task is invoked every 25 msec. It first reads the QDEC decoder TP channels and computes a new base position and orientation. Then, the control command from the communication thread is compared with the current state of the base and correction values are computed which are then used to change the PWM settings of the motors. The control task then exits.



4.3 Motion- and Motor-Control

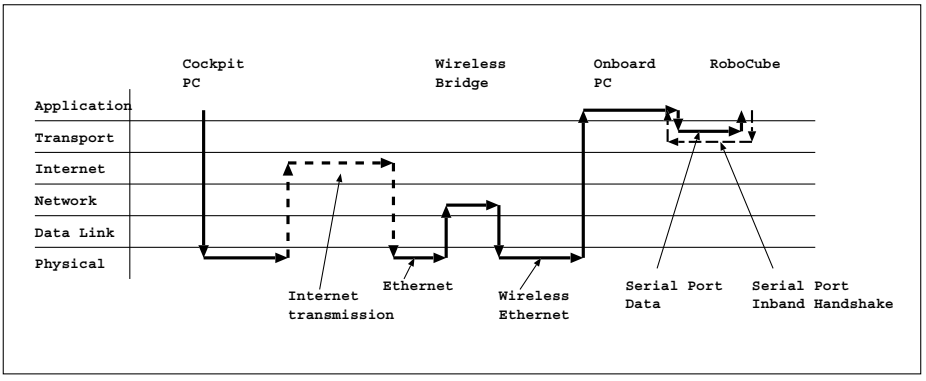
The motion layer interacts with the higher layers via a shared memory buffer that is written by a higher thread and is read by the motion thread (figure 4.2). The write operation is made atomic by delaying the execution of the motion thread during write operations. So, target-values in the motion-controller can be asynchronously set by higher level behaviors. The motion-controller so-to-say transforms the target-values on basis of odometric data to appropriate target-values for the motor-control. The motion- and motor-control layers are based on generic software modules for differential drive robots, featuring

- PID-control of wheel speed
- odometric position- and orientation-tracking
- rotational and translational control

4.4 The Strategic and Path-Planning Layers

A core function on these layers is operator communication, i.e., the transmission of control states from the operator's console or so-called cockpit to the control hardware. To ensure a low-latency operation over the Internet link, a protocol based on UDP packets has been implemented. The packets are formed at the cockpit by synchronous evaluation of the control state and transmission to the onboard PC of the RoboGuard platform via Internet. Here, they are received and transmitted to the RoboCube via the serial port. The communication behavior parses the packets and makes its content available to other behaviors via shared memory.

To ensure low-latency-operation, there is no retransmission on lost packets although UDP does not guarantee successful delivery of packets. However, since packets are



transmitted synchronously and are only containing state information, there is no need to resend a lost packet since the following packet will contain updated state information. By exploiting this property of the protocol, low-latency operation can be assumed.

The communication between the RoboCube and the onboard PC uses inband handshaking to prevent buffer overruns in the RoboCube software. The communication layer software in the RoboCube confirms every packet with a 0x40 control code. Only if this control code has been received, the onboard PC communication layer software transmits the next packet. If the RoboCube communication layer software did not yet confirm a packet when a new packet arrives from the Internet transport layer, this packet is discarded so that the control layer software only receives recent packets, again ensuring low-latency operation.

Plausibility checks on the same layer can be used to discard packets or to modify the implications of the information they contain. This is done in a rule-based module. This functionality is optional and allows a convenient incorporation of background knowledge about particular application domains. The strategic layer also includes self-sufficiency behaviors like energy-management. Depending on the preferences of the customer, the arbitration of these behaviors can be handled in the rule-base. For example, a low-priority mission could be autonomously aborted if the base is likely to run out of energy during its execution.

The path-planning layer handles functionality to facilitate the operation of the base. It can incorporate world-knowledge on different scales, again depending on the preferences of the customer. Its simplest functionality consists path stabilization, i.e., jitters from the manual control can be smoothed away by temporal filtering. Behaviors for obstacle avoidance protect the system from accidental or malicious misuse, and help to move along narrow hallways and cluttered environments. Last but not least, it is possible to let the base navigate completely on its own when detailed world-knowledge in form of maps is provided.

5 Conclusion

The paper described the mobile base of RoboGuard, a commercial surveillance device. The base is developed with concepts and technology from behavior-oriented robotics,

leading to low production costs and high user friendliness. Unlike "classical" robots, the base does not rely on extremely precise mechanics and detailed models. Instead, tight couplings between sensors and motors through software modules or behaviors running in (virtual) parallel are used. This allows a smooth fusion of steering commands from a human operator with autonomous functionalities like obstacle avoidance and motion stabilization. Furthermore, the generic modularity of a behavior-oriented approach facilitates customization to the specific needs of a particular customer.

References

[AHE+90] W.A. Aviles, T.W. Hughes, H.R. Everett, A.Y. Umeda, S.W. Martin, A.H. Koyamatsu, M.R. Solorzano, R.T. Laird, and S.P. McArthur. Issues in mobile robotics: The unmanned ground vehicle program teleoperated vehicle. In *SPIE Mobile Robots V*, pages 587–597, 1990.

[BB98] Andreas Birk and Tony Belpaeme. A multi-agent-system based on heterogeneous robots. In Alexis Drogoul, Milind Tambe, and Toshio Fukuda, editors, *Collective Robotics, CRW'98*, LNAI 1456. Springer, 1998.

[Bir97] Andreas Birk. Autonomous recharging of mobile robots. In *Proceedings of the 30th International Symposium on Automative Technology and Automation*, 1997.

[Bir98] Andreas Birk. Behavior-based robotics, its scope and its prospects. In *Proc. of The 24th Annual Conference of the IEEE Industrial Electronics*. IEEE Press, 1998.

[BKW98] Andreas Birk, Holger Kenn, and Thomas Walle. Robocube: an "universal" "special-purpose" hardware for the robocup small robots league. In *4th International Symposium on Distributed Autonomous Robotic Systems*. Springer, 1998.

[BKW00] Andreas Birk, Holger Kenn, and Thomas Walle. On-board control in the robocup small robots league. *Advanced Robotics Journal*, 14(1):27 – 36, 2000.

[Bro86] Rodney Brooks. Achieving artificial intelligence through building robots. Technical Report AI memo 899, MIT AI-lab, 1986.

[Bro91] Rodney Brooks. Intelligence without reason. In *Proc. of IJCAI-91*. Morgan Kaufmann, San Mateo, 1991.

[BWB+98] Andreas Birk, Thomas Walle, Tony Belpaeme, Johan Parent, Tom De Vlaminck, and Holger Kenn. The small league robocup team of the vub ai-lab. In *Proc. of The Second International Workshop on RoboCup*. Springer, 1998.

[BWBK99] Andreas Birk, Thomas Walle, Tony Belpaeme, and Holger Kenn. The vub ai-lab robocup'99 small league team. In *Proc. of the Third RoboCup*. Springer, 1999.

[DC97] S. Dewitte and J. Cornelis. Lossless integer wavelet transform. *IEEE Signal Processing Letters 4*, pages 158–160, 1997.

[Eng89] Joseph F. Engelberger. *Robotics in Service*. MIT Press, Cambridge, Massachusetts, 1989.

[Lan89] Christopher C. Langton. Artificial life. In Christopher G. Langton, editor, *Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE '87)*, volume 6 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 1–48, Redwood City, CA, USA, September 1989. Addison-Wesley.

[McF94] David McFarland. Towards robot cooperation. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From Animals to Animats 3. Proc. of the Third International Conference on Simulation of Adaptive Behavior*. The MIT Press/Bradford Books, Cambridge, 1994.

[MF99] N. Massios and Voorbraak F. Hierarchical decision-theoretic robotic surveillance. In *IJCAI'99 Workshop on Reasoning with Uncertainty in Robot Navigation*, pages 23–33, 1999.

[Pfe96] Rolf Pfeifer. Building "fungus eaters": Design principles of autonomous agents. In *From Animals to Animats. Proc. of the Fourth International Conference on Simulation of Adaptive Behavior*. The MIT Press/Bradford Books, Cambridge, 1996.

[QUA] The quadrox website. http://www.quadrox.be.

[Ste91] Luc Steels. Towards a theory of emergent functionality. In Jean-Arcady Meyer and Steward W. Wilson, editors, *From Animals to Animats. Proc. of the First International Conference on Simulation of Adaptive Behavior*. The MIT Press/Bradford Books, Cambridge, 1991.

[Ste94a] Luc Steels. The artificial life roots of artificial intelligence. *Artificial Life Journal*, 1(1), 1994.

[Ste94b] Luc Steels. A case study in the behavior-oriented design of autonomous agents. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From Animals to Animats 3. Proc. of the Third International Conference on Simulation of Adaptive Behavior*. The MIT Press/Bradford Books, Cambridge, 1994.

[Ver96] Dany Vereertbrugghen. Design and implementation of a second generation sensor-motor control unit for mobile robots. Technical Report Thesis, Tweede Licentie Toegepaste Informatica, Vrije Universiteit Brussel, AI-lab, 1996.

[Wil91] Stewart W. Wilson. The animat path to ai. In *From Animals to Animats. Proc. of the First International Conference on Simulation of Adaptive Behavior*. The MIT Press/Bradford Books, Cambridge, 1991.

Ontology Design and Its Application in the Petroleum Remediation Domain

Lin-Li Chen[1] and Christine W. Chan[2]

[1] University of Regina, Faculty of Engineering,
Saskatchewan, Canada S4S 0A2
Chenli@uregina.ca
[2] University of Regina, Department of Computer science,
Saskatchewan, Canada S4S 0A2
Chan@cs.uregina.ca

Abstract. The purpose of this paper is to suggest a process for ontology design and its application. Current knowledge modeling methodologies tend to focus on the particular task, which makes them difficult to be reused. In this paper, we propose three different ontology models suitable for modeling large scale type domains. We apply the models to the problem domain of petroleum waste management, and discuss knowledge representation at the different levels of ontologies.

1. Introduction

Ontologies provide terms for describing knowledge about a domain, and is understood as an agreed upon vocabulary of common terms and meanings within a group of people. Most research on ontologies focuses on what one might characterize as domain factual knowledge because knowledge of that type is particularly useful in natural-language understanding [1].

The communication and knowledge sharing problems are important between users and organizations. To reduce the gap of misunderstanding, an agreement must be achieved about the shared knowledge among users and software. The objective of this project is to design a flexible multi-level ontology for different application purposes. We focus on designing different levels (models) of ontologies to support users' needs, and we apply this ontology design to the problem domain of petroleum remediation.

The rest of this paper is organized as follows. Section 2 introduces background literature on ontology design. Section 3 presents the design method and process. Section 4 describes the problem domain for building an ontology. Section 5 introduces the ontology tool and some implementation results. Finally, we briefly conclude the paper.

2. Background Literature on Ontology Design

Noy [4][5] describes previous projects related to fundamental principles for ontology design. The CYC project [3] attempts to create a general ontology for common sense

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 16–23, 2001.

knowledge, and contain more than 10,000 concept types used in the rules and facts encoded in the knowledge base. The upper level of the CYC hierarchy is the "thing", which is separated into three sub-classes: individualObject, intangible, and representedThing. The hierarchy under "thing" is that not all the sub-categories are exclusive. Sowa's Ontology [7] states his fundamental principles for ontology design as "distinctions, combinations, and constraints". He uses philosophical motivation as the basis for his categorization. The upper level is a category for every possible combination of distinctions. At the lower level, it uses constraints to rule out categories that cannot exist. In our ontology design, we also distinguish among different levels of models according the content and use of knowledge.

3. Design Method for Ontology

Building ontologies involve many steps as shown in Figure 1.

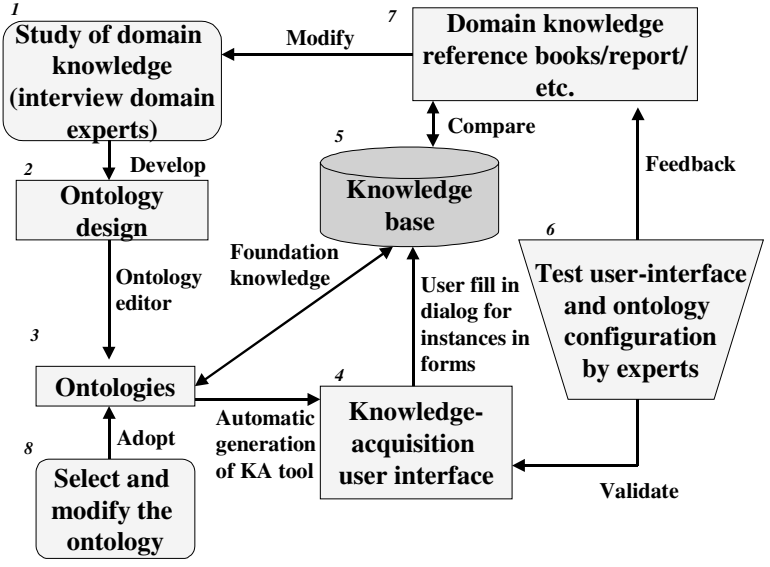


Fig. 1. Basic architecture of remediation ontology

In Figure 1, (1) the domain knowledge is obtained through either relevant document study or interviews with domain experts. Knowledge analysis proceeds after the knowledge is collected from experts. (2) After the knowledge is obtained and analyzed, an initial ontology design can be developed. (3) A proper ontology development tool is chosen and the ontology editor is used to create ontologies. Ontologies are foundation knowledge of knowledge bases. (4) To develop the petroleum remediation domain ontology, an ontology tool called Protégé-2000 was used. This tool can automatically generate a knowledge acquisition tool, which supports a user interface for domain experts to fill in information according to the ontology structure. The details of the tool and interface will be discussed in section 5-

2. (5) A knowledge base can be built based on ontologies and user input from the knowledge acquisition user interface. (6) The user can evaluate this ontology by testing the definition of terms and concepts of domain knowledge. Experts or users can validate the knowledge user interface when they fill in the domain knowledge. In addition, the knowledge base must satisfy the published reports and references. (7) After defects have been reported by the users, or if the resulting knowledge base does not match with the published related reports, the ontology structure must be re-designed through studying the necessary documents and continuing discussions with experts. (8) To create other ontologies, the user can simply adopt an ontology similar to the new problem domain from step 3 (Ontologies). The user can modify and adapt the middle level (domain or task ontologies) and domain specific level to suit the new domain. This process is described in detail as follows.

3.1 Design Process

Step 1: Domain study and initial ontology design: The purpose of this step is to study published remediation literature, reference books, textbooks, and government reports, which should contain descriptions of standard protocols for selection of remediation procedures. These resources can be used to identify issues in representing remediation knowledge.

Step 2: Extend and improve initial ontology design: The second step is to use some cases from published papers or interview domain experts for a more precise and detailed ontology design. The focus of this step is to extend the information or knowledge in the ontology structure.

Step 3: Ontology re-design: In the ontology development process, re-designing the ontology continues for the duration of the task or project. Changes will often occur in the ontology's structure because a new factor is added in the task or process. In addition, if the result does not satisfy the relevant documentation on the problem domain, the ontology has to be re-designed.

Step 4: Ontology Evaluation: Evaluating an ontology is one of the most difficult areas in ontology design. In general, there are two aspects to ontology evaluation [4][5]: (1) evaluating formal features of an ontology: such as consistency, well-formedness, completeness of definitions, etc., and (2) evaluating utility and usability of ontology, i.e., how adequate it is for the task it was created for, how well it represents the domain of interest. In our work, we focus on the second aspect because whether ontology covers every aspect of its intended domain can be judged differently by different experts. Therefore, for the remediation ontology developed, we conducted tests to determine whether the ontology can completely represent the domain knowledge in the user interface.

The evaluation processes are described as following:

1. First we must test the conceptual coverage and practical usefulness of this ontology. This involves considerations such as what percentage of sentences could be represented fully, and testing the feasibility of reuse based on the ontology structure and representation. The data or information from the knowledge base design has to be found which allows reuse and transfer to similar technical domains. Thus, we make use of an ontology at different levels (Figure 2).

2. Next we test understanding of the user interface for knowledge acquisition-layout of a user interface, configuration of ontology, knowledge contained, etc. This examination measures whether the user understands the purpose of the user interface layout, and whether this configuration of the user interface allows easy access for users to fill in information.
4. Finally, we can test the result of knowledge base.

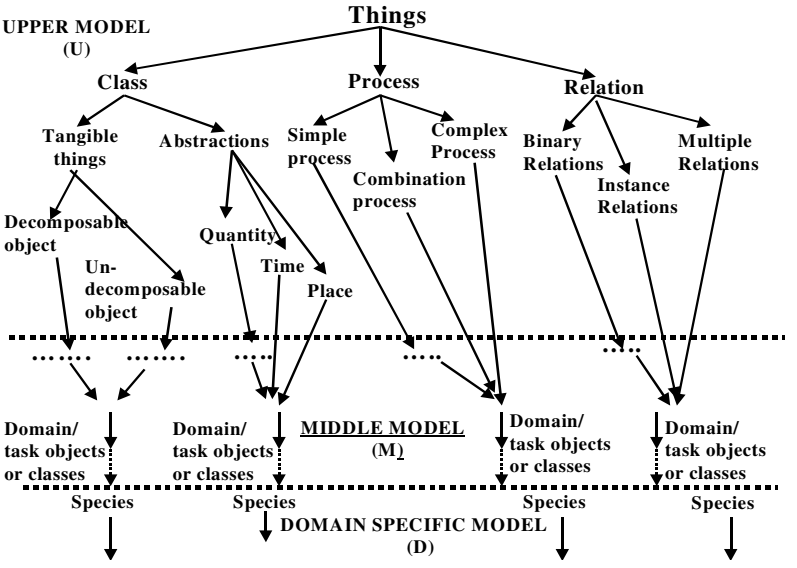


Fig. 2. Example of classification of different level of knowledge representation

3.2 Design Template

The structure that we used for developing the remediation ontology is shown in Figure 2. The upper model represents general knowledge, which is domain and task independent. The upper model (U) offers the possibility for defining objects (classes) and properties by supplying a hierarchy of general concepts. There are three types of ontologies: class, process, and relation ontology.

The middle model (M) contains principles and concepts, which consists of common sense relevant to the domain of interest. This level contains concepts pertaining to different types of connections between objects (classes), such as is-a, has, in, and part-of relations, etc. For example, groundwater is a type of water. The detail of the ontology structure is shown in Figure 3, which shows a simple example of the classification of different items under "Things". It illustrates that in the petroleum remediation domain, the ontology has a category for *liquid*, which is the parent category of *water*, which in turn is the parent of *groundwater*. *Groundwater type* is an attribute of the class of groundwater. *Potable water*, *non-potable water*, and *fresh water* are values of the attribute "groundwater type". A species link between a category and its attributes can only exist at the lowest level of the category hierarchy,

which is between groundwater and groundwater type in the example. All the links between classes are is-a links (relationships).

While the domain specific knowledge cannot be reused in another domain, the middle model can be reused. The domain specific model (D) can be integrated into the ontology by using the classes and properties in the middle model. Protégé-2000 supports developing ontologies, automatically generating the knowledge acquisition tool, and converting the ontology to a database for further information retrieval, querying, and problem solving. Before describing the ontology developed, we present the application domain in the next section.

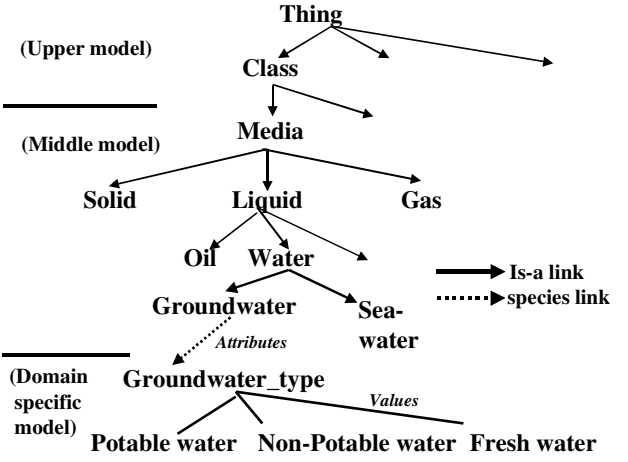


Fig. 3. Example of sub-hierarchy of media that illustrates is-a and species links in a category

4. Application of Method to Building an Ontology: Problem Domain

Petroleum contaminants are the chemical substances in petroleum that can cause serious pollution to our environment. Significant attention has been paid to the problems of petroleum-contaminated soil and groundwater. The contaminated sites are contaminated primarily due to nonaqueous phase liquids (NAPLs), which are typically classified as either light nonaqueous (LNAPLs) or dense nonaqueous phase liquids (DNAPLs), and leakage and spillage of petroleum-related facilities such as storage tanks and pipelines. The contaminants leak and spill from the gas tank to the first layer of soil. Then, with time, the contaminants seep through the soil to the deeper layer, the groundwater layer.

The aim of petroleum waste management is contaminant remediation. The environmental engineering experts must make a decision as to whether a particular action must be undertaken either to control or reduce the contaminant in the soil and groundwater. However, contaminated sites have different characteristics depending

on pollutants' properties. Therefore, the selection of remediation techniques for different site conditions varies significantly. The decision for a suitable method at a given site often requires expertise on both remediation technologies and site hydrological conditions [6].

5. Implementation of Ontology

5.1 Implementation Tool

Protégé-2000 was the tool adopted for implementation. It can automatically generate a knowledge acquisition tool, which presents a user interface for domain experts to fill in information according to the ontology structure.

Protégé-2000, developed by Stanford Medical Informatics of Stanford University, is a Java-based implementation tool that uses domain ontologies as the basis for generation of knowledge-acquisition tools. Computer science researchers use the word ontology to denote different types of models. In Protégé-2000, the models are based on class hierarchies. Protégé-2000 supports the development of problem solvers from reusable components, and the generation of a domain-specific knowledge-acquisition tool with a meta-tool. With Protégé-2000, the developer can create a domain ontology and use the ontology as the basis for generating a knowledge-acquisition tool with a meta-tool. Finally, the knowledge-acquisition tool can be used to create instances of the domain ontology [2].

5.2 Implementation Results

Figure 4 shows the three different levels of ontologies in the implementation: Remediation ontology, Remediation technical ontology, and Remediation extend ontology.

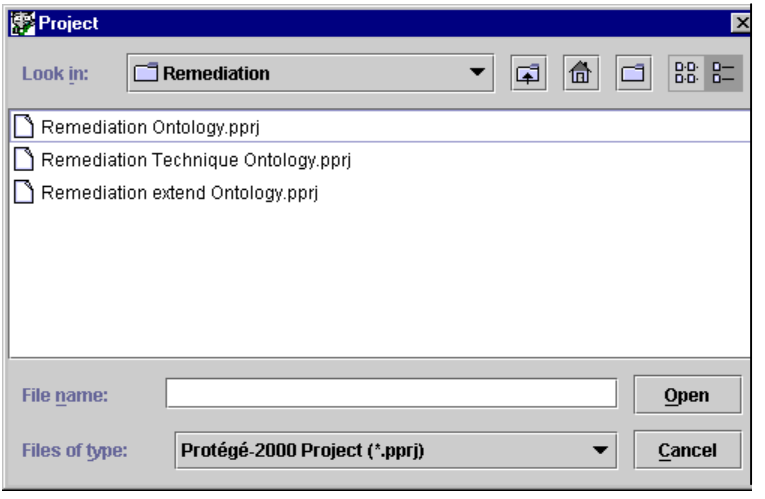


Fig. 4. Example of different levels of ontology hierarchy

The structure of remediation ontology focuses on general knowledge, which can be easily shared by other users or groups. It corresponds to the upper model. The remediation technique ontology represents the common knowledge (Figure 5) that corresponds to the upper and middle models, and these do not include detailed knowledge. For example, the class of chemical-substance has two meta-classes: organic-substance, non-organic-substance. In common sense, each of these sub-classes can be distinguished into two different sub-compounds, toxic and non-toxic. However, we do not place attributes such as carbon, THP, and so on. The rationale is that the sub-compounds still belong to the middle model and not the domain model. Only domain model has attributes because this model is unlikely to be reused.

The domain model is shown in Figure 6. After the user chooses the concept of interest, in this case, groundwater, the frame corresponding to this concept appears on the right-hand side of the Figure 6. This frame allows the user to fill in some of the attributes of the concept that need to be present in the specific domain knowledge. For example, the class of groundwater has the attributes of flow, hydraulic conductivity, pH, and so on. The sub-classes such as potable water and non-potable water also share these properties.

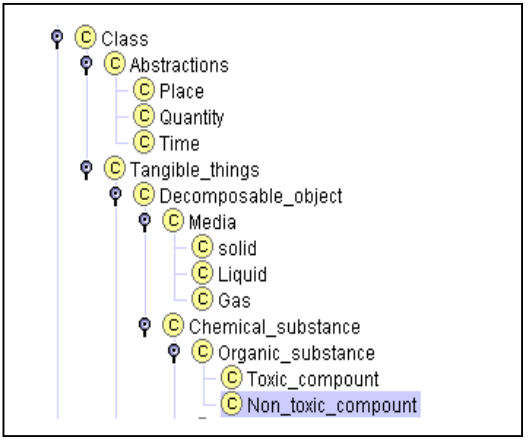


Fig. 5. Example of middle model

6 Conclusion

We have proposed in this paper an ontology design method and illustrated the method by applying it to the domain of petroleum remediation. Some results from the implemented ontology are also presented.

Acknowledgement. We are grateful for the financial support of Natural Sciences and Engineering Research Council (NSERC) of Canada.

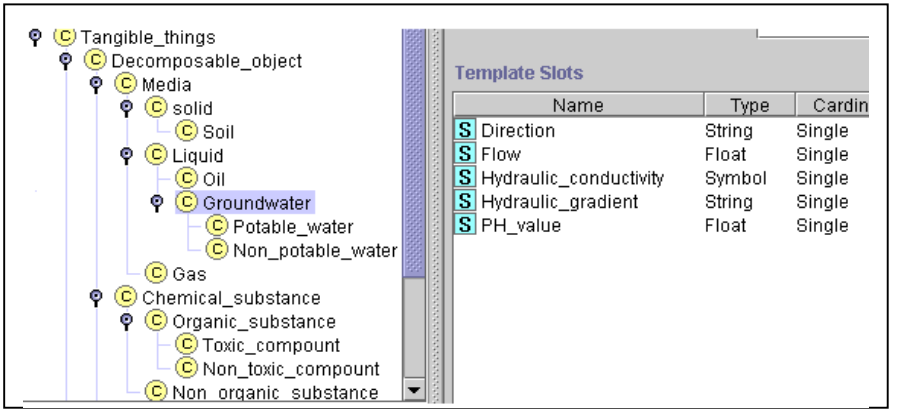


Fig. 6. Example of domain specific model

References

1. Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R., "What are Ontologies, and Why do we Need Them?", *IEEE Intelligent Systems*, Vol.14, No.1, pp.20-26,1999.
2. Eriksson, H., Fergerson, R. W., Shahar, Y., and Musen, M. A., "Automatic Generation of Ontology Editors", *Twelfth Banff knowledge Acquisition for Knowledge-bases System Workshop*, Banff, Alberta, Canada, 1999.
3. Lenat, D. B., Guha, R. V., Pittman, K., Pratt, D., and Shepherd, M., "CYC: Toward Programs with Common Sense", *CACM*, Vol.33, No.8, pp.30-49, 1990.
4. Noy, F. N., "The State of the art in Ontology Design: a Survey and Comparative Review", *AI Magazine*, Vol.18, No. 3, pp.53-74, 1997.
5. Noy, F. N., "Knowledge Representation for Intelligent Information Retrieval in Experimental Science", Ph.D thesis, College of Computer Science Northeastern University, Boston, MA02115, 1997.
6. Sims, J. L., Suflita, J. M., and Russell, H. H., "In-situ Bioremediation of Contaminated Ground Water", *Ground Water Issue*, EPA/540/S-92/003, U.S.EPA, R.S. Kerr Environ. Res. Lab., Ada, OK, pp.11, 1992.
7. Sowa, J., "Top-level Ontological Categories", *International Journal of Human-Computer Studies*, Vol.43, No.5/6, pp.669-686, 1995.

A Sales Agent for Website Personalization

Weiqin Chen[1], Ryan Shaw[1], Leif Mortenson[1], Tom Foley[1],
and Riichiro Mizoguchi[2]

[1] Silver Egg Technology Co., Ltd., Daidomon Honbu Bldg. 5F, 9-22 Toyotsu-cho, Suita, Osaka, Japan
{wendy, ryan, leif, tom}@Silveregg.co.jp
[2] Institute of Scientific and Industrial Research, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567, Japan
miz@ei.sanken.oska-u.ac.jp

Abstract. In this paper we present the design and prototype implementation of a real-time, adaptive, ontology-based sales agent and recommendation engine. It supports the personalization of Internet services by taking into account product knowledge, sales expertise and customer preferences. Using a hybrid of ontology engineering and machine learning techniques, the sales agent resolves the start-up and knowledge management problems inherent in other web personalization technologies. It provides for the dynamic adaptation of customer profiles based on behavioral data and domain knowledge models. This approach is domain-independent and applicable to a wide-variety of web commerce and services sites.

1 Introduction

Knowing which products meet the needs of which customer is critical to the success of most businesses. Before the Internet, companies managed customer knowledge indirectly through their sales force. Each salesperson learned his/her customer's needs and satisfied those customers with individualized services. For example, a salesperson might apply his/her experience to map from the inferred needs of a customer and recommend a product likely to satisfy the customer.

In obviating the traditional salesperson, the Internet has created a customer knowledge management gap. Without salespeople, Internet merchants are deprived of the traditional means of knowing and satisfying their customers. New methods of acquiring and managing customer knowledge, and then applying this knowledge to provide individualized service, are required. We therefore present an ontology-based sales agent that fills the Internet knowledge management gap. It will provide efficient adaptive personalization based on a hybrid of AI techniques including ontology engineering and machine learning.

This paper is organized as follows. Following this introduction, Section 2 presents problems with other web personalization approaches and introduces the solution of our

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 24–30, 2001.

sales agent. Its structure, the features of its component tools, and the AI technologies used to implement them are discussed in section 3. Section 4 supplies our conclusion and suggestions for further consideration.

2 Problem Statement and the Solution of the Sales Agent

This section discusses the problems existing in current web personalization approaches, and introduces our solution.

2.1 Current Web Personalization Situation

In the last several years, the Internet industry has seen many attempts to build customer loyalty through personalized content. Leading websites like Amazon.com have begun to offer personalized product recommendations based on customer profiling. In general, personalization involves profiling customers and then customizing a response, such as making a purchase recommendation, based on the customer's profiled characteristics.

Rule-based methods are a popular method of personalization. While powerful, this approach is open-loop. Expensive experts and engineers must intervene to acquire knowledge and manually devise rules in order to modify the response of the system.

A recent alternative is collaborative filtering [8][11], an adaptive method which correlates responses to similar customers automatically. For example, the purchasing patterns of one customer can be used as the basis for making recommendations to another, similar one. Though this approach is closed-loop and adaptive, it has severe shortcomings. Collaborative Filtering is prone to 'obvious' errors when data is scant or noisy. It functions poorly for the most sophisticated repeat customers whose preferences are may be atypical. Because it depends on customer data, collaborative filtering cannot classify or respond properly until sufficient precedent data has been accumulated. The result is a classic "start-up problem" – it is difficult to launch a new site or new products based using this approach. Yet, it is precisely at start-up that recommendation quality is most critical to establish customer loyalty.

Various refinements have been proposed to compensate for the data dependence of collaborative filtering. For example, Autonomy [1] builds a profile of user interests based on individual user's regular actions and employs pattern-matching technology to make recommendation. Others [10] apply data mining to user behaviors to extract usage knowledge. [3] implemented a hybrid system Fab, where they combined the content-based approach, using the techniques of information retrieval (IR), with collaborative filtering approach, trying to incorporate the advantage of both and inherit the disadvantages of neither. The practical shortcoming of these hybrids is that they are incapable of discovering the deep levels of domain knowledge that guide the behavior of human service agents and underlie customer response.

In the design of our system, we determined that structured deep knowledge of products, and the preferences of the website owner and customer must be taken into

account in personalization. Unfortunately, no existing approach effectively combines declarative knowledge management with adaptive response.

2.2 Solutions of the Sales Agent

Our sales agent takes a knowledge-based approach that integrates customer behavior data. It tries to resolve the shallow knowledge limitation of adaptive techniques by enabling companies to easily model their domain knowledge and use that knowledge on-line. It sidesteps the data-dependency issue of collaborative filtering by modeling the customer within the framework of a pre-defined product ontology:

- The agent uses ontology engineering technology [6] to model the product types, relationships and sales expertise. It makes recommendations based on its deep knowledge about relationships among products.
- It provides website operators with a method to describe their own requirements, and integrate individual customer's preferences with the site's when making recommendations.
- It uses machine learning to infer customer preferences from his/her behaviors (clicks, purchases, etc.) and select appropriate products based on the inferred preferences.
- It uses machine learning to infer customer preferences from his/her behaviors (clicks, purchases, etc.) and select appropriate products based on the inferred preferences.

By employing this hybrid of machine learning and ontology engineering, the sales agent will empower websites to cultivate knowledge about customers and products to create highly competitive, intelligent services.

3 System Infrastructure Design and Prototype Implementation

Our objective is the development of an adaptive, ontology-based sales agent for Internet personalization. Specifically we will devise generalized method for acquiring, representing, manipulating, and visualizing product knowledge, customer knowledge and sales knowledge. The agent therefore requires the following ontology-related tools (Fig. 1).

3.1 Product Modeling Tool

The Product Modeling Tool builds and manages models of products based on the product ontology (Fig. 2). The product information includes product names, relations and constraints. This tool has the following features:

- Allows users to navigate visualize and edit a product knowledge base consisting of product classes, relations, constraints and instances.
- Maps the product knowledge base to a conventional, external product database.
- Checks for consistency of the product knowledge base.

A similar idea is found in webKB [2], which also constructs the knowledge base using a predefined ontology. WebKB tries to use machine learning to automate the construction process and extract new website instances, while our sales agent emphasizes the role the user in mapping between product knowledge and the product database.

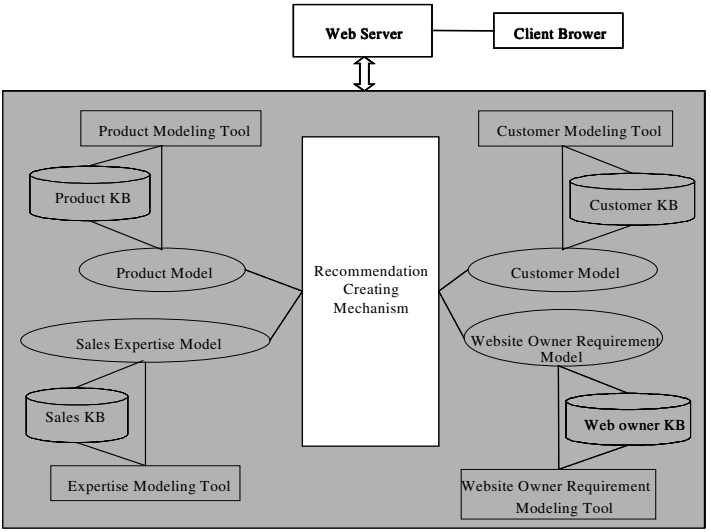


Fig. 1. Infrastructure of the Sales Agent

3.2 Expertise Modeling Tool

The Expertise Modeling Tool supports modeling of sales agent behavior, including expert knowledge such as how salespeople target recommendations for particular customers. It has the following features:

- It enables an expert salesperson familiar with the products to specify preferential relationships among products according to his sales experience, based on the product ontology defined in the Product Modeling Tool (see Fig. 2);
- After collecting website usage information, it applies data mining to determine the probability of useful cross-selling relationships. For example, it might determine that customers who buy a tent have a 60% possibility of buying a flashlight.

– It learns from the effectiveness of its recommendations and dynamically modifies its recommendation strategy accordingly.

Our sales agent allows expert salesperson to apply manually specified product relationships in making recommendations and to adapt and refine the recommendations using machine learning or data mining -- methods extensively studied by many researchers [4, 5, 7].

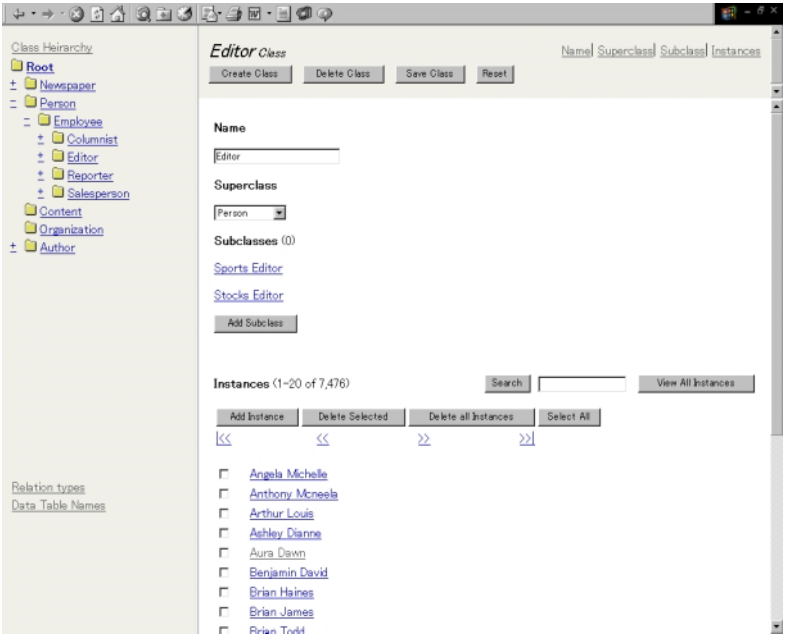


Fig. 2. Product Knowledge Modeling

3.3 Customer Modeling Tool

The Customer Modeling Tool records customers' behaviors (event stream) and derives their preferences and habits. It learns from the customer's behavior and makes inferences about the customer's interests using a probabilistic network. It can also be integrated with collaborative filtering to augment customer profiles. Because the profiling is dynamic, it can readily respond to changes in customer interest. Customer profiles typically consist of:

– registration information such as cust_id,
– behavior history. For example, action(behavior, time, item, …),
– interests (with probability) to products. For example, p(cust_id, prod_id) = 0.5.

3.4 Website Owner Requirement Modeling Tool

When making recommendations, the sales agent should take into account not only the customer's, but also the website's preferences. The Website Owner Requirement Modeling Tool is built for this purpose. It allows the website owner to specify how certain products should be recommended to certain types of customers (see Fig.3).

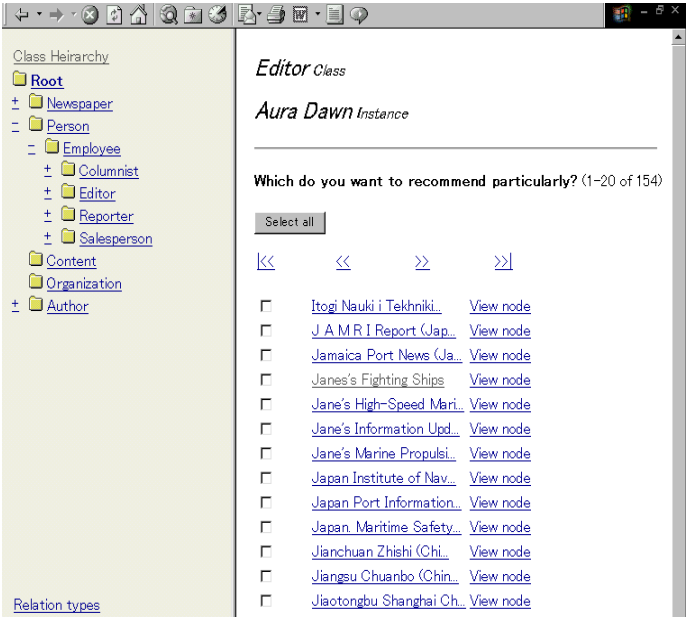


Fig. 3. Website Owner Requirement Modeling Tool

3.5 Recommendation Creating Mechanism

The Recommendation Creating Mechanism creates personalized content based on the models of products, customers, sales expertise and web owner's preferences. This involves conventional matching, sorting, and dynamic HTML techniques.

4 Conclusion and Further Consideration

This paper presented our ongoing project—an ontology-based sales agent, its infrastructure, and main features. We have already implemented a prototype system. Combining AI technologies such as ontology engineering and machine learning, the proto-

type addresses critical problems of current approaches to personalization. Our agent will enable websites to acquire and apply customer knowledge, product knowledge, sales expertise and website owner requirements, and offer intelligent, personalized service.

Consumer privacy is one of several issues that merit further investigation. Next generation sales agents must balance the need for service based on customer knowledge with the consumer's desire for anonymity.

We are currently engaged in performance evaluations of the system and developing a method of benchmarking recommendation performance. We hope to make statistical results of these evaluations available soon.

References

1. Autonomy. URL: http://www.autonomy.com/tech/wp.html
2. Craven, M.: Learning to Construct Knowledge Bases from the World Wide Web. Artificial Intelligence, 1-2 (2000) 69–113
3. Balabanovic, M., Shoham, Y.: Content-based, Collaborative Recommendation. Comm. of the ACM, 3 (1997) 66-72
4. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in knowledge discovery and data mining. AAAI Press/MIT Press, CA. (1996) 1–31
5. Michalski, R., Kubat, M., Bratko, I., Bratko, A.: Machine Learning and Data Mining: Methods and Applications. John Wiley & Sons Ltd. (1998)
6. Mizoguchi, R.: A Step Towards Ontological Engineering. URL: http://www.ei.sanken.osaka-u.ac.jp/english/step-onteng.html
7. Mitchell, T.: Machine Learning and Data Mining. Comm. of the ACM, 11 (1999) 30–36
8. Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of the ACM Conference on Computer Supported Cooperative Work (1994) 175–186
9. Resnick, P., and Varian, H.: Recommendation Systems. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in knowledge discovery and data mining. AAAI Press/MIT Press, CA. (1996) 1–31
10. Mobasher, B., Automatic Personalization Based on Web Usage Mining. Technical Report TR99-010, Department of Computer Science, Depaul University (1999)
11. Schafer, J. B., Konstan, J., Riedl, J.: Recommender Systems in E-Commerce. In: Proceedings of the ACM Conference on Electronic Commerce (1999) 158–166

Anomaly Detection of Computer Usage Using Artificial Intelligence Techniques*

Jongho Choy and Sung-Bae Cho

Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
Tel: +82-2-2123-2720, Fax: +82-2-365-2579
{hosoft,sbcho}@candy.yonsei.ac.kr

Abstract. Intrusion detection systems (IDS) aim to detect attacks against computer systems by monitoring the behavior of users, networks, or computer systems. Attacks against computer systems are still largely successful despite the plenty of intrusion prevention techniques available. This paper presents an IDS based on anomaly detection using several AI techniques. Anomaly detection models normal behaviors and attempts to detect intrusions by noting significant deviations from normal behavior. Raw audit data are preprocessed and reduced into appropriate size and format using Self-Organizing Map (SOM). Different aspects of a sequence of events are modeled by several hidden Markov models (HMMs), and a voting technique combines the models to determine whether current behavior is normal or not. Several experiments are conducted to explore the optimal data reduction and modeling method. For the optimal measures, system call and file access related measures are found useful and overall performance depends on the map size for each measure. Voting technique leads to more reliable detection rate.

1 Introduction

Nowadays, virtually every serious medium to large sized web site adopts IDS to protect it from various attacks. Ever uprising interests on and needs for IDS have led to much research in this field. On the market, there are several products based on diverse intrusion detection techniques. Intrusion detection is to find attacks exploiting illegal uses or misuses. Basically, an IDS analyzes information about users' behaviors from various sources such as audit trail, system table, and network usage data. Current intrusion detection techniques can be divided into two groups according to the type of information they use: misuse detection and anomaly detection. Misuse detection uses knowledge about attacks, whereas anomaly detection uses normal behaviors. Misuse detection attempts to model attacks on a system as specific patterns, and systematically scans the system for occurrences of the patterns [3][7][12]. It has the advantage that known attacks

* This work was supported in part by a grant from the Korea Information Security Agency.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 31–43, 2001.

can be detected reliably with a low false-positive error rate where normal behaviors are mis-recognized as attacks owing to its knowledge about attacks. It is also economical because it just requires scanning known attack patterns. The shortcoming is that it cannot detect novel attacks against systems and collecting attack information is difficult.

On the other hand, anomaly detection addresses the problem of detecting novel attacks against systems. It attempts to detect intrusions by noting significant deviations from normal behavior [1][8][9]. One clear drawback is that non-intrusive behavior that happens to fall outside the normal range will be labeled as an intrusion, suffering a high false-positive error rate. The other difficulty is to analyze large amount of data to model normal behaviors. In spite of the potential drawbacks, anomaly detection has more potential as the intrusion detection technique because it has the ability to detect unknown and novel attacks against computer systems.

There are several issues to address to develop an anomaly detection based IDS. What to model? How to model? How to detect? How to improve the detection rate and reliability of the system? are among them.

First, we should decide the target activities to monitor. A target activity, which we'll call an event, can be a low-level system call or a higher level command issued by users. In addition to the granularity of an event, we should decide which aspects of each event to consider. Because there is usually a large amount of raw audit data to analyze, we also need a tool to effectively reduce the data and extract information from it. In this paper we use Sun Microsystem's Basic Security Module (BSM) [11] to collect the audit data. We extract measures from BSM audit data and transform them into one dimensional data using SOM that is a neural network capable of clustering input patterns into fixed vectors automatically. The reduced data can be used to model normal behaviors for intrusion detection. To select the optimal measures of a BSM event, we evaluate the effect of the measures by investigating the performance in IDS.

Second, modeling plan and a modeling tool are needed. To determine current behavior is an anomaly, a normal behavior model must be built against which to compare. A normal behavior can be modeled in several ways. We test 3 modeling plans: single model without user discrimination, separate models for individual users, and models for user groups. HMM is used as a modeling tool.

Third, we should be able to determine whether current behavior is normal or not. We use HMM as the primitive anomaly detection tool at each measure of an event. Later, we consider multiple measures at the same time and make final decision using voting method to improve the performance and reliability of the system.

The rest of this paper is organized as follows. In Section 2, we give a brief overview of related works and background. The overall design, preprocessing and intrusion detection is given in Section 3. Experimental results are shown in Section 4.

2 Backgrounds

The first phase of intrusion detection is to collect audit data from user and system activities. IDS extracts several measures from audit data using BSM, and reduces them in order to profile user's normal behaviors. After normal behavior modeling, IDS can determine whether current behavior is normal or not. Typically measures that are related to the system bugs exploited by invaders for a long time or important system resources are selected.

BSM collects data whenever an event occurs, makes a record and appends it to the tail of audit data. Therefore audit data consists of a series of audit records as time goes on. However, because these data are too huge to be processed in real time, we extract useful information from BSM and ignore the others. There is no exact answer to the question which measures are useful for detecting intrusion. Although much information is used to detect intrusion, it cannot guarantee a high detection rate because important information can be hidden.

Extracting effective measure is very important because it determines the performance of an IDS. Measures widely used in IDS are CPU time, I/O access time, file access, system call, network activity, login/logout, etc. To select the optimal measures in IDS, we have extracted several statistics from the BSM audit data based on the well-known attacks and investigate the effect of each measure by testing the performance of it. Table 1 summarizes the measures used in this paper.

Table 1. Measures extracted from the BSM to detect intrusion

Group	Measures
System call	ID, return value, return status
Process	ID, IPC ID, IPC permission, exit value, exit status
File access	access mode, path, file system, file name, argument length

Drawback of misuse detection techniques that they can detect known attacks only has stimulated much research on anomaly detection techniques to cope with novel attacks. The temporal sequence of event under consideration is evaluated how far it deviates from the normal model with various techniques. In [6], after normal behavior model is built based on user activities, event sequence vector similarity is evaluated using pattern-matching technique. [2] and [13] model each program's usage pattern. There are several works that evaluate collections of modeling and matching techniques. [2] compares pattern matching, BP neural network and Elman neural network. [13] compares frequency, data mining and HMM. Results in [2] and [13] show exploiting temporal sequence information of event leads to better performance. Temporal sequence information is considered as a significant factor in [6].

For multiple measure combination, various techniques can be used including voting, neural network and expert system. Linguistic interpretation is also possible using fuzzy reasoning [4].

3 Intrusion Detection System

3.1 Overview

The intrusion detection system developed in this paper is composed of preprocessing module which is in charge of data filtering and data reduction, and anomaly detection module which is responsible for the normal behavior modeling and anomaly detection, as shown in Fig. 1. Normal behaviors are modeled into profile database through model learning.

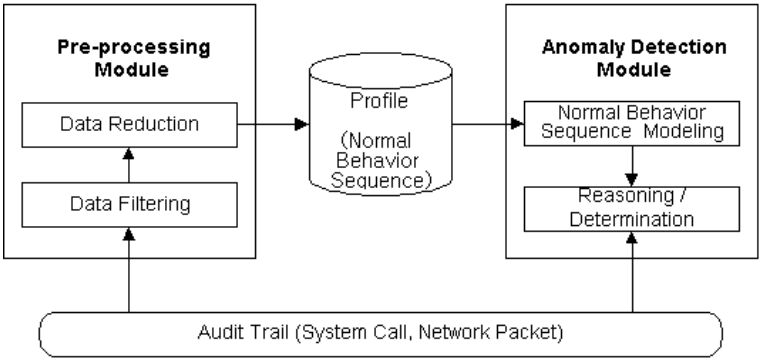


Fig. 1. Overview of intrusion detection system

3.2 Preprocessing

Problem of audit data is on huge amount of information that cannot be dealt with in real time. If all events are audited, audit data may be collected as the amount of hundreds of mega bytes per day. Therefore they should be reduced into the small size to be processed in real time. Reduced data have many measures, and the data can be useful in statistical analysis but they need to be transformed into low dimensional data for various modeling techniques such as neural networks and HMM.

Here, reducing size of measure means that $x_1, x_2, ..., x_n$ are converted into $x'_1, x'_2, ..., x'_k$ where k is much smaller than n. Fig. 2 shows the flow of reducing audit data. In this paper, we reduce the measure size based on the locality of user action. We observe the range of the measures and save actual ones used in the system as table where we find the value of measure of current action. As a

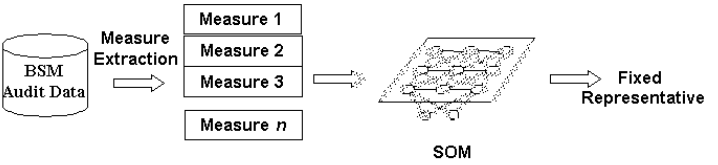


Fig. 2. Overall flow of reducing BSM audit data

result of mapping of measures, we can obtain reduced data. Several measures can be useful in the statistical analysis, but if these are converted into one representative, they can be easily used to model normal behaviors by modeling techniques. We propose a novel method based on SOM that self-organizes the map automatically according to the input pattern without answer, and that converts multi-dimensional data to low dimensional one. SOM is an unsupervised learning neural network, using Euclidean distance to compute distance between input vector and reference vector [5]. Algorithm of SOM is as follows. Here, $i(x)$ is the best matching neuron to input vector, λ_i means neighborhood function and η is learning rate.

1. Initialize the reference vectors $(w_j(n))$ $\quad (n = 0)$
2. Compute distance and select the minimum value $i(x) = \arg\min \parallel x(n) - w_j(n) \parallel$
3. Update the reference vectors $w_j(n+1) = w_j(n) + \eta(n)\lambda_{i(x)}(n, j)(x(n) - w_j(n))$
4. Repeat from 2 to 3 until the stop condition satisfies

Several measures can be extracted from one audit record of BSM that includes an event and the information is normalized for the input of SOM. As an output of SOM, we can get one representative instead of many measures; that is, one record is converted into one representative. The reduced information can be used in various modeling techniques such as neural network and HMM.

3.3 Behavior Modeling and Intrusion Detection Using HMM

An HMM is a doubly stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic processes that produce the sequence of observed symbols [10]. This model can be thought of a graph with N nodes called 'state' and edges representing transitions between those states. Each state node contains initial state distribution and observation probability at which a given symbol is to be observed. An edge maintains a transition probability with which a state transition from one state to another state will be made. Fig. 3 shows a left-to-right HMM with 3 states.

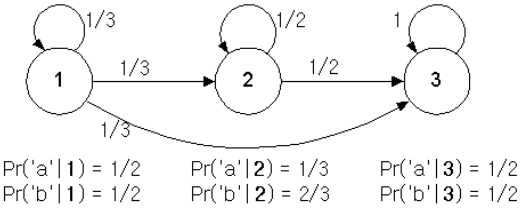


Fig. 3. An example of left-to-right HMM

Given an input sequence $O = O_1, O_2, ..., O_T$, HMM can model this sequence with its own probability parameters using Markov process though state transition process cannot be seen outside. Once a model is built, the probability with which a given sequence is generated from the model can be evaluated. A model λ is described as $\lambda = (A, B, \pi)$ using its characteristic parameters. The parameters used in HMM are as follows:

$T =$ length of the observation sequence
$N =$ number of states in the model
$M =$ number of observation symbols
$Q = \{q_1, q_2, ..., q_N\}$, states
$V = \{v_1, v_2, ..., v_M\}$, discrete set of possible symbol observations
$A = \{a_{ij} | a_{ij} = \Pr(q_j \text{ at } t+1 | q_i \text{ at } t)\}$,
state transition probability distribution
$B = \{b_j(k) | b_j(k) = \Pr(v_k \text{ at } t | q_j \text{ at } t)\}$,
observation symbol probability distribution
$\pi = \{\pi_i | \pi_i = \Pr(q_i \text{ at } t = 1)\}$, initial state distribution

Suppose, for example, a sequence *aba* is observed in a model λ in Fig. 3 and initial state is 1, then the probability with which the given sequence is generated via state sequence 1-2-3 is calculated as follows:

$$\begin{aligned}
&\Pr(O = aba, q_1 = 1, q_2 = 2, q_3 = 3 | \lambda) \\
&\quad = \pi_1 \cdot b_1(a) \cdot a_{12} \cdot b_2(b) \cdot a_{23} \cdot b_3(a) \\
&\quad = 1 \cdot 1/2 \cdot 1/3 \cdot 1/2 \cdot 1/2 \cdot 1/2 \\
&\quad = 1/48
\end{aligned}$$

The probability with which the sequence is generated from a model can be calculated by summing the probabilities for all the possible state transitions. In practice, more efficient method, known as forward-backward procedure, is used. Normal behavior modeling and anomaly recognition is shown in Fig. 4.

Anomaly Recognition. Anomaly recognition matches current behavior against the normal behavior models and calculates the probability with which

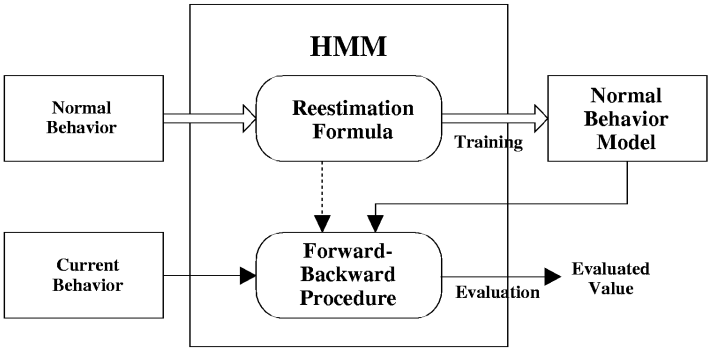


Fig. 4. Anomaly recognition and normal behavior modeling using HMM

it is generated out of each model. Forward-backward procedure or Viterbi algorithm can be used for this purpose [10]. Each probability is passed to the determination module to be determined whether it is normal or not.

Normal Behavior Modeling. During normal behavior modeling, HMM parameters are determined based on the normal behavior sequences generated in preprocessing step. The determination of HMM parameters is to adjust $\lambda = (A, B, \pi)$ to maximize the probability $\Pr(O|\lambda)$. Because no analytic solution is known to do it, an iterative method called Baum-Welch reestimation is used [10].

3.4 Multiple Models Combination Using Voting Technique

When one event is evaluated through each model, a vector of evaluation values is generated. A method to combine multiple models is required to finally decide whether current sequence is an anomaly. In this paper, each model determines whether current sequence is abnormal or not according to the measure it is responsible for. Each model participates in final anomaly decision. Each model is given a weight W_m according to their confidence. Typical voting methods include unanimity, majority and OR voting. In OR voting, anomaly is determined if at least one member votes positively. Voting is to determine whether or not the total result R is greater than or equal to the T dependent on voting method.

$$R = \sum W_m * V_m \quad \begin{pmatrix} W_m & : & \text{model weight} \\ V_m & : & \text{model voting value} \end{pmatrix}$$

$$\text{anomalous if } \begin{cases} R = 1 & \text{(unanimity)}, \\ R \geq 0.5 & \text{(majority)}, \\ R > 0 & \text{(OR voting)} \end{cases}$$

4 Experimental Results

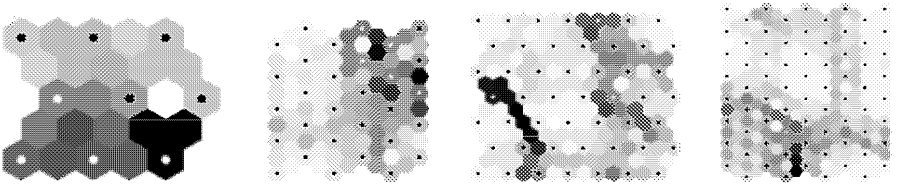
We have used data obtained from three graduate students for one week. They have mainly used text editor, compiler and programs of their own writing. Total 60,000 data have been collected and among them 30,000 are used for training and another 30,000 are for test. 17 cases of user-to-root (u2r) intrusion, one of the most typical intrusions, are included in each user's test data set. An unprivileged user acquires super user privilege by exploiting a subtle temporary file creation and race condition bug. When combining multiple models, we used just one student's data.

In anomaly detection, it is natural that false-positive error is higher than misuse detection. If false-positive error is high, the IDS may not be practical because users may be warned from the intrusion detection system, though their action is not illegal. Therefore, we should regard false-positive error as critical error. Actually, we investigate Receiver Operating Characteristics (ROC) curve [2] in the experiment that presents the variation according to the change of false-positive error.

4.1 Pre-processing

It is generally known that large map size of SOM can classify better than small one but it is not certain when the locality of input pattern is high. In order to find the optimal map size of SOM, we have considered quantization error of SOM and map usage. We have also observed the distance between reference vectors to investigate the distribution of input pattern map with change of map size. After that, using HMM, we have tried to find the optimal measures to improve the performance of IDS by investigating the change of ROC curve for each measure.

Effect of Map Size. Applying SOM to real problems requires to be considered the map size of SOM. There is no answer to which map size of SOM is good for intrusion detection. In order to solve this problem, we consider two factors: quantization error and map usage. The map size of low quantization error and high map usage in SOM can be regarded as good.



(a) 3 × 3 SOM (b) 5 × 6 SOM (c) 7 × 7 SOM (d) 8 × 9 SOM

Fig. 5. Variation of distance in reference vectors with map size

Fig. 5 shows the variation of reference vectors with changing map size of SOM. In the figure, dark area means that one reference vector is distant from the other. As the map size gets larger the distance among reference vectors is closer, which also indicates that the locality cannot disappear though the map size of SOM is large. Also, as shown in Fig. 6, as the map size is larger, quantization error is smaller, but the probability of usage is lower. As can be seen in Fig. 6, the gradient of error curve is lax when the map size is larger than 25. We can also see that probability of usage is above 50 % when the map size is smaller than 50. Therefore, useful map size should be selected between 25 and 50.

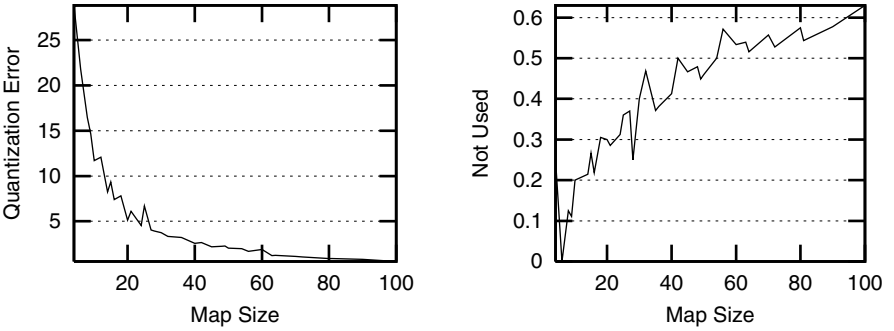
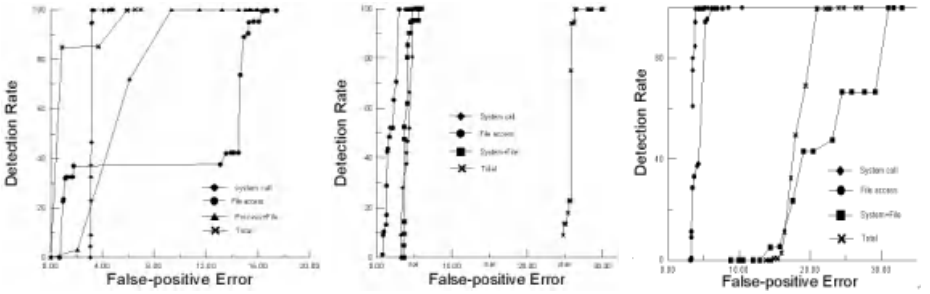


Fig. 6. Variation of quantization error and map usage according to the map size

Useful Measures. In this paper, we extract 3 measures from the data that are related to system call ID, return value, return status), process (ID, IPC ID, IPC permission, exit value, exit status) and file access (access mode, path, argument length, file system). We have also taken into account the combinations of 3 measures. First of all, we reduce their size based on the locality, convert them into a representative, and make a sequence so that it can be used in modeling normal behaviors by HMM. Several map sizes such as 5×5, 6×6 and 7×7 are considered for the experiment. Using HMM, we have 10 states and make the number of symbols varied with the map size of SOM. Fig. 7 shows the ROC curves according to the map size of SOM. In case of map size of 25 (5×5), such measures as system call and combination of process and file access including all measures are useful and file access can detect intrusion well in the map size of 36 (6×6). As can be seen in 7×7 map, a larger map does not guarantee a performance improvement. On the contrary, the performance decreases compared with the smaller map size.

4.2 Modeling and Intrusion Detection

Effectiveness of HMM based Intrusion System. At first, single model is used to model all the users. Optimal HMM parameters are obtained by experiments: a length of 30 and number of states of 10. Fig. 8 shows the change in



(a) 5×5 (b) 6×6 (c) 7×7

Fig. 7. Measure effect of IDS when map size changes

sequence evaluation values over time when an intrusion occurs. Attack was occurred between time 11 and 32. The sequence evaluation values between time 11 and time 32 decrease radically due to abnormal behavior, which implies the system effectively detects the intrusion.

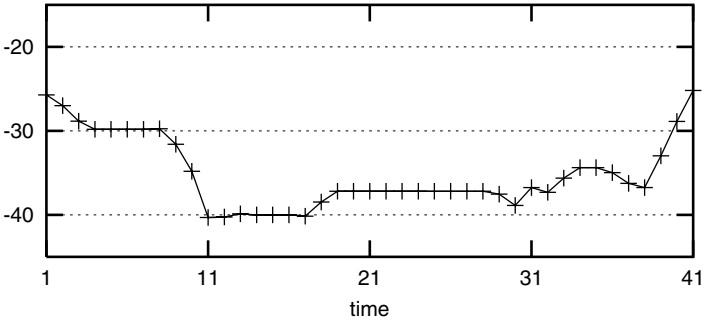


Fig. 8. Change in sequence evaluation value when an intrusion occurs.

Performance Variation According to Modeling Plan. This experiment is conducted with separate model for each user. Table 2 shows the intrusion detection rate and false-positive error rate for each model. Separate modeling has shown lower overall false-positive error rates than single modeling.

We have got low evaluation values with other users' model in evaluating user behaviors as shown in Table 3. This shows the proposed technique is effective in modeling user's normal behaviors and can also be used in detecting intrusions from sniffed user accounts.

Finally, a group of users who have shown similar behaviors are modeled together in the same model. This decreases the modeling costs that will increase

Table 2. The anomaly detection rate for separate user modeling

	single modeling	separate user modeling		
		user1	user2	user3
mean (S.D)	-14.74 (10.42)	-15.54 (5.88)	-4.72 (8.42)	-14.25 (12.34)
threshold	-32.45	-26.21	-31.35	-36.92
detection rate	100%	100%	100%	100%
false-positive error rate	2.95%	4.31%	1.73%	1.96%

Table 3. The number of sequences whose evaluation value is less than -50 when different user's model is used

		evaluated user		
		user1	user2	user3
referenced user model	user1	0	366	55
	user2	566	315	336
	user3	475	346	0

as the number of users increases. We consider the fact that user 1 and user 3 have worked on similar tasks and the difference between the behaviors of the two are relatively small when we grouped user 1 and user 3 together. User grouping reduces user1's error rate, whereas that of user3 increases, as shown in Table 4 and Table 5. This seems to be because population under test is not so large as to guarantee the similarity with group and the relative similarity of the two users are not so high as expected.

Performance Enhancement with Multiple Model Combination. The model based on system call measure and that on measure reduced by SOM are combined using voting. Unanimity, majority and OR voting methods are tested. Each measure is given the same voting weight.

We have used the best result from each model because threshold for each model may differ from each other. Final detection rate does not change because each model's detection rate is 100 %. False-positive error rate has enhanced compared to its primitive models as shown in Table 6.

5 Concluding Remarks

In this paper, we have proposed an intrusion detection system that reduces raw audit data using SOM, models user's normal behavior using HMM and

Table 4. The anomaly detection rate for group user modeling

	Separate modeling			Group modeling (user1, 3)
	user1	user2	user3	
mean (S.D)	-15.54 (5.88)	-4.72 (8.42)	-14.25 (12.34)	-16.20 (8.70)
threshold	-26.21	-31.35	-36.92	-27.83
detection rate	100%	100%	100%	100%
false-positive error rate	4.31%	1.73%	1.96%	4.09%

Table 5. The anomaly detection rate for each modeling method

	single modeling	separate modeling	group modeling
detection rate	100%	100%	100%
false-positive error rate	2.95%	2.67%	3.30%

Table 6. The performance of multiple model combination

	system call	SOM reduced	voting method		
			unanimity	majority	OR voting
detection rate	100%	100%	100%	100%	100%
false-positive error rate	5.33%	23.53%	1.18%	25.75%	25.75%

determines anomaly by analyzing events that are generated in sequences. This system effectively detected an intrusion when one occurred. False-positive errors were raised in sequences where the normal behaviors were not learned during the training phase. It can be concluded that provided with the complete normal behaviors, HMM can be used as an effective intrusion detection technique.

Comparison of modeling methods shows that individual user modeling has fewer errors than single modeling. However, modeling by user grouping has not shown improved performance. With sufficiently large number of users, it might show better performance and modeling capability. Multiple model combination has achieved an improvement in detection rate and reliability of the system.

References

1. S. Forrest, S.A. Hofmeyr and A. Somayaji, "Computer immunology," *CACM*, vol. 40, no. 10, pp. 88–96, October 1997
2. A.K. Ghosh, A. Schwartzbard and M. Schatz, "Learning program behavior profiles for intrusion detection," *Proc. Workshop on Intrusion Detection and Network Monitoring*, pp. 51–62, Santa Clara, USA, April 1999
3. K. Ilgun, R.A. Kemmerer, and P.A. Porras, "State transition analysis: A rule-based intrusion detection system," *IEEE Trans. on Software Engineering*, vol. 21, no. 3, March 1995
4. J.S. R. Jang, "Fuzzy Inference System," *Neuro-Fuzzy and Soft Computing*, Prentice-Hall, NJ, 1997
5. T. Kohonen, *Self-Organizing Maps*, Springer Press, 1995
6. T. Lane and C.E. Brodley, "Temporal sequence learning and data reduction for anomaly detection," *Proc. ACCS '98*, pp. 150–158, 1997
7. W. Lee, S. Stolfo, and P. Chan, "Learning patterns from Unix process execution traces for intrusion detection," *Proc. AAAI97 Workshop on AI Methods in Fraud and Risk Management*, 1997
8. T.F. Lunt, "A survey of intrusion detection techniques," *Computer & Security*, vol. 12, no. 4, pp. 405–418, June 1993
9. P.A. Porras and P.G. Neumann, "Emerald: Event monitoring enabling responses to anomalous live disturbances," *Proc. 20th NISSC*, pp. 353–365, October 1997
10. L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989
11. Sunsoft, *Solaris 2.5 Sunshield Basic Security Module Guide*, 1995
12. G. Vigna and R.A. Kemmerer, "Netstat: A network-based intrusion detection approach," *Proc. NISSC'98*, pp. 338–347, October 1998
13. C. Warrender, S. Forrest and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," *Proc. IEEE Symposium on Security and Privacy*, pp. 133–145, May 1999

Application of Self-Organizing Maps to Classification and Browsing of FAQ E-mails

Hyun-Don Kim and Sung-Bae Cho

Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
neoace@candy.yonsei.ac.kr, sbcho@csai.yonsei.ac.kr

Abstract. Widespread use of computer and Internet leads to an abundant supply of information, so that many services for facilitating fluent utilization of the information have appeared. However, many computer users are not so familiar with such services that they need assistant systems to use the services easily. In the case of Internet portal services, users' e-mail questions are answered by operator, but increasing number of users brings plenty of burdens. In this paper, we propose a two-level self-organizing map (SOM) that automatically responds to the users' questions on Internet, and helps them to find their answer for themselves by browsing the map hierarchically. The method consists of keyword clustering SOM that reduces a variable length question to a normalized vector, and document classification SOM that classifies the question into an answer class. The final map is also used for browsing. Experiments with real world data from Hanmail net, the biggest portal service in Korea, show the usefulness of the proposed method. Also, the browsing based on the map is conceptual and efficient.

1 Introduction

Proliferation of computer and increasing interest of Internet lead many people into information-network based services. However, it is difficult for people who are not familiar with such services to get what they want in those services. They are used to suffer from several difficulties, such as installation of program, usage of specific function, and so on. It is time-consuming and difficult work to overcome such problems even for expert users as well as beginners. Information-network service corporations, ISP (Internet Service Provider) and IP (Information provider) companies, open a line which helps users by telephone, provide answers about questions by web board systems like FAQ (Frequently Asked Questions), and respond to the questions by e-mail. However, the explosive increasing number of users makes service corporations to recognize the needs of automatic response system.

At the time of writing this paper, more than 5 million people use the Hanmail net that is the biggest portal service in Korea and users' questions per day come to about 200 cases. It is redundant and time-consuming to respond to duplicated questions by hand, and even worse user may not satisfy with the response time. Automatic processing of users' questions might be not only efficient for operators who can avoid redundant task but also satisfactory for users. This paper proposes a two-level self-

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 44–55, 2001.

organizing map (SOM) that automatically classifies the users' questions for Hanmail net, and provides conceptual browsing facility. We show that the automatic response system based on the model is plausible for e-mail data of which number and size per class are various.

In Section 2, we introduce related work and explain the problem with data that are real world query mails obtained from Hanmail net. Section 3 illustrates the architecture of automatic classification and browsing system. Section 4 shows the experimental results to show the usefulness of the proposed method.

2 Backgrounds

2.1 Related Works

Generally speaking, automated text categorization (TC) is a supervised learning task, defined as assigning category labels (pre-defined) to new documents based on the likelihood suggested by a training set of labelled documents [1]. The automatic response of the query mails in the Internet is also a sort of the automated categorization of texts into topical categories.

TC has a long history, dating back at least early 60s, Until the late '80s, the most effective approach to the problem seemed to be that of manually building automatic classifiers by means of knowledge engineering techniques. Machine learning paradigm has emerged in the '90s [2]. Within recent paradigm, learning phase builds a classifier that learns the characteristics of one or more categories from the pre-classified training data set, and the classifier is used in categorizing the unseen data. The advantages of this approach are on accuracy comparable to human performance and considerable saving in terms of expert manpower, since no intervention from either knowledge engineers or domain experts is needed [2].

We use the idea of TC to classify the questions of Internet. However, our task must be not a supervised learning but an unsupervised learning, because our data is not uniformly distributed on the classes. To solve this problem SOM is used for machine learning

2.2 User Questions of Hanmail Net

To classify the query mails received, we collect and analyze users' questions at first. The number of query mails collected is 2232, and the number of classes is 69. Table 1 gives account on the data groups, and Table 2 shows the distribution of users' questions that have been received during one month. A half of the entire data comes from group A, and group B contains the queries that must be forwarded to operator without classification. Group C represents less frequent queries, so it is difficult to classify those classes using a conventional statistical method.

Table 1. Characteristics of each group of users' questions

Group	Characteristics
A	Frequency is so high that occupies a half of data
B	The query must be forwarded to operator for manual treatment
C	Frequency is low
D	Remaining queries

Table 2. Distribution of data in each group

Group	# of class	# of query
A	6	1002 (44.9%)
B	7	585 (26.2%)
C	36	127 (5.7%)
D	20	518 (23.2%)
Total	69	2232 (100.0%)

Table 2 shows that queries are concentrated on group A and some classes in group D cannot be easily characterized, so it is much difficult to extract features which help classification. Also, because E-mails are generated by novice, there are many expressions that are colloquial, abbreviated, and solecistic. Such characteristics of data bring difficulty to classify the queries.

3 Classification and Browsing

The system consists of two parts: classification and browsing subsystems. The classification system also consists of two parts. The first part is preprocessing and keyword clustering which help to encode the input vector for the next classification module, by reducing the dimension of the input mail. The dimension of vector is reduced by a SOM for keyword clustering. The second part is classifying the queries and matching them with the corresponding answers by another SOM called mail classification SOM.

The browsing system is based on the completely learned mail classification SOM. It helps users to search their answer conceptually by navigating the system hierarchically with topology-preserving property of SOM. A user can find answer by changing the resolution of the information of the map. Fig. 1 shows the overall system architecture.

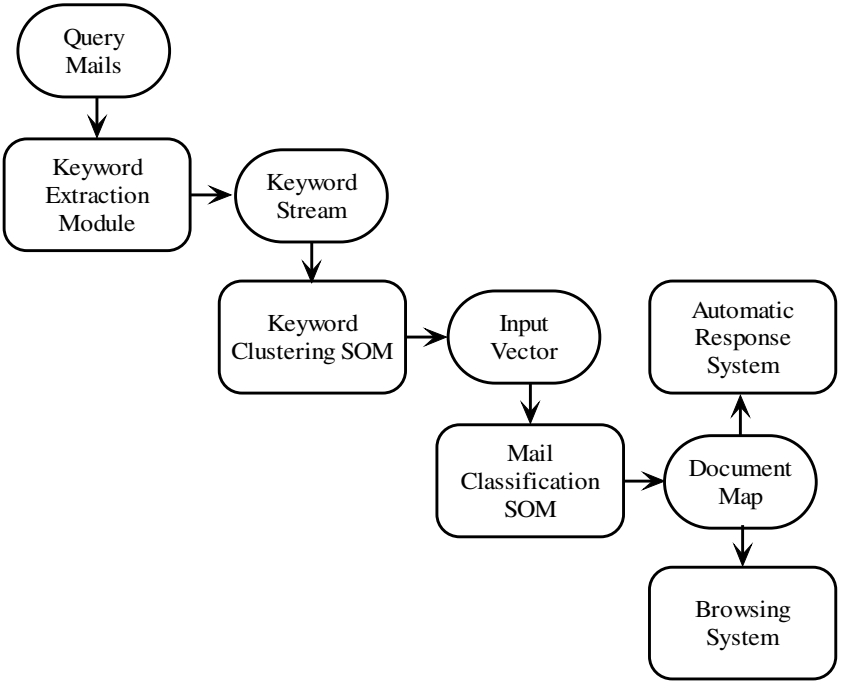


Fig. 1. Flow of system architecture

3.1 Keyword Extraction

Query mails are written by natural language, especially Korean or compound of Korean and English. The query mails must be encoded as normalized vector form to get the input vector for neural network. To vectorize the query mails keyword should be extracted at first, because queries include meaningless components such as auxiliary word, the ending of word, and article. Since the components do not affect the meaning of the mail, they should be removed. Fig. 2 shows an example of query mail and the corresponding keywords extracted. The whole mail is reduced to only ten keywords in this example mail.

3.2 Keyword Clustering

After the keyword extraction process as shown in Fig. 2, the keywords should be converted to vectors of real numbers. There are several methods that analyze and vectorize the information, like vector space model [1]. The computational and spatial reason leads to choose the SOM-based approach as the basic method. This is asked on Scholtes to encode documents based on their words, like a neural filter and a neural interest map for information retrieval [2][3].

Fig. 2. An example of keyword extraction

Keyword clustering SOM is used in reducing and clustering the category of words for query mails. The input of keyword clustering SOM is a keyword identification vector with context information of a sentence that includes the keyword. Because SOM collects and clusters similar data at the neighborhood, similar keywords are mapped in adjacent nodes in SOM [4][5]. Keyword clustering SOM plays the similar role of the thesaurus that discriminates the synonyms in the conventional text processing.

Learning algorithm of the SOM is as follows [4][5]:

$$m_i(t+1) = m_i(t) + \alpha(t)\, n_{ci}(t)\, \{x(t) - m_i(t)\} \tag{1}$$

Here, $\alpha(t)$ is a function of learning rate, $n_{ci}(t)$ is a function of neighborhood, $m_i(t)$ is the weight of the ith node, and $x(t)$ is input data of SOM. At the function of neighborhood c means the index of winner node, and winner is obtained by the following equation [4][5]:

$$\|x - m_c\| = \min_i \{\|x - m_i\|\} \tag{2}$$

All keywords in a query mail are represented by a vector, and each word is represented by an n-dimensional real vector x_i. Input vector of keyword clustering SOM is encoded as follows [6]:

$$X(i) = \begin{bmatrix} E\{x_{i-1} \mid x_i\} \\ \alpha x_i \\ E\{x_{i+1} \mid x_i\} \end{bmatrix} \tag{3}$$

Here, E denotes the estimate of the expectation value evaluated over the group, and α is a small scalar number (e.g., 0.2) [6][7][8][9][10]. The input vector in keyword index i, $X(i)$, consists of one symbol vector x_i and two context vectors of x_i-1 and x_i+1. x_i-1 means the keyword occurring just prior to keyword x_i, and x_i+1 means the keyword occurring just posterior to keyword x_i. As mentioned above, all occurring

prior keyword vectors are summed, and then averaged by number of occurrence of *xi*. It is the same for the posterior keyword. The two vectors give the context information of the sentence to the input vector $X(i)$. SOM is trained with these two vectors and it can cluster the similar keywords.

We assume that number of all keyword is 8, as shown in Fig 2. So each keyword can be expressed by 1 bit of total 8 bits stream, like below.

Sub-scribe	017	mo-bile	e-mail	Noti-fica-tion	ser-vice	re-ceived	full	Mes-sage	Cell-pho-ne	arri-val	Ans-wer

Because keyword e-mail and arrival are expressed before keyword notification for two lines, the expectation value of priori keyword for keyword notification is like below.

0	0	0	1/2	0	0	0	0	0	0	1/2	0

Because keyword service is expressed after keyword notification for two lines, the expectation value of posterior keyword for keyword notification is like below.

0	0	0	0	0	1/2	0	0	0	0		0

Finally, the encoded vector looks like that:

Expectation value of priori keyword	0	0	0	0	□	0	0	0	0	0	0	0	Expectation value of posterior keyword

3.3 Mail Classification

The keyword clustering SOM maps a set of similar keywords into a node of the SOM, because all keywords appeared in a query have a winner node in SOM. According to each winner node, a query mail can be converted into a histogram for keywords that make up a query mail. For example, if the size of keyword clustering SOM is 3 by 3 and a query mail consists of three keywords: one mapped on node (1,2) and others mapped on node (0,0). Then the encoded vectors are shown as follows:

2	0	0	0	0	1	0	0	0

The dimension of input vectors is the same with the output number of keyword clustering SOM. Therefore, the size of the map decides the reduction degree and dimension of the input vector of the mail classification SOM. For example, the SOM which has *m***n* size map produces *mn* dimension input vector. With this method, input vector can be produced, but there is a shortcoming. Produced vectors are very sensitive to the specific query mail, i.e., vectors overfit to query data. Because queries are written by various people and have various wording, vectors which reflect a specific case are not desirable. To overcome the variation problem, a smoothing process that has frequently used in pattern recognition area is applied. Such process blurs the sensitive integer vectors to non-specific real vectors. Gaussian kernel is used for blurring the histogram as follows [11]:

0.25	0.5	0.25
0.5	1	0.5
0.25	0.5	0.25

In addition, Shannon's entropy [1][6] that improves the separability of keywords is utilized. Entropy converts the frequency, which is increased just by 1, to weighted frequency that can be increased by real values. Keywords that reflect the characteristic of specific classes have higher degree of frequency, and others have lower degree of frequency. The entropy weight calculated as follows:

$$\begin{aligned} &E_w = H_{\max} - H(w), \\ &H_{\max} = \log(number\_of\_classes) \\ &H(w) = -\sum_i \frac{n_i(w)}{\sum_j n_j(w)} \ln \frac{n_i(w)}{\sum_j n_j(w)} \end{aligned} \tag{4}$$

Here, $H(w)$ means the entropy value of keyword w, $n_i(w)$ means the frequency of occurrence of word w in class i, and $H_{\max}$ is the maximum entropy, that is log 67 in our case. V_i the ith component of blurring histogram and imposing entropy to it is calculated as follow:

$$V_i = \sum_w (F_w \times \frac{G_i}{E_w}) \tag{5}$$

Here, G_i means the ith component of the smoothing kernel, E_w means the entropy weight of keyword w, and F_w means the frequency of keyword w. after vector blurring

and entropy weighting processes are finished, inherent vector of one query mail is produced.

The trained mail classification map with the vectors above, maps each query data into a specific node. According to the characteristics of SOM, the query mails that belong to the same class are mapped in the same or neighborhood positions. The new query mail sent by a user is encoded by the above process. The input vector that is mapped on one node, is classified as a certain class and then automatic response system sends the corresponding answer which matches with the winner node to the user promptly.

3.4 Browsing

The browsing system that visualizes the completely trained mail classification SOM helps users to find their answer interactively. This has a hierarchical structure that consists of three levels, and each level contains several keyword labels. Each label represents a node of the map [12].

The label must reflect the content of queries that are mapped in a node, because the meaningful label can help user to search the answer correctly. The basic measure produces the label is the frequency of each keyword. But the importance and class-separability of keywords are different from each other; some keywords have high frequency but they have trivial meaning. Similarly, some keywords have low frequency but they appear only at specific classes. The frequency is modified to reflect such phenomena as follows :

$$L_{wi} = F_i \cdot \frac{Doc_{Fin}}{Doc_{Fout}} \tag{6}$$

Here, L_{wi} means the importance of the keyword w in a node i of the map, F_i means the frequency of w in a node, Doc_{Fin} means the number of documents which have the keywords w in a node, and Doc_{Fout} means the number of documents which have the keywords w in all the nodes except node i.

To provide the conceptual browsing, the whole map is displayed at the first level, and the user grasps the overall distribution of questions. If the user selects a node by clicking the position that probably has similar questions, it displays the second level map that is a part of the whole map around the selected node. At this level, user can get more specific information. If the user selects a node again, the third level map is displayed. At the third level, the user can obtain the answer for his question in a node by clicking the nearest keyword.

4 Experimental Results

Experiments are conducted for 67 classes, 2206 query mails. Since the size of keyword clustering SOM is fixed to 10 by 10, the dimension of the input vector of the mail classification SOM is 100. The first experiment decides the size of mail

classification SOM, where entire 2206 data are used for training the map. The result is shown in Table 3.

Table 3. Recognition rate for training data

Size of keyword clustering SOM	Recognition rate	
100 x100	1553/2206	70.40%
120 x 120	2014/2206	93.74%
150 x 150	2098/2206	95.01%
160 x 160	2075/2206	94.06%

The analysis of recognition rate with the keyword clustering SOM of 150□150 is summarized in Table 4. The recognition rates of group A and group D are not good. In case of group A, four classes which do with the e-mail account of Hanmail net are not classified well. Four classes include account change, deletion, duplication, and information confirmation. In these classes, the extracted keywords are so similar that the keywords cannot reflect the attribute of query mail. That is the reason why the recognition rate is so low. Group D includes 16 classes that have trivial information about the remaining 51 classes. It deteriorates the classification rate because the keywords are so similar with the remaining classes.

Table 4. Recognition rate for each group of data

Group	# of class	Recognition rate	
A	6	954/1002	94.0%
B	7	561/585	95.9%
C	36	124/127	97.6%
D	18	459/492	93.3%
Total	67	2098/2206	95.01%

Next, we have fixed the size of keyword clustering SOM as 150 □ 150, and divide the data set into two groups. One is training data of 1545 query mails, and the other is test data of 661 query mails. Experiments are performed with the change of threshold, which is quantization error. Threshold value also affects the rejection rate. The result is shown in Table 5.

Fig. 3 through Fig. 5 show an example of hierarchical browsing sequentially.

Table 5. Recognition rate for test data

Threshold	Rejection rate	Recognition rate
0.5	46.4%	58.5%
0.4	60.1%	64.0%
0.3	72.9%	82.7%

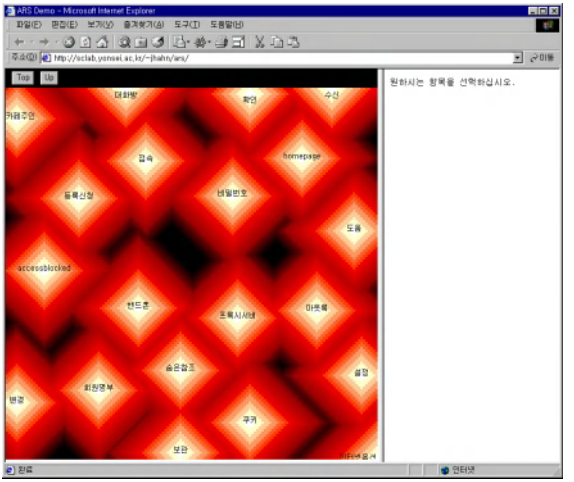


Fig. 3. The first level of browsing in our example: The left frame displays the map, and the right frame displays the window for answer.

5 Concluding Remarks

In this paper, we have proposed an automatic response system of query mails of a portal service corporation in Korea, and shown that the system is promising by conducting several experiments. As shown in experimental result the accuracy rate of training data is high, and the accuracy of test data is plausible but the rejection rate is a little high. We are under way to increase the accuracy while maintaining the low rejection rate. Moreover, we are also attempting to enhance the browsing interface more appropriately to the novice users.

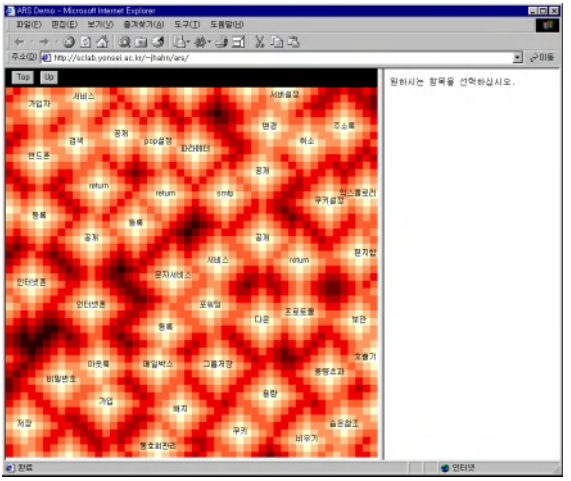


Fig. 4. The second level of browsing in our example.

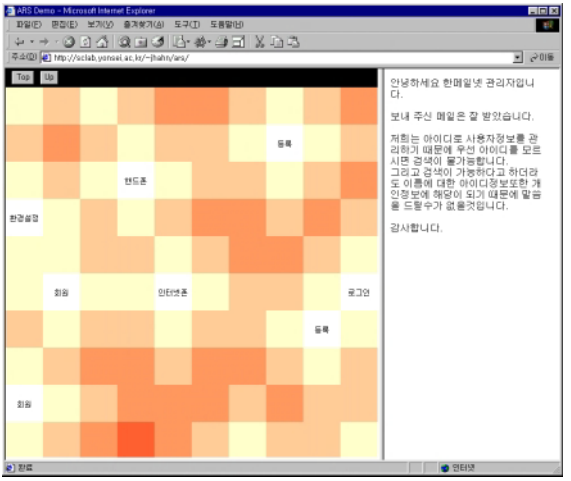


Fig. 5. The third level of browsing in our example: The right frame shows the answer.

References

1. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addisom-Wesley (1988)
2. Scholtes, J. C.: Kohonen feature maps in fulltext database--A case study of the 1987 Pravda. In Proc. Informatiewetenschap, STINFON, Nijmegen, Netherlands (1991) 203-220

3. Scholtes, J. C.: Unsupervised learning and the information retrieval problem. In Proc., Intl. Joint Conf. on Neural Networks, IEEE Service Center, Piscataway, NJ (1991) 95-100
4. Kohonen, T.: Self –organized formation of topologically correct feature maps. Biol. Cyb. (1982) 43: 59-69
5. Kohonen,T.: Self-organizing Maps, Springer, Berlin Heidelberg (1995)
6. Ritter, H. and Kohonen, T.: Self-organizing semantic maps, Biol. Cyb. (1989) 61: 241-254
7. Kaski, S., Honkela, T., Lagus, K. and Kohonen, T.: Creating an order in digital libraries with self-organizing maps. Proc. World Congress on Neural Networks (1996) 814-817
8. Honkela, T., Kaski, S. Lagus, K. and Kohonen, T.: Exploration of full-text databases with self-organizing maps. In Proc., Int. Conf. on Neural Networks, IEEE Service Center, Piscataway, NJ, (1996) 1: 56-61
9. Lagus, K., Honkela, T., Kaski, S. and Kohonen, T.: WEBSOM for textual data mining, Artificial Intelligence Review. (1999) 13: 345-364
10. Kaski, S. S., Honkela, T., Lagus, K. and Kohonen, T.: WEBSOM--self-organizing maps of document collections. Neurocomputing (1998) 21: 101-117
11. Gose, E. and Johnsonbaugh, R. and Jost, S.: Pattern Recognition and Image Analysis, Prentice Hall PTR (1996)
12. Lagus, K. and Kaski, S.: Keyword selection method for characterizing text document maps. In Proc. Intl. Conf. on Artificial Neural Networks, Vol. 1. IEEE, London (1999) 371-376

Model Based Program Specification and Program Generation – In a Case of Vehicle-Engine Control System Design

Setsuo Ohsuga [1], Shunsuke Chigusa [2], and Katsuyuki Sugaya [3]

[1]Department of Information and Computer Science
Waseda University
4-1 Ohkubo, Sinjuku-ku, Tokyo, 169-8555, Japan
ohsuga@ohsuga.info.waseda.ac.jp
[2] Toyota Motor Corporation
1200 Mishuku Susono Shizuoka, 410-1118, Japan
chigusa@fantasia.ml.toyota.co.jp
[3]Department of Information and Computer Science,
Waseda University
3-4-1 Ohkubo, Sinjuku-ku, Tokyo, 169-8555, Japan
sugaya@ohsuga.info.waseda.ac.jp

Abstract. A new style of program generation is discussed. A key concept is model and its building process. Programming is a kind of problem solving. What users want in programming are represented in the form of model in a comprehensive way for users. A computer system interprets the model and generates programs automatically as automatic problem solving. This idea is being applied to make programs for the embedded system in vehicle-engine to control its operations. The objective is to enable engine designers as non-specialists of programming to develop programs themselves.

1 Introduction

Recently in many engineering fields, products as the objects of design are becoming more and more complex. This tendency is accelerated by introduction of computers embedded in various engineering products such as vendor machines, telephone terminals, and copy machines. These computers enable engineers to design these products to adapt to changing environment. This is the largest one out of many reasons to make these design objects grow complex. Until the near past the control mechanisms have been realized in the form of electronic hardware devices or sometimes by mechanical elements and their capabilities as control units were limited. It has prevented the operations of products to be too complex. Today computer control broke through the limit and very fine control becomes possible. It induced the growth of the complexity of the products themselves. For example, vehicle-engine changed rapidly in a short period with considerable increase of engine power per a cylinder. It became possible because computer control enabled designer to design rather complex mechanism, for example from two valves to four valves per cylinder, and nevertheless to control very finely the operation of these complex products.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 56–71, 2001.

It brought however a difficulty in designing these objects in many fields because of heterogeneous natures between technologies of programming and those in the fields which the computers are embedded. In fact, designing computer control program in a vehicle engine for example is very different from the traditional method of designing main body of the engine. The vehicle engine designers design most parts of the engine but these designers are not always the professionals of programming and accordingly they have to ask the professional group the programming tasks. They have to make programming specifications for the purpose. Very often it takes unexpectedly large cost and time. But the still worse is that very often the program cannot be made optimal because not only engine designer is the non-specialist of programming and accordingly program specification cannot be made optimal but also programmers are ignorant of the details of vehicle engine. It is because, today, specification must be made considering the program to be generated in mind. As the result, it is worried that the program specification cannot follow the progress of the other part of vehicle engine technology.

One of the reasons to bring this difficulty is that the programming as a part of vehicle engine design is separated from the other part and must be left to those who are of the different specialty from vehicle engine. It is difficult to expect those programmers to become the specialists of engine design because in many cases programmers have many jobs covering wide area of problem fields and therefore cannot stay to a specific field. This is due to the difference of the natures of specialties of machine design and programming. As one of the possibilities for resolving this difficulty therefore, it is discussed in this paper a way of making specification without taking the programming in mind and generating program automatically from this specification. AI technology is used. But AI has not been able to deal with complex problems involving various tasks [ROS97]. A new method is introduced in this paper.

Though the method is discussed in this paper using a control system embedded in vehicle-engine as an example, it is not limited to this case but can be extended to all engineering products that involve computers as parts. As these computers are expected to work so that these engineering products show the highest performance, the requirement to program design must be included as a part in the requirement for whole product design.

The authors believe that this technology becomes more and more important because the computer's role in designing the engineering products is becoming larger.

2 Requirement to Programming

What is required for automatic programming is that the generated program meets the following two conditions.

(1) Semantic and logical legalities; Every program must assure the semantic and logical legalities from the engineering point of view. That is, every program has to generate legal output to the legal input for controlling the object.

(2) Physical condition; Every program must satisfy the requirement for the physical conditions such as processing time, memory space, etc. In particular real time requirement is crucial. That is, most control programs are required to work along with the real time operations of the total objects and to achieve their goals in a limited time. Hence the processing time requirement is the major one.

3 Current Style of Program Development for Controlling Vehicle Engine Operation

The development procedure of an engine control system is not so much different compared with the common control system. The feature of engine control is the following two points.

(1) Processing timing is decided severely.

(2) Much fail-safe is incorporated.

For example, while engine rotates two times, "ignition timing control" must read the value of each sensor, and must determine the following timing. Since this control is greatly concerned with the performance of engine, it wants to process as correctly as possible. An engine designer determines timing etc. using various knowledge and know-how. "Control specifications" has indicated such specification. "Control demand specifications" is passed to a software designer and the "Implement specifications" for making a program is created. " Implement specifications" is described in consideration of the specification of the micro-computer to use etc. And a program is created on the basis of " Implement specifications." However, it is rare that they are made newly. In many cases, the specifications of the similar engine developed in the past are used. That is, a past example is used as a prototype. It is easier there to be for a prototype and to correct it. Moreover, two or more control systems can be intermingled now to one micro-computer by improvement in semiconductor technology in recent years. Consequently, the scale of a program also became large and its degree of complexity has also increased. Therefore, a control program is developed independently for every function, and is collected at the end. However, a control program comrade's mismatching, such as a collision of a priority, may arise as a problem. In this case the rearrangement of a program developed independently are needed. In order to solve such problems, use of Real-time OS is also examined[SAT97].

4 Model Based Specification

A program is a description of computer’s activity in a programming language. The reason why programming is not easy for human users is not only because the users must use programming language but also because the users must think of and represent the activity in the computer’s way. If users can describe the activity in the

way convenient for the users, in other words, if the engine designer can make the requirement for the control program in terms of engine technology, not in the term of computer technology, then programming specification becomes familiar and easier for engine designer. Since the human way and the computer's way of describing activity are different, the former must be converted to the latter. There are following two problems.

(1) What is the convenient way for users to make specification,

(2) How do computers translate the user's specifications to computer's programs.

In this paper a model-based method of specification and knowledge-based method of translation are adopted for (1) and (2) respectively.

4.1 Modeling Scheme

A program is a formal representation of specific activities of a computer. This representation must be created in human brain. A program specification is an intermediate form in the process of translating human idea to a formal program. In order to make the specification more comprehensive for users, it is necessary to represent this translation process explicitly. Ohsuga proposed a modeling scheme that represents human idea as correctly as possible while remaining sufficient formality to allow computers to deal with problems represented in this scheme [OHS96,OHS99a,OHS99b]. It is composed of three different parts; user model, subject model and object model.

In order to represent problems correctly, person's view must be included in the problem representation. Everything in the world can be an object of human interest. That is, a person has interests in everything in the world. Even the other person's activity can be an object of a person's interest. Thus it forms a nest structure of subject's interests. Any problem can be represented formally by a model in this scheme. For example, programming is a subject's activity to represent the other subject' activity in interest. Automatic programming is still the other (third) subject's activity to represent the second subject's programming.

Program is a special type of automatic problem solver and thus three subjects at the different levels concern programming. Subject S1 is responsible to execute a task, Subject S2 makes program for the task of S1 and Subject S3 automates S2's programming task. The activities of these subjects can be represented, processTrans (S1, Task), makeProgram(S2, S1, Task, Program) and automateActivity (S3, S2, System), respectively. This observation leads ones to a Multi-Strata Structure of Objects and Activities to represent related concepts in consideration and to a Multi-Strata Model as a model of Multi-Strata Activities (Figure 1). In reality this is a problem model.

A multi-strata model is composed from three different sub-models; Pure Object Model, User Subject Model and Non-User Subject Model. These are in the following relations (Figure 2).

Problem Model = Subject Model + Pure Object Model

Subject Model = User Subject Model + Non-User Subject Model

This is a general scheme and the detail is abbreviated here. Only a part related to the objective of this paper is discussed in the sequel. Refer to [OHS98, OHS00]

To make a subject model is to represent the relation of various activities that have been defined and represented. An activity is said defined when (1) it is represented explicitly by a specific form of predicate and (2) either it is the programmed activity or it is provided with an expansion rule of the form,

activity (Subject, Var-1, Var-2, --, Var-M) :-

activity1(Subject 1, Var-i1, --, Var-ir),

– –

activity n (Subject n, Var-k1, --, Var-ks).

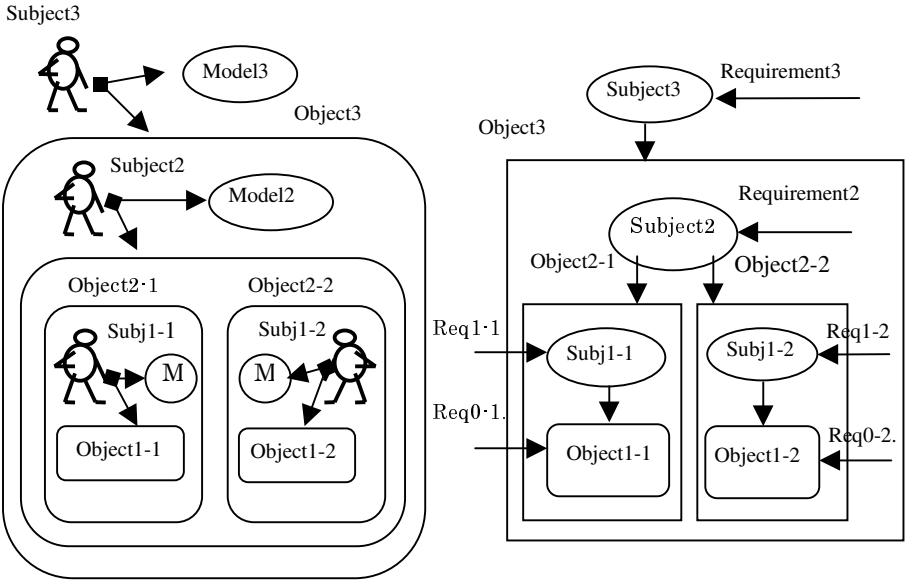


Fig. 1. Multi-strata object and multi-strata model

Here 'activity i' in the right is a smaller activity that has either the similar expansion rule or programmed activity. The programmed activity is an activity to which a computer-program is provided that produces the same output as the required activity to the same input. It executes a specific function that can share a role to achieve a final goal. For example, in case of vehicle-engine control, an activity to decide an ignition time depending on the volume of air-intake and number of rotation

is defined as a programmed activity. These activities have been defined and accumulated in this field.

Subject and Subject i, (i = 1,2,--,n), represents the subjects of the respective activities. These are necessary as a general scheme. These subjects make a structured subject model and activities are related to each other indirectly through the subject model. But for the specific case of this paper, every subject is the same computer and is ignored. Then activities are no more structured by the subjects but are related only through relation rules as the expansion rule as above directly. Var-1, Var-2, --, Var-M, Var-i1, --, Var-ir, Var-k1, --, Var-ks are variables.

4.2 Relation between Activities

There are two different relations among activities that can be represented in this form. One is the relation between a macro activity and its constituent activities. Another is the semantic relation to show the connectivity of two or more activities.

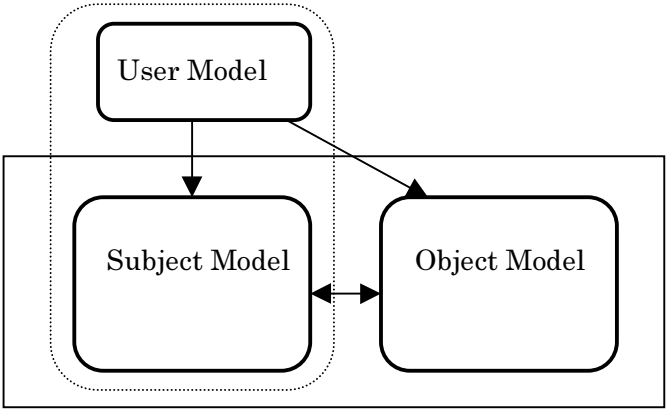


Fig. 2. Constituents of multi-strata model

4.2.1 Macro Activity

A macro activity is defined as a composite of mutually related activities. It is given a name and dealt with as an independent activity. The macro activities thus defined may define a still larger macro. Thus a hierarchy of macro activities is formed (Figure 3). The lowest level activities in this hierarchy are programmed activities. In many cases these programmed activities are incomprehensive for the user who is not a specialist of computers. On the contrary the higher the level of an activity in this hierarchy, it is more comprehensive for the field engineers because the macro activities are defined

to represent those activities that are necessary to represent the behavior of the object in which the computer is to be embedded. The expression mentioned as above is used to represent a macro activity. In this case, the rule may be represented as follows.

activity (Subject, Var-1, Var-2, --, Var-M) :-

smallerActivity1(Subject 1, Var-i1, --, Var-ir),

--

smallerActivity n (Subject n, Var-k1, --, Var-ks).

where 'activity' is a macro and it is defined as composed from a number of small activities. The variables are classified into three classes; input variables, output variables and internal variables. In order for simplifying the expression of variables in the rule, a macro activity is defined as to include only input variables and output variables without internal variables. Let it be represented as 'activity(Subject, Input, Output)'. That is, $\cup_i$ Var-i = 'Input' $\cup$ 'Output'. Every 'smallerActivity' in the right hand of the rule includes also input variables and output variables. Output of some activity can be the input for the other activity in the right hand of this rule. Therefore some variables in the set $\cup_i(\cup_j$ Var-ij) are intermediate variables that are closed in the right hand in this rule. In the ordinary logical expression, these internal variables must also be included in the macro activity. Instead of abbreviating them a special expression of formula to let them closed in the right hand like,

activity(Subject, Input, Output):-
($\exists$Internal/Domain)[smallerActivity1(Subject 1, Input 1, --, Output1),
--
smallerActivity n (Subject n, Input n, --, Output n)].

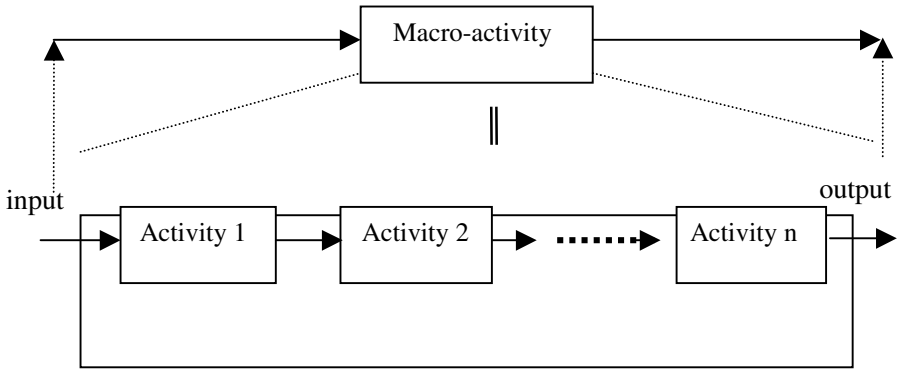


Fig. 3. Macro-activity

That is, every internal variable is quantified in the right hand of the rule.

In the application to the vehicle-engine control-program, what is required is an activity to generate information (output) to the actuators given the information (input) from the sensor(s). It is represented by a macro activity and two types of conditions are imposed to the activity. One is that the macro activity must be composed of programmed activities keeping semantic legality of input-output relations and the other is that processing time is within the required limit. To develop a program is to find such a macro activity meeting the conditions. This method is possible when complete knowledge is provided to the system in order to represent necessary set of macro operations to represent every requirement. This is not always the case. Very often necessary knowledge is lacked. For example, if a new activity is defined, it cannot be used in this framework until a new macro including the activity is generated.

4.2.2 Semantically Legal Connection between Activities

Among a number of activities that are defined independently, only some groups of activities can be connected preserving semantic legality. This is called semantically legal connection among activities. The output(s) of some activity(ies) becomes the input of an activity in this group (Figure 4). It is represented as follows.

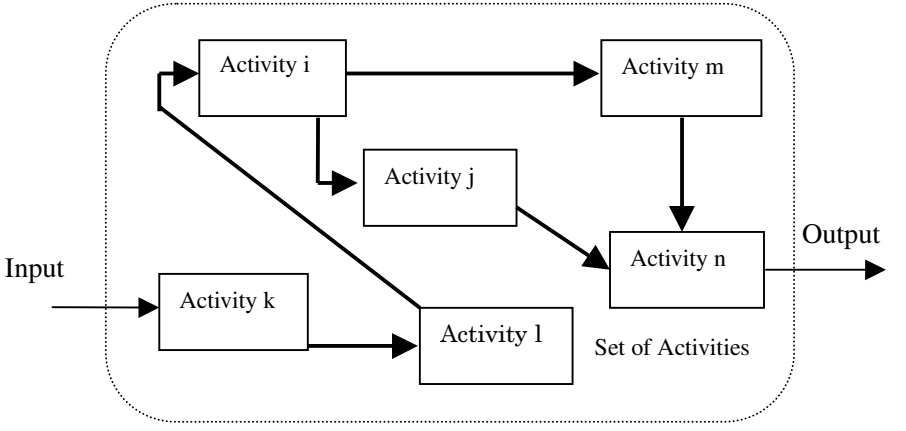


Fig. 4. Semantically legal connection

activity(Y1, Y2, --, Yn, Z) :-

(∃X1, X2, --, Xn/D1,D2,--Dn)[activity 1(X1, Y1), activity 2(X2, Y2),

--, activity n(Xn, Yn)], defineActivity (Y1, Y2, --, Yn, Z).

Here the variables Y1, Y2, --, Yn, and Z are the inputs to and the output from the 'activity' respectively. The variables X1, X2, --, Xn are the inputs to the activities that are in the semantically legal connection with the 'activity'. The semantically legal connection must be made by person and is represented in the form of knowledge. A set of these connections generates various compounds of activities. These compounds are generated by deductive operation. For example, assume that, given a specific value to Z, a question 'activity(Y1, Y2, --, Yn, c)?' is asked. The inference system retrieve the above rule, replaces the query by the right hand of which Z is substituted by c, obtains the instance values b1, b2, --, bn to the variables Y1, Y2, --, Yn to make 'defineActivity (Y1, Y2, --, Yn, c)' true, substitutes each of b1, b2, --, bn into Y1, Y2, --, Yn of 'activity1', 'activity 2',--, 'activity n' respectively, and generates a set of new queries, 'activity 1(X1, b1)?', 'activity 2(X2, b2)'?, -- and 'activity n(Xn, bn)?'. This is repeated until the query is answered true. The first query, 'activity(Y1, Y2, --, Yn, c)?' in the above explanation, may be a definition of an actuator and the last query(ies) may be a (set of) sensor(s). Then a sequence of activities included in this deduction form a compound of activities. This is a pre-programming stage of programming and a program is generated from the result. Among many compounds of activities the one that satisfies the second requirement given to the program to be generated, say processing time and/or memory space, is selected.

The activity included semantically legal connection can be any macro activity. Thus the macro-activity definition and semantically legal connection are combined in various ways.

Since semantically legal connection is given locally in the small group of activities, it can happen to form a loop together with the other connections. It must be detected at the time a connection is given and the loop is replaced by a macro. Within this macro a loop parameter is introduced such that the macro operation can be translated to a loop in the program.

4.3 Model-Based Specification of Program

What is required for program specification is to give information with which computer system can generate a compound of activities satisfying the requirements. It is to provide a set of activities and their relations. In many cases in designing a vehicle-engine, a set of activities that have been developed before are referred. These engines developed so far are classified by various indices such a engine type, number of cylinders, total volume, compression ratio, horse power, maximum torque, etc. When a new engine with the specified characteristics is to be developed, some engines that have the similar characteristics as the new one are referred. Those activities included in these engines developed in the past form an initial set of activities. The compounds of activities have also been made for every past engine and are shown on the display. It is possible to re-use the same compounds as before to the new engine. But in the other case, it is not possible because of the change of the requirement. Then the new compounds have to be generated. It is intended in this research project to take the following procedure.

(1) To fix the set of activities;

The user selects the activities from the initial set of activities and merges them. In some, but not so many, cases a new activity is developed and added to the set. Creating the new activity is the matter of research.

(2) To define new semantically legal connection;

The semantically legal connections that have already defined before are shown on the display. The user defines new legal connections for the new groups of activities in the set of activities.

(3) To define new macros;

The user defines compounds and makes new macros directly. He/she verifies the logical validity of the new macros partially.

The result is the model of problem to develop programs for controlling the vehicle-engine.

5 Knowledge Based Generation of Program

Programming is divided into two stages. The first is to make compounds of activities such that required condition between specified inputs and outputs are satisfied. The second is to translate the compounds to programs.

5.1 Generating a Compound of Activities as Problem Solving

It is not difficult to obtain a compound of activities for a specific instance of input and output. It is to solve a problem of route finding based on the semantically legal connection of activities. It starts finding the route from the activity for actuator, 'outputActivity'. Assume that there is the following rule.

outputActivity(Y1, Y2, --, Yn, Output) :-

(∃X1, X2, --, Xn/Dx1,Dx2,--Dxn)[activity 1(X1, Y1), activity 2(X2, Y2),

--, activity n(Xn, Yn)], defineActivity (Y1, Y2, --, Yn, Output).

Then a specific value c is selected for the output variable Output and generate a query '(∃Y1, Y2, --, Yn/Dy1,Dy2,--Dyn)outputActivity(Y1, Y2, --, Yn, c)?'. Then the inference system evaluates defineActivity (Y1, Y2, --, Yn, c) and obtains the values b1, b2, --, bn for the Y1, Y2, --, Yn that make defineActivity (b1, b2, --, bn, c) true. Then it generates a set of new query,

(∃X1/Dx1)activity 1(X1, b1)?

(∃X2/Dx2)activity 2(X2, b1)?

(∃Xn/Dxn)activity n(Xn, b1)?

Each of these expressions is an abbreviated form. If an 'activity i' needs multiple inputs as 'activity', Xi/Dxi is not a single variable but must be a vector. This operation is repeated until an 'inputActivity' representing a sensor operation is reached. Since the semantically legal connection is defined locally between individual activities, number of paths can result. Some of them goes from the input to the output. The system looks for such paths by back tracking. Among number of possibilities, the one to satisfy the other requirement (especially the time constraint) is finally selected.

If a macro activity has already been defined, then it is represented as follows.

outputActivity(Y, Output):- inputActivity(Y), macroActivity(Y, Output).

To the given query, the right hand expression becomes the new query in which Output is substituted by c. Then the 'macroActivity' is expanded by the macro rule. The effect of making macro activities is that the possible path can be found without searching.

5.2 Generalizing a Solution Tree

Since the program is generated from the compound activity the activities in this selected path from the input to output is reserved. This is not a simple sequence but a structure of activities representing a solution to the problem of generating a compound of activities. But this is the solution for the specific output. Before translating it into program, it is generalized in order to recover the instance to the original variable. This is to expand the solution tree so that it represents the solution for any value of output in the intended region.

This is discussed precisely in [OHS98, KAW00] and is not repeated here. The final solution tree is composed of programmed activities at the leaf. The structure represents a control structure of the target program.

5.3 Simulation for Estimating Processing Time

At the time a solution tree is first made, it is not yet a real but a tentative solution. The program generated from the tree must satisfy the other requirements. Among those requirements given to the program, that of processing time is the major one because in most cases a computer system embedded into an object system works as a real-time controller.

Usually the embedded compute system is required to do various tasks. A program is necessary for each task and those programs have to do the tasks in a strictly limited time as a whole. For example, in case of vehicle engine control system, more than twenty programs are executed in an MPU. Most of these programs are evoked according to the rotation of the engine. Some others are evoked by an internal clock mechanism and still the others are evoked occasionally by driver's actions. Some of these programs are evoked in parallel. To every program is given a priority of processing and a real-time multi-processing OS is used to control the operations. When some programs are activated in parallel, these are put into a queue line and are processed according to the order of the priorities. Those programs that are of the

lower priorities have to wait the end of processing of high priority programs. It may take a long time and can go over the limit time. It is necessary to analyze the system operation. If the system is assured to satisfy the requirement, then the solution tree for each program is fixed and the program is generated from the tree.

This is a queuing problem and its simulation system has been developed. This is a general queuing process simulator. In order to achieve the simulation, it is necessary to estimate the processing time of each program. As has been discussed, a program is generated in two steps; to generate a compound of activities and to translate it into programming language. It is desirable that the simulation can be executed at the end of the first step because a solution tree may have to be changed by the result of the simulation.

After a solution tree for a program is generated, the processing time is estimated from the tree. The processing time of any simple programmed activity can be estimated relatively precisely. But the processing time of the whole compound is not deterministic but probabilistic because the tree includes disjunctive nodes representing branch conditions in the corresponding program. The processing time of the compound is thus obtained as a random variable depending on the probability of branch at every branch point. It is not easy to estimate correctly the probability of branch. Therefore an approximated distribution of processing time is made.

The simulator is composed of three subsystems; a ‘call’ generator, a ‘call’ processor and a time management sub-system. The ‘call’ generator has three components; engine rotation, real time clock and random number generator. It generates ‘calls’ under the given operation condition of vehicle engine such as climbing a steep with the given speed. Each ‘call’ corresponds to a program execution. The future ‘calls’ are listed on a ‘call’ generation table with the remaining time to the ‘call’ can really occur. This table is the ‘call-generation queue’. The remaining time is generated by a call-generation routine. In fact the relative remaining time is used. The relative remaining time is the time needed for generating the ‘call’ after the prior ‘call’ is generated in the queue.

Every program is listed on a property table with such characteristics as program execution time and priority. The ‘call’ processing subsystem manages the queue and accumulates the simulation data. The ‘calls’ in a queue are listed on a waiting table with the relative remaining time. This table is called the ‘call-processing table’.

The time management sub-system advances the simulation time to the nearest future event. The event is either the generation of new ‘call’ or the end of processing of the ‘call’ at the top of the queue, that is, the end of the program. The time management sub-system advances the time by the amount the smaller one either of the heads of the ‘call-processing queue’ and the ‘call-generation queue’. This amount is reduced from the remaining time of ‘calls’ at the head of both queues. The ‘call’ with the remaining time reduced to 0 is erased.

In some cases, a program cannot be processed within the required time limit. Then some trouble occurs. This effect is different by the program. Some one may have fatal effect on the performance of the total engine operation but the others have the smaller effects. The effects of the individual cases must be analyzed in advance. The total

effects have to be estimated by the simulation. The priority of the program must be decided based on the effects. If the processing times of some programs are considered too large, the system goes back to generate the other compounds of activities.

If the required time constraint cannot be satisfied in whatever the way, then the capability of the MPU is not enough. A new, more powerful MPU must be introduced.

5.4 Generation of Program

Once the solution tree is fixed, it is translated to a program code. The structure represents a control structure of the program to be generated. For example, disjunctive nodes are included in the tree, each of which corresponds to either a branch or a loop depending on the lower part of the structure. Simply speaking, if the same structure appears in the lower part of this node, then this part of the structure including the disjunctive node is translated to a loop. Otherwise the node represents a branch. The activities connected to a conjunctive node represent a sequential operation. A way to translate the solution tree into a program is discussed precisely in [OHS98] and [KAW00]. Its details are not included in this paper. Refer to these papers.

5.5 General Scheme of Program Generation – Program Design by Object Designer

To summarize, the process of program specification and generation is as follows. It is assumed that the specification of the design object in which the computerized control system is to be embedded has already been given.

[1] Model Building

(1) User refers the past similar cases of object design and retrieve the compounds of activities used there.

(2) User decides tentatively a set of activities referring the cases and adding his/her own if necessary.

(3) User makes semantically legal connections in the groups of activities and/or macro activities. These operations are achieved on the display and the system generates the corresponding rules.

(4) User checks the model my means of syntax checker to achieve partial inference.

(5) The system visualizes this process of building a model.

(6) User modifies the model if he/she thinks it is necessary.

[2] Problem Solving and Program Generation

(1) Problems are given to the system in the form of queries on the output activities of which the output variables are substituted by arbitrarily selected instance values.

(2) A tentative instance solution tree is made by deduction.

(3) The solution tree is generalized by recovering the instances by the original variables with domains. The processing time is estimated to the solution tree.

(4) For the set of solution trees, a simulation is achieved to test for the time constraint for the operation as a whole.

(5) Each solution tree is translated to a program.

(6) Final simulation is executed for assuring the operation in the very-near-to-real environment. A special simulation system has already been developed and is being used in many automotive-manufacturing companies.

[3] Program Specification as an Intermediate State

This system is being developed now. Even if it may succeed however, some time may be necessary until it is accepted in the field because it urges the field engineers to change the way of development of vehicle engine control system. This research project therefore includes an additional part to generate specifications in the conventional style from the model. It is to be used for developing programs by human programmers in the traditional way. This total process is illustrated in Figure 5.

It is possible to represent formally the problem solving in the latter half of this process using meta-rules. Then this part can be automated. But its representation is complicated and is not included in this paper.

6 Conclusion

A new style of program generation was discussed. A key concept is model and its building process. What users want in programming are represented in the form of model in a comprehensive way for users. A computer system interprets the model and generates programs automatically as automatic problem solving. This idea is being applied to make programs for an embedded system in vehicle-engine to control its operations. The objective is to enable engine designers as non-specialists of programming to develop programs themselves. The authors expect that this method will bring a large effect on vehicle-engine design not only on the cost and development time but also on the possibility of developing new ideas in vehicle-engine design. So far the control part of engine was almost invisible to the designers on the other part of the engine because of the substantially different technology used there from the other parts. By introducing a comprehensive model, they become able to look through inside this part and to consider whole parts inclusively. To keep balance in consideration throughout the objects is the most important attitude for the designer in designing a complex object.

Acknowledgment. The authors wish to thank T.Aida of Waseda University, K.Nakamura and K.Shiozawa, and M.Genba of AdIn Research, Inc., M.Amano of

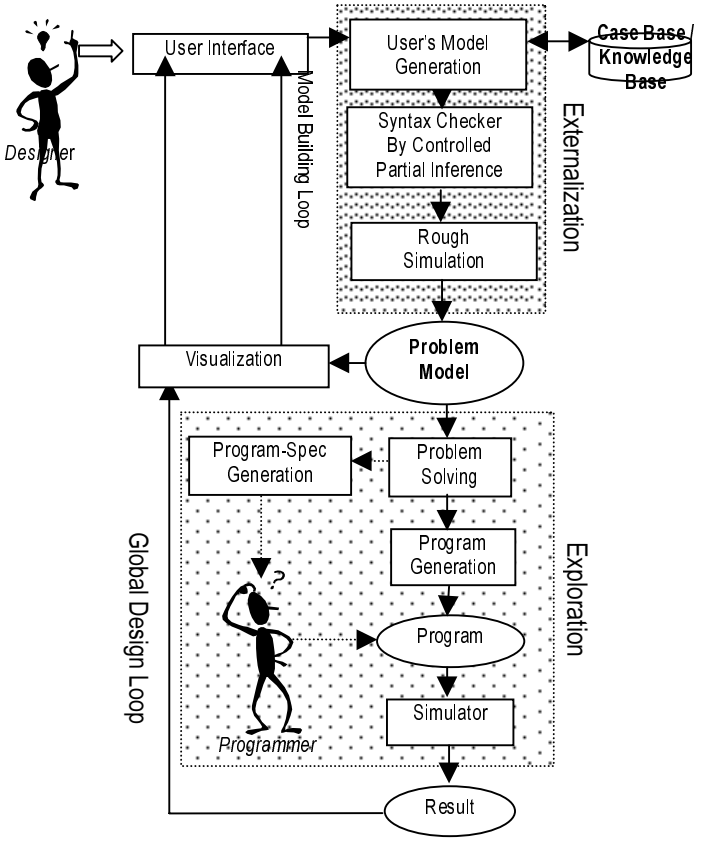


Fig. 5. Total process of program design and generation system

Toyota System Research Inc., T.Masui of Toyota Motor Corporation for valuable comments and discussions.

References

[KAW00] S.Kawasaki and S.Ohsuga; A Method of Generating Program Specification from Description of Human Activities, (to appear in) Proc. ISMIS'2000, 2000

[SAT97] K.Sato; Real-Time OS for Automotive Control Systems, Toyota Technical Review, Vol. 47 No.1

[OHS96] S. Ohsuga; Multi-Strata Modeling to Automate Problem Solving Including Human Activity, Proc.Sixth European-Japanese Seminar on Information Modelling and Knowledge Bases,1996

[OHS98] S. Ohsuga; Toward Truly Intelligent Information Systems - From Expert Systems To Automatic Programming, Knowledge Based Systems,Vol.10, 1998

[OHS99a] S. Ohsuga; A Modeling Scheme for New Information Systems -An Application to Enterprise Modeling and Program Specification, IEEE International Conference on Systems, Man and Cybernetics, 1999
[OHS99b] S. Ohsuga; New Modeling Scheme To Let Computers Closer to Persons, International Workshop on Strategic Knowledge and Concept Formation, 1999
[OHS00] S. Ohsuga; How Can AI Systems Deal with Large and Complex Problems? (to appear in) International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), Special Issue on Intelligent Agent, World Scientific Publishing, 2000
[ROS97] F. H. Ross; Artificial Intelligence, What Works and What Doesn't ? AI Magazine, Volume 18,

Classification Based upon Frequent Patterns

Wim Pijls and Rob Potharst

Erasmus University,
P.O.Box 1738, 3000 DR Rotterdam
{pijls, potharst}@few.eur.nl

Abstract. In this paper a new classification algorithm based upon frequent patterns is proposed. A frequent pattern is a generalization of the concept of a frequent item set, used in association rule mining. First of all, the collection of frequent patterns in the training set is constructed. For each frequent pattern, the support and the confidence is determined and registered. Choosing an appropriate data structure allows us to keep the full collection of frequent patterns in memory. The proposed classification method makes direct use of this collection. This method turns out to be competitive with a well-known classifier like C4.5 and other comparable methods. For large data sets it seems to be a very appropriate method.

1 Introduction

Machine learning and data mining are major issues in present-day AI research. An important issue in this field is classification. Industrial applications of classification include quality control, pattern recognition, hazard detection, load forecasting, diagnostic systems and marketing. Several classification methods have been developed over the past two decades. We mention, among others, neural networks, decision trees, genetic algorithms etc. In addition, detecting hidden rules in large data sets has become a major issue in data mining; specifically, we mention rough sets, logical analysis of data and association rules. In this paper we work out a very simple idea: to use the rules obtained by an algorithm for association rules as the rules to carry out classification. In fact, we propose a classification algorithm based upon frequent patterns. A frequent pattern is a generalization of the concept of a frequent item set used in association rule mining[1]. The first step in the algorithm is the construction of the collection of frequent patterns. Without any refining operation, those frequent patterns are used for classification. Thus, we might say that the new algorithm is a brute-force data mining method using a raw collection of patterns. Such an approach appears to be feasible, which is not surprising given today's high values of speed and memory size. We will compare our classifier with other existing classifiers.

Overview. The outline of this paper is as follows. Section 2 recalls some definitions and facts about frequent patterns. The new classification algorithm

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 72–79, 2001.

a	b	c	*class*
1	1	3	2
1	1	3	2
1	2	1	1
1	2	1	1
1	2	3	2
1	2	3	1
2	1	3	2
2	1	3	2
3	1	2	1
4	2	1	1
4	2	1	2
4	2	3	1
4	2	3	1

Fig. 1. An example: a data set S

is presented in Section 3. We also design a variant algorithm which actually performs a relaxation of the classification procedure. The algorithms are illustrated by experimental results. In Section 4 we show that there is a clear relationship between our method on the one hand and three other methods on the other, viz. classifying based on associations, rough sets and logical analysis of data. Section 5 draws conclusions and points to some further work.

2 Frequent Patterns

In this section, we discuss some definitions and facts on frequent patterns. A data set is a set of *records* or *cases*. Each record consists of an n-tuple (n is fixed) of discrete values. The positions in a record correspond to attributes. Suppose that we have $n = 4$ and the four attributes are: income, married, children, creditworthy. An example of a record in a data set of customers is *(5, yes, 3, yes)*, which means that the customer related to this record has income group equal to 5, is married, has 3 children and is creditworthy. As mentioned above, we require discrete values. If an attribute has continuous values, discretization is required. One attribute is appointed to be the target attribute. The values which are taken by the target attribute are called *classes* or *class labels*. Figure 1 shows a simple example of a data set, which has four attributes: a target attribute and three other ones called a, b and c.

Suppose that a data set D has n attributes $a_1, a_2, \ldots, a_n$, where a_n is the target attribute. A *pattern* (also called an *item set*) is defined as a series of m ($m \leq n-1$) equalities of the form $(a_{i_1} = v_{i_1}, a_{i_2} = v_{i_2}, \ldots, a_{i_m} = v_{i_m})$, where a_{i_k} is a non-target attribute and v_{i_k} is a value that can be taken by attribute a_{i_k}, $1 \leq k \leq m$. A record R is said to be a *supporter* of a pattern $P = (a_{i_1} = v_{i_1}, a_{i_2} = v_{i_2}, \ldots, a_{i_m} = v_{i_m})$, if R matches pattern P, i.e., attribute a_{i_k} occurs in R and has value v_{i_k} for each k, $1 \leq k \leq m$. The *support* of a

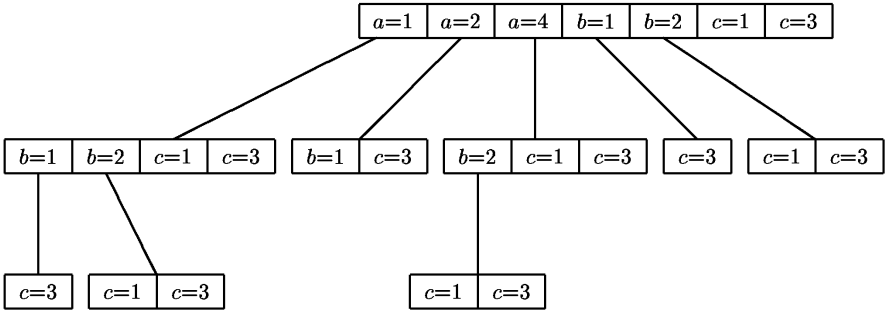


Fig. 2. The frequent patterns of S stored into a trie.

pattern P, denoted by $supp(P)$, is defined as to the number of supporters of P. Given a threshold or minimal support (denoted by *minsup*) a pattern is called *frequent* if $supp(P) \geq minsup$.
The best-known algorithm for finding frequent patterns is Apriori [1]. In the originating paper, a *hash tree*, a tree with a hash table in each node, was proposed to represent itemsets. We utilize a different data structure which replaces the hash nodes by completely filled arrays with dynamic length. This data structure is equivalent to a trie[2]. It stores the full collection of frequent patterns in an efficient and compact way. An example of a trie is shown in Figure 2, which displays the set of frequent patterns in data set S of Figure 1 for *minsup*= 2. Each path from the root, not necessarily ending at a leaf in the trie represents a frequent pattern and vice versa. For instance, $(a = 1, b = 2, c = 1)$ denotes a path as well as as a frequent pattern. The property that any heading subpattern in a frequent pattern is frequent as well, makes a trie a proper data structure for storing frequent patterns. So, the subpaths $(a = 1, b = 2)$ and $(a = 1)$ (a path of length 1) also represent frequent patterns. The patterns $(a = 3)$ and $(c = 2)$ and any extensions of these patterns are infrequent and hence not included in the trie of Figure 2.

As mentioned before, $supp(P)$ is equal to the number of the supporters of P. For a given class c and a pattern P, the class support of c in P, denoted by $supp_c(P)$, is equal to the number of supporters for P with class label c. The confidence of a class c, also called the relative support, in a pattern P is defined as $conf_c(P) = supp_c(P)/supp(P)$. The class of a pattern P, denoted by $class(P)$, is defined as a class c for which $supp_c(P)$ and hence also $conf_c(P)$ is maximal. For $c_0 = class(P)$, the value $conf_{c_0}(P)$ is also denoted by $conf(P)$. The following examples illustrate the definitions. Consider patterns $P = (c = 3)$ and $Q = (b = 2, c = 3)$ in data set S in Figure 1. The following quantities hold for P: $supp(P) = 8$, $supp_1(P) = 3$, $supp_2(P) = 5$, $conf_1(P) = 3/8$, $conf_2(P) = 5/8$. Since class label 2 has the greatest support, we have $class(P) = 2$ and $conf(P) = 5/8$. For Q we have: $supp(Q) = 4$, $supp_1(P) = 3$, $supp_2(P) = 1$, $conf_1(P) = 3/4$, $conf_2(P) = 1/4$, $class(P) = 1$ and $conf(P) = 3/4$.

3 A Classifier Based upon Frequent Patterns

In this section we present our new classifying algorithm, which we call PatMat, since it is based on matching frequent patterns.

PatMat. To build a classifier we need a data set D which acts as training set. The collection of frequent patterns in D is constructed and is stored into a trie T. Any algorithm designed to find frequent patterns is appropriate. Apriori is the best-known algorithm in this area. We assume that $supp(P)$ and also $supp_c(P)$ for every c is stored into the cell at the end of path P. To calculate these quantities, only a slight extension of Apriori is needed. The classification of a record R proceeds as follows. Each path or equivalently each frequent pattern P is examined for being a supporter of R. The pattern P for which $conf(P)$ is maximal is selected and its class value $class(P)$ determines the class for R. The following code reflects the above classification process.

Algorithm PatMat
(1) **function** *classifying*(R)
(2) $bestconf = \max\{conf(P) \mid P$ a frequent pattern supported by $R\}$;
(3) P_0 is a frequent pattern such that $bestconf = conf(P_0)$;
(4) c_0 is a class such that $conf(P_0) = conf_{c_0}(P_0) = bestconf$;
(5) **return** c_0, P_0;

In order to determine the value of *bestconf* (see line (2)), the trie needs to be traversed. Hence, for any record P that is being classified, a trie traversal is performed. Clearly, some parts of the trie may be discarded. If R does not support a pattern corresponding to a cell P, then R does not support any extension of P either. In case cell P is visited, the subtrie below P (if any) may be discarded.
Theoretically, it might happen that there is no frequent pattern supported by R (see line (2)). However, it can be shown that the occurrence of such an event is very improbable. In fact, in all our experiments this did not occur a single time. If we want the algorithm to cover all possible cases, we could assign a default class to such a record R, for instance the most frequent class.
From the fact that the confidences are relative frequencies approximating pattern probabilities, it can be seen that the algorithm is an approximation to the Bayes classifier under the assumption that the training set has proportions that reflect the class probabilities. By restricting attention to *frequent* patterns we try to enhance the quality of the approximation.
An important parameter to be tuned for our method is the value of *minsup*, the minimal support that is allowed for the patterns to be represented in the trie. Since the number of frequent patterns decreases when *minsup* is raised, we want *minsup* to be as low as feasible in view of the memory needed for the trie. However, even for small problems where a complete trie with *minsup*= 1 could be held in memory, we noticed a slight deterioration of performance for too small values of *minsup*. For all data sets we have examined so far with this method a *minsup* of about 1% of the total number of records in the complete data set was found to be optimal.

We conducted several experiments with PatMat. We will give a more detailed description of those experiments at the end of this section. We noticed many collisions in line (3), mostly with *bestconf* = 100%. A collision means that patterns P and P' are found with *bestconf* = *conf*(P) =*conf*(P') or equivalently, pattern-class combinations (P, c) and (P', c') are found with *bestconf* = $conf_c(P)$ = $conf_{c'}(P')$. We formulated the following criteria for solving collisions.

- if $conf_c(P_i)$ =$conf_{c'}(P_j)$ and $supp_c(P_i)$ >$supp_{c'}(P_j)$ for any classes c and c', then P_i takes precedence over P_j;
- if the previous rule still leaves a collision, then a longer pattern (i.e., a higher number of attributes) precedes a shorter one.

Applying these criteria the number of collisions nearly vanishes. Once a pattern P_0 is chosen, usually a unique value c_0 can be determined in line (4).

A major advantage of the new method over other methods is its transparency. Note, that the classification function not only returns c_0, the appointed class, but also P_0, the pattern that caused c_0 to be picked as the appointed class. Thus, it is possible to give a clear account for the target label assigned to a test record. A record is put into a class, since it meets a certain pattern and many training records meeting that pattern belong to the same class. Thus, an instance could be classified as class A, because it has value 3 for attribute 2, and value 1 for attribute 4, and 97% of such instances in the training set belong to class A.

A variant algorithm. We have mentioned that line (3) in the code of the procedure *classifying* gives rise to a lot of collisions. In addition, we noticed that many so-called *near-collisions* occurred. A *near-collision* means that a combination (P, c) can be found with *bestconf* $\approx$ $conf_c(P)$ or more precisely $conf_c(P) > 0.9 * bestconf$. Because of the great number of collisions, we decided to allow multiple classification: a record may be classified into multiple classes, or put another way, a record may take multiple class labels. In order to take the collisions into account, the following code was used.

Algorithm Variant
(1) **function** *multiple-classifying*(R)
(2) *bestconf*= max{*conf*(P) | P a frequent pattern supported by R};
(3) $\mathcal{C} = \{c|$ $conf_c(P) > 0.9 * bestconf$ for any frequent P supported by $R\}$;
(4) **return** $\mathcal{C}$;

A record R is considered to be classified wrongly, if the class label of R does not occur in $\mathcal{C}$, the set resulting from the call *multiple-classifying*(R). As will be shown in the next paragraph, the variant algorithm has a considerably lower error rate.

Experiments. In this section we will describe some experiments. The new algorithms, PatMat and its variant, were implemented under the Borland

C++ 5.02 environment. These algorithms were compared with C4.5, a well-known decision tree classifier. For our experiments we actually used the Java implementation of C4.5, Release 8 from [11]. We used eight data sets from the UCI ML Repository[8]. If a data set contained continuous attributes, these were discretized using the method in [5]. The three algorithms under consideration were applied to each of the eight data sets. In order to compare the performance of the algorithms, we measured the error rate of each classification run. The error rate is defined as the number of wrongly classified records divided by the total number of classified records. The difference between the error rate of PatMat and that of the Variant algorithm is a measure for the number of collisions during PatMat. A test run for a combination of a data set and an algorithm consists of the following steps. A random selection is made containing 2/3 of the records in the data set. This set acts as the training set. The remaining 1/3 of the records is used as the test set. The error rate was measured in each run. This action was carried out ten times. The average error rate and the standard deviation were computed over these ten runs. The outcomes are displayed in Figure 3. For finding frequent patterns in PatMat and its variant, $minsup = 1\%$ is chosen. Figure 3 shows that PatMat performs slightly better than C4.5. Remarkably, we noticed that over 90% of the records were classified with $bestconf = 100\%$ and almost all records were classified with $bestconf \geq 90\%$. None of the test runs took longer than a few seconds, although some of the data sets consist of thousands of cases.

	C4.5		PatMat		Variant	
	aver	stddev	aver	stddev	aver	stddev
Austra	15.3	3.14	14.5	2.05	3.1	0.81
Breast-w	5.3	2.03	4.6	1.66	1.5	0.63
Diabetes	21.4	3.41	21.4	1.73	17.7	2.23
German	28.2	2.03	26.9	1.63	17.0	2.30
Glass	30.8	6.33	28.1	5.30	20.9	4.00
Led7	27.6	1.26	28.0	1.02	21.8	1.58
Lymph	25.6	4.77	18.9	5.55	2.8	2.08
Heart	24.4	2.85	18.6	2,52	1.5	1.34
Pima	22.3	1.43	21.0	2.21	16.5	2.00

Fig. 3. Error rates of classification methods

4 Comparison with Other Methods

Several classification algorithms in machine learning construct a classifier that consists of a list of rules of the form $P \Rightarrow c$ where P is a pattern and c a class. In contrast with those algorithms, we do not construct such a list, but we utilize the complete trie of frequent patterns to classify each new case. Our algorithm uses an interesting mix of lazy and eager learning. At first sight, it resembles

lazy learning methods like k-nearest-neighbor or IGTree [4], since it starts to look for interesting patterns only after it is given a new case. However, unlike is the case with those methods, our trie is not a lossless compression of the training set. During the construction of the trie, generalization takes place by restricting the attention to *frequent* patterns. In the remaining portion of this section we will compare our classification method with three methods known from the literature on classification.

Closest to our algorithm is the CBA method[6,7], a rule-based classifying system. A rule in a CBA classifier is nothing but a frequent pattern enhanced with a class number. Since the number of patterns and hence, the number of rules is extremely high, a selection needs to be made. Only those rules are preserved that classify at least one case from the training data. For many data sets, this still results into a list of ten thousands of rules. An intricate method is applied to extract this set from the collection of frequent patterns in an efficient way. Eliminating rules that do not classify any case of the training set may be harmful. Some of these rules may apply to unseen cases. We found a slight deterioration of the performance when such rules were left out. Therefore, our trie contains *all* frequent patterns without restrictions.

Another rule-based classifying algorithm is provided by the Rough Sets method, abbreviated as RS here. This method was designed by Z. Pawlak[9]. To explain the relation between PatMat and Rough Sets, we introduce some new notions. A pattern is defined to be *perfect* if its confidence equals 100%. A perfect pattern is called minimal, if it is no longer perfect when one attribute is removed from the pattern. For any perfect pattern P, at least one minimal perfect subpattern P' can be found. It is easily seen that every extension of a perfect pattern P is perfect as well and has the same class label as P. Consequently, classifying with a perfect pattern P is equivalent to classifying with a related minimal perfect subpattern P'.
In the Rough Sets theory, a central role is played by so-called *reducts* and *value reducts*. Any classification rule in RS has a value reduct in its left-hand side. The following interesting relationship holds: a minimal value reduct is equivalent to a minimal perfect pattern. It follows that every value reduct in RS with sufficient support also occurs in the trie used by PatMat. Consequently, RS assigns the same class label to any record as PatMat does, provided that collisions, if any, are resolved in the same way. The converse doesn't hold. Records that are classified in Patmat by a pattern with confidence $< 100\%$, cannot be classified at all by RS, since RS requires 100% confidence. For those records, RS resorts to a default class. Only recently, some generalized models of RS have been introduced, in which 100% implications are replaced by approximate dependencies, see e.g. [12]. As mentioned in the previous section, we noticed in our experiments that over 90% of the records was classified by a pattern P with $conf(P) = 100\%$. This phenomenon makes RS and PatMat similar.

The Logical Analysis of Data (LAD) method developed by a group around P. Hammer[3] is very similar to the Rough Set method, but has the further drawback that it works only for boolean attributes. The patterns used in the LAD method are equivalent to value reducts in RS and hence also to minimal perfect patterns in PatMat.

5 Concluding Remarks

We have presented a new classification method based on frequent patterns. It appears to perform adequately, in some cases it scores clearly better than C4.5. The main advantage of the method above the existing methods is its transparency: it is very easy to state the reason why an object is classified as it is. It is argued that our classification method generalizes the Rough Sets and LAD methods, making it more appropriate for noisy data and large data sets. Presently, we are developing an extension of the method proposed in this paper to be used for target selection in direct marketing.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo, *Fast Discovery of Association Rules*, Chapter 12 in: U.M. Fayyad et al. (eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, pp. 307-328, 1996.
2. A. V. Aho, J.E. Hopcroft and J.D. Ullman, *Data Structures and Algorithms*, pp. 163-169, ISBN 0-201-00023-7, Addison-Wesley Publishing Company, 1983.
3. E. Boros, T. Ibaraki, E. Mayoraz, P. Hammer, A. Kogan and I. Muchnik, *An Implementation of Logical Analysis of Data*, IEEE Transactions on Knowledge and Data Engineering, Vol. 12, No 2, pp. 292-306, March/April 2000.
4. W. Daelemans, A. van den Bosch and A. Weijters, *IGTree: using trees for compression and classification in lazy learning algorithms,* in: Artificial Intelligence Review 11, pp. 407-423, 1997.
5. U. M. Fayyad and K.B. Irani, *Multi-interval discretization of continuous-valued attributes for classification learning,* IJCAI-93, pp. 1022-1027.
6. Bin Liu, Wynn Hsu and Yiming Ma, *Integrating Classification and Association Rule Mining,* in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York, 1998.
7. Bing Liu, Yiming Ma, and Ching-Kian Wong, *Improving an Association Rule Based Classifier*, Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2000) Lyon, France, LNAI 1910, pp. 504-509.
8. C.J. Merz and P. Murphy, *UCI Repository of Machine Learning Databases*, `http://www.cs.uci.edu/~mlearn/MLRepository.html`
9. Z. Pawlak, *Rough Sets, Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht etc., 1991.
10. J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1992.
11. Ian H. Witten and Eibe Frank, *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, 2000.
12. W. Ziarko, *Variable Precision Rough Set Model*, J. of Computer and System Sciences, Vol. 46, pp. 39–59, 1993.

An Evolutionary Approach to Constraint-Based Timetabling

Dharmendra Sharma and Nivlesh Chandra

Department of Mathematics & Computing Science
University of the South Pacific
Suva, Fiji Islands
{sharma_d, chandra_n}@usp.ac.fj

Abstract. In this paper we present a timetabling system that uses a combination of a genetic algorithm and constraint satisfaction to maximally satisfy constraints of different strengths. A prototype system is presented for university examination timetabling. Experiments performed with different parameter settings for the genetic algorithm on some real world data are reported. The results obtained from the prototype are promising. An extension to the system is proposed to support incremental processing of user-supplied constraints. This is needed to support user-guided exploration of the solution space and to capture the incremental nature of human timetabling.

1 Introduction

One of the most interesting problems for computer solution is the timetable problem which has been traditionally viewed as a problem in Operations Research. This problem has important practical applications and has been extensively studied. It is known to be NP-hard [2, 3]. The problem typically incorporates many nontrivial constraints of various kinds and has attracted much interest for solving it using several metaphors. Since the first attempted genetic algorithm-based solution to the problem (as reported in [1]) several attempts have been made to further study and develop genetic algorithms to solve different kinds of timetable problems.

Timetabling of the end of semester examinations at the University of the South Pacific (USP) had motivated the present study. Examination timetables at USP in the past have been mainly constructed by hand, using previous timetables, relying on past experiences and feedback from students. A timetable that is thought of to be the best is one i). that minimises clashes so that no student is assigned to sit more than one paper at the same time, and ii). in which there is some time gap between the time that a student finishes one examination and sits for the next. This timetable is initially released and feedback is sought from the students and staff. If clashes that may have been overlooked are found, the timetable is revised to reflect this. The task of examination timetabling is an instance of the timetable problem. Constraints are used to direct search for a solution from a combinatorial space of possibilities.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 80–92, 2001.

Genetic algorithms (GAs) are stochastic algorithms that are well known to work for optimisation problems generally [4] and have a good scope for application to timetable problems since these problems require optimisation of constraints. GAs are search methods that model the natural phenomena of genetic inheritance and Darwinian strife for survival [5].

The present work reports on a genetic algorithm approach that finds an optimal timetable which best satisfies all the given constraints. – extending the work reported in [6]. The progress made in developing a timetabling system using genetic algorithms and constraint processing is reported. The results from tests of the developed prototype system (called ET for examination-timetabler) on real data from USP and the experiences gained in extending the system to perform incremental constraint processing using genetic algorithms to support user-guided exploration of solutions are presented and discussed

2 The Problem

The examination timetable problem can be described by

- a list of classes with known class sizes,
- a list of rooms with known capacities,
- a list of three-hour sessions (or time periods), and
- a list of constraints.

At USP, there are about 320 classes, about 15 rooms of various capacities are used and there is currently a 10 day (with 2 sessions per day) exam period.

The problem is to find the optimal timetable (class-room-session) satisfying a given set of constraints which usually comprises of a mixture of *hard* constraints (constraints that have to be fully satisfied) and *soft* constraints (constraints that may be *relaxed*).

The basic problem is to assign examinations for known class sizes to a known number of sessions (or time periods) such that there are no *conflicts* (or *clashes*), the room capacities are not exceeded and all the hard constraints and a maximal subset of soft constraints are satisfied. A typical timetable is a tuple <*course*, *day*, *session*, *room*>.

Some common constraints that are observed in practice at USP are given below.

1. Certain exams are required to have at least x number of sessions between them and certain other exams – for example, any two courses taken by a student must not have examinations at the same time.
2. Precedence constraints between exams restrict the ordering.
3. There is a limit on number of examinations that can be held at the same time.
4. Certain courses may have examinations at the same time.
5. Multiple exams may be held in one location.
6. Exams may be split over several locations.

The primary objective is to find a conflict free timetable. However, there are several secondary objectives:

1. To have the examinations spread out as evenly as possible for each student.
2. To schedule examinations for large classes early in the exam period.
3. To optimise (i.e. to lengthen or shorten) the duration of the exam period as desired.

The above constraints can be classified into two groups:

I. constraints that are always required to be satisfied in the timetable, and
II. constraints which are attempted to be satisfied but may be violated.

The Type II constraints include user-supplied preferential constraints. Their satisfiability is desired in timetables but if they cause a violation of Type I constraints then they are relaxed. Some examples of Type I constraints can be that there should be no overcrowding in the rooms and two courses which are taken by the same student should not be scheduled to be examined together. An example of Type II constraint can be that the rooms should be used to its maximal limit, that is, there should be minimal wastage of space when allocating courses to rooms.

3 ET Constraints

For the development of ET the following constraints have been considered.

1. If a course is unable to be scheduled in a way that satisfies all the constraints, there is no attempt to decompose it into smaller groups so that these could then be assigned different rooms but their examination done concurrently (where such assignments result in satisfaction of all the constraints).
2. There are three kinds of constraints that are considered in order to schedule the examination timetable. These are as follows.

    a). There is to be no overcrowding in the rooms (that is, the total number of students in a room should not be greater than the capacity of the room).

    b). Certain courses cannot be scheduled at the same time. This is due to the fact that if a student is taking more than one course then he/she cannot have the examination of two courses at the same time.

    c). There is to be a minimal wastage of room space. This means that the rooms that are scheduled to have examinations should be at least 80% full.

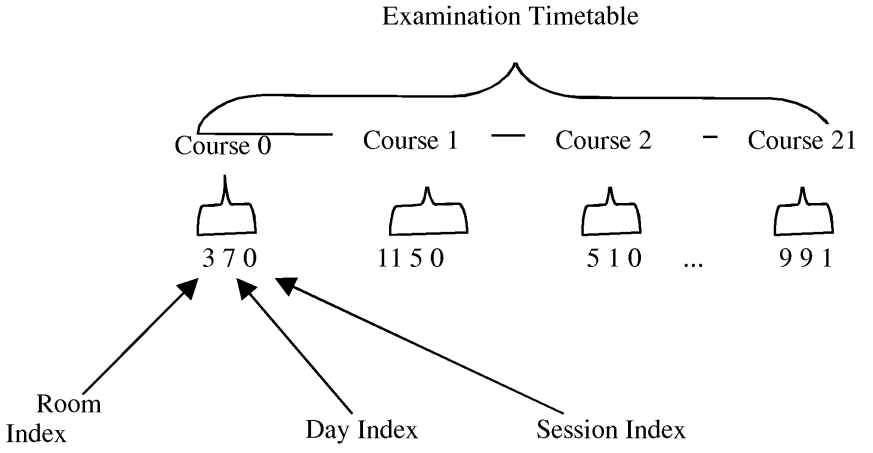
Although it would be desirable to satisfy all the constraints as listed above, some relaxation may be required and a mechanism for relaxation has to be provided. The main constraints that must be satisfied are (a) and (b) as listed above. Although it

would be excellent if (c) were also satisfied, but if the satisfaction of (c) means a violation of either (a) and/or (b), then it is best to relax (c).

4 Representation

Before GAs can be made use of, the chromosome structure of the individuals (i.e. a template for the solution) in the population needs to be defined for the problem. This is same as finding the encoding scheme to represent a phenotype (in this case a examination timetable) into a genotype (a bit string on which GAs can work on). Traditional GAs use bit strings as chromosomes but in the current work a slight deviation that gives some improvement in performance has been made. Instead of using bit strings, a string of integers is used. Using this representation causes a vast decrease in the length of the chromosome, thus making it more efficient for processing at the same time giving the same flexibility as in traditional encodings.

For example, as a sample problem of timetabling 22 courses (that are currently offered in Semester II by the Mathematics and Computing Science Department) over a span of 10 days where each day has 2 sessions, the chromosome encoding is as follows.



The chromosome is made up of genes. These genes correspond to the courses and their respective locations on the chromosome denote the course number. Each gene has 3 alleles. These correspond to the room, day and session where the course is scheduled. In this respect the chromosome of each individual corresponds to a possible examination timetable.

5 Algorithm

Below is a version of genetic algorithms (adapted from [4, 7]) that is used in ET.

```
Step 1.  Generate the initial population and set the
         generation counter to zero.

Step 2.  Find the fitness of the initial population.

Step 3.  Select the individuals for the next generation
         and put them in temporary storage.

Step 4.  Apply genetic operators on the individuals in
         temporary storage.

Step 5.  Find the fitness of the individuals in
         temporary storage.

Step 6.  Replace the old generation with the
         individuals from the temporary storage.

Step 7.  Increment the generation counter by 1.

Step 8.  If termination criterion is not satisfied,
         loop back to Step 3.
```

The important features of the algorithm are summarised as follows.

5.1 Production of the Initial Population

The initial population is produced randomly by assigning random values to each of the alleles. Since each of the alleles have a specified domain of values that it can take, the choice of values for each one of them is not purely random.

5.2 Fitness Calculation

The method of calculating the fitness of each individual is twofold. Firstly the chromosome structure (genotype) of an individual is translated into a timetable (phenotype). Based on this, penalties are allocated. Listed below, is the scheme that has been used to allocate penalties. Each of the entries in the timetable is checked to ensure that there is no violation of the three constraints. If any entry in the timetable violates any one of the constraints the respective penalty is accumulated for that individual.

CONSTRAINTS	*PENALTY VALUE ALLOCATED*
Room Overcrowding	47
Course Constraints not satisfied	47
Wastage of Space in Rooms	
If total room capacity used is less than 50%	8
Else	
If total room capacity used is less than 60%	5
Else	
If the total room capacity used is less than 70%	3
Else	
If the total room capacity used is less than 80%	1

The wastage of space in Rooms constraint is further decomposed because although a room full to at least 80% of its capacity is desirable, it should be noted that this constraint could be relaxed. One way of implementing this is by having different penalty levels which reflect different degrees of enforcement to each of the conditions expressed in the constraints. For instance, a room, which is less than 50% full is less desirable, compared to one that is 60% full. But if a timetable, which has all the rooms at least 60% full cannot be found then one which has at least 50% full rooms, is acceptable.

The values for the penalties are empirical. They were found to best suit the problem (that of penalising individuals in the population for not satisfying the constraints). Other values can be used but it should be noted that the more important it is to have a constraint satisfied, the higher a penalty value should be allocated to individuals that violate it. This concept adequately caters for constraints of different strengths.

Once the penalty for each individual is calculated, the second step of finding the fitness value from the penalty value is performed. The reason to change penalties to fitness values is trivial and relates to the basis for GAs – they are optimisation tools used to maximise functions. In order to get a timetable that closely satisfies the constraints, there is a need to minimise the penalties of the individuals. The lower the penalties of the individuals, the better they satisfy the constraints. Thus to calculate the fitness, there is a need to do a mapping so that the individuals with a higher penalty are mapped to a lower fitness and vice-versa.

The following model was initially used to map a penalty to a fitness value.

$$g(x_i) = 1/P(x_i)$$
$$F(x_i) = g(x_i) / \sum_i g(x_i)$$

where $P(x_i)$ is the penalty of the individual x_i
$g(x_i)$ is the fitness of individual x_i
$F(x_i)$ is the fitness value between 0 and 1 for the individual x_i.

The motivation for this model was that for higher penalty values, the reciprocal gets mapped to small values and vice-versa. But one inherent problem with this

method was that when the reciprocal of a large number was calculated, the result could not be taken to be the true value since this operation is prone to round-off and truncation errors [5].

Thus an alternative model (as given below) was used. This formula was derived intuitively and was tested for correctness and checked to make sure that it mapped large penalty values to small fitness values and vice-versa.

$$F(x_i) = (2*A - P(x_i)) / S$$

where $F(x_i)$ is the fitness of the individual x_i
$P(x_i)$ is the penalty of the individual x_i
A is the average penalty of the population (for the current generation)
S is the sum of the penalties of the population

This model does a mirror effect on the penalty (having the mirror at the average penalty value of the population) to get the fitness value. The function of this model is illustrated below.

Assume that **P** is the penalty of an individual and **A** is the average penalty of the population and □ is **A** - **P**. Now if **P** > **A** then □ is negative, which means that this individual has a high penalty relative to the average of the population and so needs to be mapped to a lower fitness. But when □ is added to **A**, it has the effect of subtracting the absolute value of □ from **A**. Thus the formula maps a high penalty value to a low fitness value.

Similarly it can be seen that the formula maps a low penalty value to a high fitness value - the case where **P** < **A**.

5.3 Selection

The method of selecting an individual is based upon the roulette wheel [3]. This principle uses the concept of assigning parts of a roulette wheel based upon the fitness of an individual in comparison to the population's fitness. The higher the fitness of the individual the larger the portion of the roulette wheel it will be assigned. After the allocation is done, the roulette is spun and the individual that is allocated to the location that it stops at is selected. This method is biased towards the fitter individuals but there is a chance of a weaker individual being selected also. This is important since there needs to be diversity in the population.

5.4 Genetic Operators

These are operators that are used to alter an individual's chromosome. In the current work, four such operators are used as discussed below.

Crossover - this operator takes two individuals, finds a random point on the chromosome and swaps the genes from this point between the two individuals, resulting in two new individuals. This operator is probabilistic.

For example, let Individual A have the chromosome structure

3 4 0 4 6 0 4 2 1 6 7 1

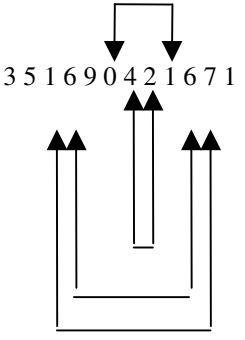
Individual B have the chromosome structure
3 5 1 6 9 1 1 1 1 2 1 0
and the crossover point is at location 4 (numbering starts from 0), then the resulting children of these individuals have the chromosome structures

child1 = 3 5 1 6 9 0 4 2 1 6 7 1
child2 = 3 4 0 4 6 1 1 1 1 2 1 0

Mutation - this operator takes an individual and changes the allele, of a randomly selected gene, to a random value that is within domain of the values that the allele can take. This operator is probabilistic.

Reproduction - When an individual does not undergo crossover, it is said to have reproduced itself. Thus this is an implied operator, which results when crossover does not get applied.

Inversion - this probabilistic operator reorders the alleles in the chromosome. Firstly two positions on the chromosome are found, say **A** and **B**. Then the allele from location **A** is swapped with that from **B**. This process is then repeated for each allele in the interval **A** - **B**. For example, assume an individual has the chromosome 3 5 1 6 9 0 4 2 1 6 7 1 and the two positions that are selected for inversion are 2 and 9 (numbering starts from zero). Then inversion causes the following:



The resulting individual has the chromosome structure of
3 5 6 1 2 4 0 9 6 1 7 1

5.5 Termination Criteria

In the tests on the prototype, the termination criterion was the current generation number. The sentinel was set as either the generation number was greater than 1000 or whether the current generation contained an individual that had a penalty value of zero.

6 Trial and Analysis

A sample trial is reported for population size: 800 and maximum generation number until which processing proceeds: 1000. For a graphical representation of the above results, refer to the Appendix.

Trial No.	Cross-over Prob.	Mut. Prob.	Inv. Prob.	Max. Penalty	Min. Pen.	Avg. Pen.	Generation when finished
1	0.0	1.0	0.0	783	306	482.28	1000
2	0.1	1.0	0.0	487	70	241.766	1000
3	0.2	1.0	0.0	580	98	254.022	1000
4	0.3	1.0	0.0	511	64	220.4	1000
5	0.1	0.0	0.1	359	359	359	1000
6	0.1	0.1	0.1	165	20	23.2	1000
7	0.5	0.3	0.2	201	6	12.844	1000
8	0.6	0.3	0.2	155	3	12.98	1000
9	0.6	0.3	0.0	150	0	10.734	906

The results above are average values that were obtained after ET was run four times for each crossover, mutation and inversion probability listed above. It is noted that mutation alone was unable to evolve individuals that had a chromosome that closely matched the solution that was sought, in this case the optimal timetable (one that had no constraints violated). A combination of the three operators, crossover, mutation and inversion, was able to generate an individual that had a penalty of 3. This individual satisfied all the Type I constraints but one Type II constraint was left unsatisfied.

The combination of mutation and crossover was able to produce an individual that had a penalty of zero. This complies with theory since mutation and crossover are two powerful operators and it is no surprise that they were able to generate a solution to this NP hard problem, which satisfied all the constraints.

After the 9th trial, an individual was found that had a penalty value of zero. The total time for execution for this trial, until the individual was found was approximately 109 seconds. This result is promising.

7 Discussion

During the trial runs, it was found that for a certain size of the chromosome, there was a lower limit below which no matter which combination of evolutionary operators was applied, the species refused to evolve an individual, which closely matched the solution that was required. Instead it got 'stuck' at a local optimum. In most instances, it was found that this lower limit was twice the size of the chromosome. Although this value was noticed after doing a limited amount of trials, it is suggested that to get a good approximation, there needed to be an extensive testing done. But nevertheless, it can be stated that there is a lower limit for the size of a population, below which GAs do not produce individuals that closely match the solution sought. One of the reasons for this phenomenon could be that for GAs to produce better individuals, the genetic pool (the combination of alleles from which individuals could be produced) must be large. (For each generation, the genetic pool consists of all the alleles that each individual in the population has.) This means that there needs to be diversity in the population, which is hard to get with a small population size.

Several methods were tried to improve the performance of the GA. One of such methods used was that of finding the best individual in the current population and seeding the next generation with this individual (a process known as *elitism* in GAs). This was done in the hope that the best individual will always be retained and thus the population fitness would increase with each generation. But quite the opposite was noticed. After some generations it was noticed that the GA was 'stuck' at a local optimum. Due to the superiority of the seeded individual, it had a high probability of being selected for the next generation. It was not long before all the generations that were generated, contained a substantial number of this seeded individual. This resulted in the convergence to a local optimum as was observed.

From the trials, looking at the movement of individuals from one generation to the next, it was found that sometimes, the fitter individuals did not make it to the next generation. This should not be the case since probabilistically they stand a better chance of surviving compared to the rest of the individuals. Although their number maybe small, at least some of these should survive to the next generation. This was found to be one of the drawbacks of the roulette wheel selection method. It prohibited genetic drift. The results could have been degraded as a result. Thus to improve on this there needs to be a better selection method needs to be incorporated that would support and actually promote genetic drift and at the same time provide a probabilistic way of selecting the individuals for the next generation (biased towards the fitter ones).

8 Incremental Constraint Processing

ET has been successfully extended to IET (for Incremental Examination Timetabling) to support incremental addition and processing of constraints using the ET genetic algorithm. IET uses the ET engine in an incremental way with modification in constraint application and selection of individuals for new generations.

For each constraint sequence given: $<c_1, c_2, \ldots, c_i>$ a corresponding generation of individuals g_i are recorded and for the next subsequence of constraints $<c_j, \ldots, c_k>$, the genetic algorithm uses g_i as the initial population to get a generation of individuals g_k maximally satisfying the combined constraint sequence $<c_1, c_2, \ldots, c_i, c_j, \ldots, c_k>$. The best individual is then chosen from g_k as the solution. If an individual meeting the required criteria cannot be found from g_k then g_i is used as the initial population to get the best solution for the complete constraint sequence $<c_1, c_2, \ldots, c_i, c_j, \ldots, c_k>$.

This procedure uses the above genetic framework in a ‘dynamic’ way and hence suffers from the same fundamental weaknesses of genetic algorithms as identified earlier. However, this approach adopts a more human timetabler-like incremental strategy and computationally, is more efficient than just the basic ET strategy.

IET utilises the fact that ET converges to a solution for each $<c_1, c_2, \ldots, c_i>$, if over-constrained no extension to the sequence yields a solution. IET works as follows. Given a set of hard and soft constraints, □

- Check consistency of □. If conflicts, resolve (user-guided).
- □ Process applicable constraints from □ and ‘anchor’ positions on the chromosomes in the initial population.

- Apply ET on resulting chromosomes and the remaining constraints (with given associated penalty values).
- IET backtracks to previous tenable position in search and asks user to relax/revise the added constraints or add any new ones.
  Each new population is generated by
  - apply GA
  - save the best solution
  - generate a new population by transferring the best individuals of the converged population and then generating the remaining individuals randomly
  - process applicable constraints from and 'anchor' positions on the chromosomes in the initial population.
  - apply ET on resulting chromosomes and the remaining constraints (with given associated penalty values).

The results obtained from this approach have been encouraging. Further experiments are being currently conducted on this approach of incremental constraint processing using genetic algorithms.

9 Conclusions and Further Work

An examination timetabling prototype (ET) has been successfully developed and tested on some real data. Some valuable experiences on using GAs to solve constraint-based problems have been gained.

The experiments show that GAs are not purely stochastic. They do involve some degree of probability but mainly they try and localise themselves to parts of the search space that hold the most promise for the optimal solution. The initial population is generated randomly but there after, each generation is based upon the fitter individuals that were obtained in the previous generation. Information from one generation to the next is passed in the form of the chromosomes of the individuals that survive. This provides a way of localising the search to an area of most promise. But since they are stochastic systems, reruns of the system with the same environment (that is, the same mutation, crossover and inversion probabilities) will not result in the same solution. Thus in order to get a solution it is best to do more than one run of the system for a particular combination of parameters and then use the average value from these trials.

GAs have proven to be effective tools for examination timetabling. The results obtained from the current prototype ET system were promising. However, ET will be enhanced in the future in the following ways with the hope of making it a general, user-interactive and incremental timetabling system to cater for both examination and lecture timetabling at USP.

- The present system needs to be 'told' the mutation, crossover and inversion probabilities, the maximum generation and the population size for which the processing is to be undertaken. The system is to be made intelligent so that it could run conduct some trial runs and based on these decide on reasonable values for each of the parameters.
- The duration of the examination period is currently fixed. The system is to also cater for the smallest possible duration or other durations as desirable.

- A facility needs to be added, which upon finding that hard constraint are satisfied but all the soft constraints are not, to deal with user-guided relaxation of the soft constraints.
- ET currently needs to be explicitly told the penalty values. It would be better if the system is made to automatically generate appropriate penalty values based on a constraint hierarchy.
- Finally, although an extension has been done to ET to cater for user exploration of solution possibilities through user-supplied constraints and an incremental processing of these constraints, further experiments need to be conducted to assess the full potential and the relative performance of the exploratory approach.

An incremental and exploratory approach to the timetabling problem will provide a basis for solutions to similar scheduling problems such as transport timetabling, nurse rostering, and machine tool scheduling. Many such problems require user-interactive exploration of solution possibilities and guidance in search for solutions.

Appendix

Sample Constraints

Spread constraints:

The format used is (n, course1, course2) meaning that there needs to be at least n+1 sessions separating course1 and course2.

(3 MA111 CS111) (3 MA111 MA131) (3 MA111 CS121) (3 MA111 CS100) (3 CS111 MA131)
(3 CS111 CS121) (3 MA131 CS100) (3 MA131 CS121) (3 MA101 CS100) (3 MA101 CS121)
(3 MA101 CS111) (3 MA102 CS121) (3 MA102 CS111) (3 MA102 CS100) (3 MA261 CS211)
(3 MA261 MA211) (3 CS211 MA211) (3 CS313 CS311) (3 CS313 CS391) (3 CS311 CS391)
(3 CS311 CS323) (3 CS323 CS391) (3 MA313 MA341) (3 MA416 MA491) (3 CS415 CS412)
(3 MA416 CS415) (3 MA416 CS412) (3 MA491 CS415) (3 MA491 CS412)

Examination rooms and their capacities:

(GYM 445) (H104 176) (S027 40) (S025 92) (S009 62) (S111 53) (S112 52) (093-001 69)
(093-105 64) (H216 35) (S014 42) (S023 40) (S026 90) (093-202 40)

Class sizes:

(MA111 426) (MA431 3)(MA331 30)(MA101 331)(MA491 2)(MA313 66)(MA261 208)
(MA341 51)(MA416 7)(MA131 267)(MA211 209)(MA102 223)(CS211 134)(CS313 70)
(CS412 4)(CS121 357)(CS100 229)(CS311 51)(CS391 16)(CS111 284)(CS415 14)(CS323 53)

Sample chromosome output as having a penalty of zero

```
0 2 1 0 7 0 0 7 0 0 6 0 0 2 1 0 6 0 0 9 1 7 3 0 0 0 0 0 7
0 0 3 1 0 3 1 0 0 0 0 9 0 0 9 0 0 9 0 0 9 1 4 1 0 7 3 0 0
0 0 0 7 0 0 7 0
```

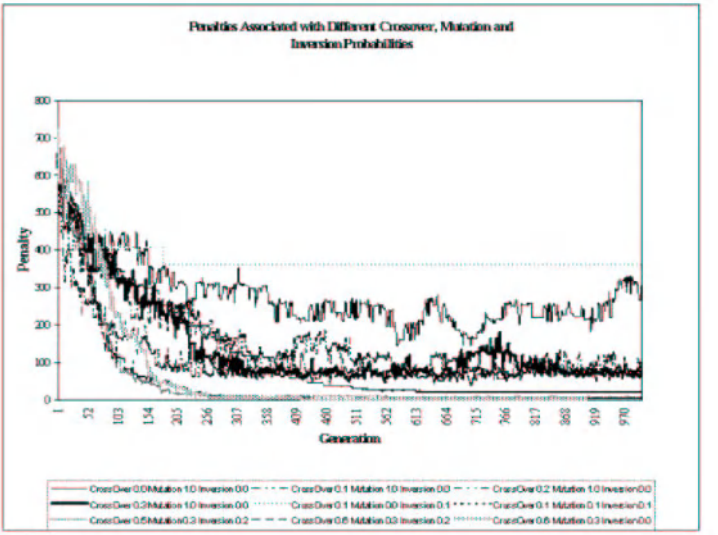


Fig. 1.

References

1. Colorni, A., Dorigo, M., Maniezzo, V.: Genetic Algorithms and Highly Constrained Problems: The Time-Table Case, in Schwefel, H.-P. and Manner, R (editors), Proceedings of the First International Conference on Parallel Problem Solving from Nature, Springer-Verlag, Lecture Notes in Computer Science (1991),Vol. 496.
2. Even, S., Itai, A., Shamir, A.: On the Complexity of Timetable and Multicommodity Flow Problems, SIAM Journal on Computing (1976), Vol. 5, No. 4, 691-703.
3. Garey, M. R., Johnson, D. S.: Computers and Intractability, W.H. Freeman and Company, San Francisco (1979).
4. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, (1992).
5. Mitchell, M., An Introduction to Genetic Algorithms, The MIT Press, Cambridge, Massachusetts, England, (1996).
6. Sharma, D., Chandra, N.: An Evolutionary Approach to Constraint-Based Timetabling, Proceedings of Constraint Processing Workshop, Australian Joint Artificial Intelligence Conference, Sydney, Australia (2000).
7. Whitley, D.: A Genetic Algorithm Tutorial, Colorado State University (1993).

BOKS: A Rule-Based System in Support of the Dutch Building Materials Regulations

Pieter Spronck and Klaas Schilstra

TNO Building & Construction Research,
Knowledge Based Systems Department,
Rijswijk, The Netherlands
P.Spronck@cs.unimaas.nl
K.Schilstra@bouw.tno.nl

Abstract. BOKS is a knowledge based system which implements the "Bouwstoffenbesluit" (Bsb), the Dutch regulations on the use of building materials. Since the Bsb is a complex set of rules, which many people need to adhere to, a knowledge based system was seen as an ideal way to allow people to check whether they are using building materials according to the law. BOKS was implemented as a first generation rule-based system using decision tables. This paper discusses BOKS, the reasons for its success as a knowledge based system, its implementation and how that implementation could be improved upon, and makes a comparison between BOKS and related systems.

1 Building Materials Regulations

At the end of the twentieth century and the start of the twenty-first, concern for the environment in industrial countries becomes more and more important and people are seeking ways to deal with the problems of industrial waste. In this respect, the Dutch government actively promotes reuse of building materials, as long as this forms no threat to the health of the environment. With the 1995 publication of the "Bouwstoffenbesluit" (a governmental decision on building materials; its name is abbreviated to "Bsb"), the government strives to protect soil and water, while at the same time stimulating the reuse of secondary building materials. Since January 1, 1999, the Bsb has become a set of official regulations, and every building contractor is legally bound to conform to them [1].

The Bsb regulations are applicable in situations where primary and secondary building materials are transported to a building site to be used there. If there is no official quality assurance of the material to be used, a specified number of samples of the material must be taken to be examined in a laboratory, to determine the quantities and emissions of the different components in the material. Based on these laboratory findings, in combination with the exact purpose the contractor has for the material, the Bsb decides whether the material can be used, and if so, under which conditions.

The main part of the Bsb is concerned with the determination of the category of the building material. The different categories are:

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 93–102, 2001.

- *Clean soil*: If the material falls in this category, it can be used without restrictions.
- *Category 1*: Category 1 materials can be used without restrictions in the situation the building contractor has specified. One of the consequences is, for instance, that if the contractor has specified that he will use the material in a layer with a specific height, he can't add to that height without running the risk of the material falling in another category.
- *Category 2*: Category 2 materials can only be used if they are isolated from the environment, and even then only in the particular situation the contractor specified.
- *Unusable*: Most building materials cannot be used at all if they don't belong to one of the preceding categories.

Besides these, there are two exemptions created for special building materials which the contractor is still allowed to use, even if they fall in the category "unusable" based on the laboratory findings.

2 Knowledge of the Bsb

The government publication of the Bsb consists of 300 pages of text [2]. This text is difficult to use, not only because of all the legalese, but also because there are numerous tables, formulas, references, schemas, footnotes and exceptions, which all have to be taken into account, and which are found distributed throughout the documents, often without references at the relevant points. Practice has shown that even Bsb experts find that it is difficult to use the Bsb correctly, because it easy to forget to take an exception or a footnote into account. On the other hand, once the Bsb has been applied to a particular situation, the results of the analysis are easy to understand.

To determine the category of a material, the Bsb expert just needs the laboratory findings and a description of the application situation. The answer needed from the expert is which of the six possible categories the material belongs to. The "input", that is, the laboratory supplied list and the situation description, can easily be understood and supplied by a layman. The "output", the category of the material, can be understood and used by a layman. The procedure to get from "input" to "output" is complex, however, even for experts, and even though the procedure is uniquely determined in the regulations.

So, while the knowledge laid down in the Bsb is well-defined, it is difficult to apply it in practice. Still, every building contractor is required to adhere to the Bsb. As early as in 1995, the Ministerial Department responsible for the development of the Bsb recognised the difficulties in making people use these regulations, unless they were made more accessible. The Knowledge Based Systems department of TNO Building & Construction Research was asked to determine if the Bsb could be implemented in a knowledge based system. TNO built a prototype, which resulted in 1998 in a project to implement a complete system. The first version was released early in 1999 under the name "BOKS" ("Bouwstoffenbesluit Ondersteunend Kennisgebaseerd Systeem"), at the same time the Bsb became a set of official regulations. The third version, implementing the latest changes to the regulations, was released in August 2000. Cur-

rently BOKS is successfully being used by about 3000 people, which amounts to the majority of the potential user base. Even the few human experts on the Bsb use BOKS, since they agree it gives more accurate evaluations than they do themselves. Reviews of BOKS in magazines and newspapers have been generally favourable.

3 BOKS

A knowledge based system is a computer program which contains expert knowledge in a certain area, and which makes this knowledge available to the user. BOKS is a knowledge based system, which contains knowledge on the Bsb, or rather, knowledge of how an expert uses the Bsb. BOKS can, just like an expert, ask the user questions about his particular situation (like "what is the name of the material?", "where is it used?" and "what are the emission values the laboratory reports?"), and after the consultation provide the user with the same conclusions an expert would reach.

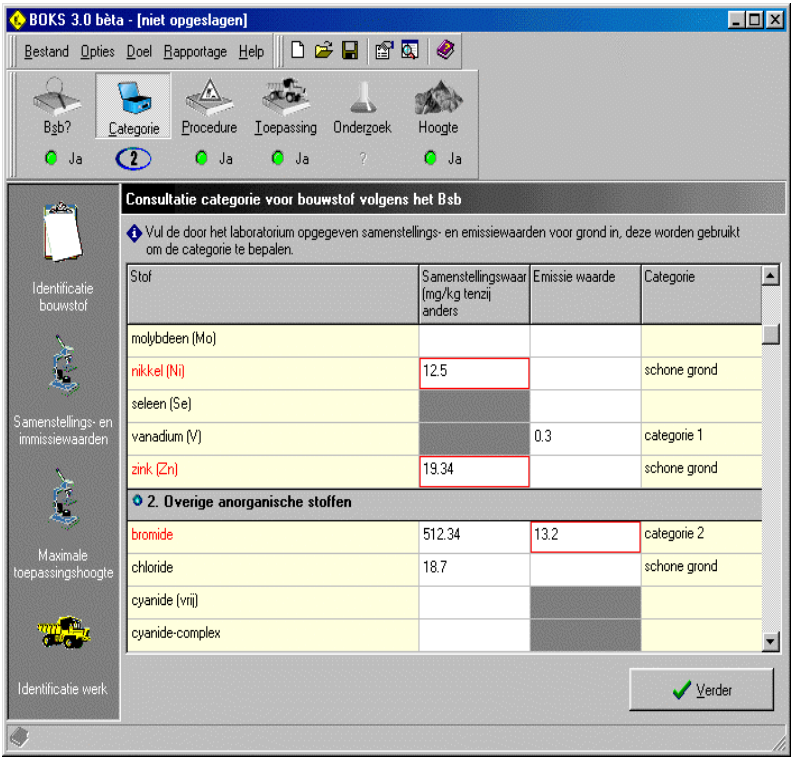


Fig. 1. BOKS shows the results of the laboratory research in a spreadsheet-like list. The rightmost column shows for each component what the category of the building material would be if that particular component would be the only one which was used to determine the category. The resulting overall category can be deduced from these intermediate categories. This final result is also visible under the category goal button at the upper left side of the screen - in this case, the encircled number "2".

The nature of the questions BOKS asks is such that a layman can answer them. Sometimes a situation arises in which a more difficult question is asked, one which most users may be able to answer, but some of them might not. In that case, the user is allowed to answer "I don't know", and BOKS will try to find out the answer to the question by asking a few different, easier to answer questions. BOKS determines in an intelligent way which questions it will ask. It contains about 150 different questions, but usually only about 30 of those are actually posed to the user during a consultation. The BOKS inference determines at each moment which question needs to be answered next.

As is usually the case with expert systems, BOKS can't discuss questions which aren't explicitly built into the system. The current release of BOKS can only determine the answer to six goal questions:

- Is the Bsb applicable to the user's situation?
- Which is the category the building material belongs to?
- What procedures must the user follow when he uses the material?
- What requirements are posed on the application of the material?
- How should a sample of the material be taken and examined?
- How high is the user allowed to stack the material?

These six goal questions (or rather, the first five goal questions) are sufficient to allow laymen to make sure they adhere to the Bsb.

4 Advantages of the Use of BOKS

The last goal question mentioned above (the determination of the maximum height the user is allowed to stack the material) is a nice example of a service BOKS provides, which can't be offered as easily by a human expert. Often, the category of a building material is limited by the height of the layer of the material which is used. Usually the user can stack the material even higher without changing the category, or have the material fall into a better category if he would just stack it a little lower. However, the formulas in the Bsb use the height of the material as input, and these formulas can't be inverted mathematically. Therefore it is not possible to directly calculate the maximum height of a material for a certain category. BOKS solves this problem by simply calculating the category for a few thousands of possible heights, thereby deducing at what levels the category changes. For a human expert this is far too much work, while BOKS performs this duty in the blink of an eye.

This is just one of the benefits the knowledge based system BOKS offers as opposed to a human expert. Some others are:

- Except for possible implementing errors in the knowledge base, BOKS is infallible and complete. A human expert may forget to take an exception to a rule into account, but BOKS won't.
- BOKS can be replicated. A human expert can only work in one place, while BOKS can be used in many places at the same time.

- BOKS can be used at any time, while a human expert is only available for part of the time.
- BOKS' results are consistent and can be reproduced. While two human experts may have a different view of a particular situation, two copies of BOKS will always provide the same answer.
- BOKS has built-in reporting facilities. For a human expert, the writing of reports is a tedious and time-consuming activity. BOKS generates its reports automatically.
- It is easy to keep the knowledge in BOKS up-to-date. Because the knowledge is implemented in decision tables with the use of a special tool (which will be discussed in the next paragraph), when regulations change it isn't difficult to locate in the knowledge base the rules which need to be changed.
- It is relatively easy to get users to adhere to new versions of the regulations. If the Bsb changes, which happens regularly, the BOKS knowledge base must be updated once and then be redistributed to the users. This is far easier and cheaper to do than retraining all the human experts.

The many benefits BOKS offers as opposed to a human expert (most of which, incidentally, are offered by knowledge based systems in general [3]) seem to suggest that a knowledge based system is to be preferred over a human expert. In the case of BOKS, this is probably true. The reason is that the knowledge domain BOKS covers, a set of regulations, is because of its nature complete, consistent, and well-defined, and is therefore particularly suited to be implemented in a rule-based system. Since a large number of Bsb experts are needed in practice, BOKS is an ideal alternative for the employment of human experts. Even if BOKS doesn't cover the Bsb completely, the questions it does answer (which are all the questions needed for people to make sure they adhere to the regulations), it answers with 100% accuracy. Most knowledge bases only cover part of the knowledge domain they implement, and aren't able to answer a question in all circumstances. As such, those knowledge bases can only be used in support of a human expert (which is, of course, also valuable). BOKS is an exception in this respect.

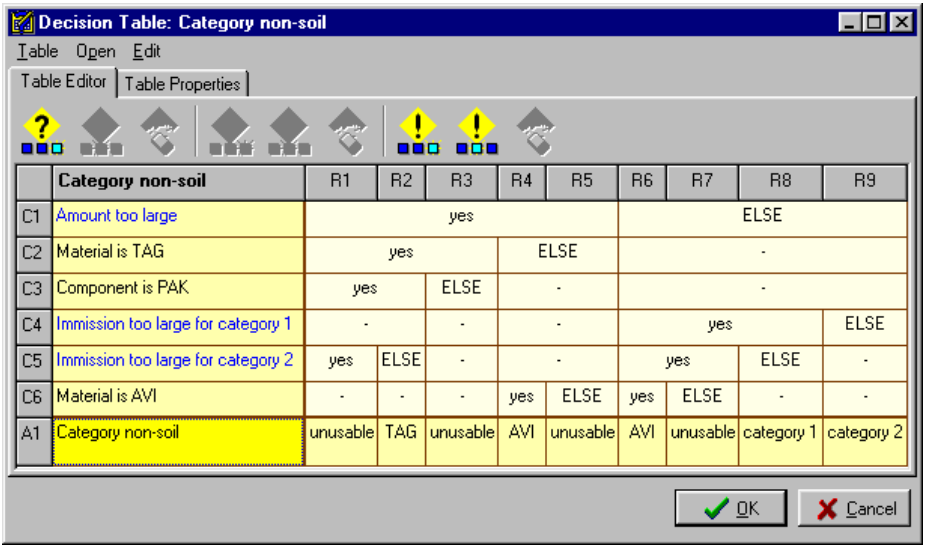
5 Implementation of BOKS

It is common practice with knowledge based systems to place the actual knowledge outside the program itself, in a knowledge base. The main reason for the existence of this separate knowledge base is that it should be easy to maintain the knowledge. Not only can such a knowledge base be changed without changing the actual program, it can also be maintained using a special tool, which allows the user to view and edit the knowledge in a user-friendly way, so even some non-programmers can do it.

A common method to store knowledge in a knowledge base is in the form of rules. Usually rules are "IF...THEN..." statements, like "IF the building material is used on the inside of a building THEN the Bsb is not applicable". There are alternative ways to implement knowledge (like in the form of "cases" or in a neural network), but a knowledge domain consisting of regulations is particularly suited to be implemented

in a rule-based system, and BOKS does just that. The BOKS knowledge base is built and maintained using the "Knowledge Base Editor" (KBE). This is a tool developed by TNO which maintains knowledge bases consisting of "decision tables" [4]. A decision table is a visual representation of a piece of rule-based reasoning (Fig. 2). BOKS contains about 180 of these tables.

The BOKS knowledge base was built by a knowledge engineer (KE) who worked in close collaboration with Bsb experts. The KE used interviews to question the experts on the use of the Bsb and implemented the results of those interviews in decision tables. This often led to new questions, which were again posed to the experts. This is a regular "propose and revise" approach to knowledge acquisition. Because of the complexity of the Bsb, the KE couldn't directly use the regulations themselves to build



	Category non-soil	R1	R2	R3	R4	R5	R6	R7	R8	R9
C1	Amount too large	yes					ELSE			
C2	Material is TAG	yes			ELSE		-			
C3	Component is PAK	yes		ELSE	-		-			
C4	Immission too large for category 1	-		-	-		yes			ELSE
C5	Immission too large for category 2	yes	ELSE	-	-		yes		ELSE	-
C6	Material is AVI	-	-	-	yes	ELSE	yes	ELSE	-	-
A1	Category non-soil	unusable	TAG	unusable	AVI	unusable	AVI	unusable	category 1	category 2

Fig. 2. This is a decision table which determines the category a non-soil building material belongs to according to one particular component. The left column (except for the bottom row) contains "conditions". To the right of the conditions are the "condition alternatives". Conditions have a value, which is determined either by asking the user for it, or by reading the value from a database, or by executing another decision table. That value is then checked against the alternatives, which determines the path that will be taken to get to the answer. The first condition is "Amount too large" (C1). If the Bsb sets an upper limit to the quantity of the component in the material and that limit is exceeded, this condition contains "yes". In that case, the left side of the alternatives (columns R1-R5) is used to continue the evaluation process, and otherwise the right side (columns R6-R9) is used. The next condition is "Material is TAG" (C2). If the quantity limit wasn't exceeded, this condition is skipped, as the dash in row C2, columns R6-R9 shows. The same holds for the condition after that (C3). By following the condition alternatives belonging to the value of the conditions, the evaluation process ends up in one of the cells on the bottom row. These cells contain the "action alternatives", and the value of the cell the evaluation ends in is given to the "action", which is contained in the lower left cell, "Category non-soil" (A1). For example, if the quantity limit is not exceeded, and the immision for category 1 isn't exceeded, the component category is "category 1".

the knowledge base, though he could use parts of it, after the experts had indicated which sections were of importance for the implementation.

The KBE allows the consultation of partially finished knowledge bases, even of individual decision tables, and the KE and the experts used this to test the implemented knowledge. The KE often chose to implement a small piece of knowledge, for instance a calculation or an exception specified in a footnote, in one separate decision table. This piece of knowledge could then be tested separately from the whole knowledge base. The decision tables themselves were also discussed with the experts. Because of the way decision tables make an inference visible, the experts were able to review the knowledge implemented in the knowledge base, and could easily determine whether the knowledge was correct and complete.

Besides a knowledge base, BOKS contains a database of building materials, for which many of the answers to questions the knowledge base might ask are already known (though BOKS is not restricted to these materials). Bsb experts can easily add to this database. The BOKS reports are also stored in a separate file, which has an XML structure, and those reports can therefore also be maintained by the Bsb experts. Last but not least, BOKS contains the complete text of the Bsb, and the reports hyperlink to the relevant paragraphs in this text. The BOKS program itself is implemented using the Borland development environment "Delphi".

The time needed for the implementation of the first version of BOKS was about 1000 man hours, not counting the work which was spent on a prototype finished in 1996. About 25% of the time went to the building of the knowledge base, 60% to the building of the application and the reports, and the remainder went to testing and administrative tasks. This excludes the time experts spent on the development of the system, but practice indicates that this must have been about half the man hours spent by the developers.

Because of BOKS' user-friendliness and the fact that BOKS only asks simple questions, people often suppose BOKS is no more than a friendly kind of spreadsheet. While the programmers are quite pleased with that, since it shows the design of BOKS is successful, that impression is not correct. Because of the complexity of the Bsb, BOKS could only be implemented in a spreadsheet if it was augmented with a lot of program code, and even then, practice has shown (because this has actually been tried), it would only be able to give partial answers to questions. On top of that, in a spreadsheet "solution" most of the knowledge would reside in program code instead of in a knowledge base, making it virtually impossible for Bsb experts to check the knowledge, let alone maintain it.

6 Improving BOKS

There are several improvements which would benefit BOKS. The first and most obvious is that the user should be able to access BOKS over the Internet. Changes to the regulations, implemented in the knowledge base, would then be immediately available to users, while currently they have to download and install a new stand-alone program. The reason that BOKS isn't an Internet based system is mainly one of budget: recent

developments to the KBE allow the deployment of knowledge bases as easily over the Internet as as a stand-alone application [5], but these techniques weren't available at the time BOKS was developed, and the budget to enhance BOKS, like with this new functionality, is very limited and goes mostly to updates to the knowledge base.

The second improvement would be to use an object oriented knowledge base for BOKS instead of a regular rule-based knowledge base. While the current knowledge base is certainly adequate for BOKS, an object oriented knowledge base would improve maintainability. At the time BOKS was developed, however, there was no tool available which would combine an object oriented knowledge model with the visual representation of knowledge (in decision tables, for instance) and the easy integration with a regular development tool. There has been demand for such a tool for some years now [6], and at TNO one is currently in development. It is called "Intelligent Objects" (IO) and will become available in 2001 [7]. If at that time a redesign of BOKS is desired, it will probably be done with this tool.

7 Comparison with Related Technologies

As a regular rule-based system BOKS has many of the features in common with classic systems such as MYCIN [8] and XCON [9], which basically belong to the category of first generation knowledge systems. As the term implies, second generation systems also exist and are considered to be more mature. These second generation systems can be seen as a response to the knowledge acquisition bottleneck, where the difficulty of transferring knowledge from the expert to a computer representation was seen to fail. This failure was commonly attributed to communication problems and a mismatch between human and computer representations [10]. A prime example of a second generation approach is CommonKADS, a knowledge engineering methodology that employs deep models to allow knowledge acquisition and maintenance to be based on the design of an operational model of the domain. Modelling the knowledge used explicitly is seen as an important aspect to construct comprehensible, controllable, maintainable systems [11].

A problem that both first and second generation systems seem to fall victim to is the gap between prototype and industrial strength knowledge systems [12]. The effort necessary to fit a knowledge system with a dedicated user interface and develop auxiliary facilities can constitute the majority of the development effort of a knowledge system, caused in part by inadequacies in the tools used. This problem negatively affects the acceptance of systems and can even lead to knowledge systems never reaching their intended users.

The focus of the effort to develop BOKS employed a different route from the one the model based approaches advise. An iterative and incremental development of the knowledge was used, with participation of the expert in the modelling process, and with a focus on developing an industrial strength system for a large group of users. This is comparable to the evolutionary and business driven approach found in regular software engineering techniques, like DSDM [13] and Rational Unified Process [14]. An important part of the development process was to design how the system would be

visualised and used, and the experts and users were also involved in decisions regarding those aspects. The fact that components were available to integrate KBE knowledge bases in a regular application, allowed great flexibility on the part of the developers to make an application to the user's liking.

Therefore, relating the development of BOKS to current directions of research, BOKS has shown that a successful system, employed by many users, can be created using a technology that has more in common with first generation than with second generation systems. This does not entail that approaches such as CommonKADS are wrong. It does, however, raise questions regarding the assumptions on why a specific approach to knowledge system development truly works well.

8 Conclusion

BOKS is an advanced, first generation knowledge based system which combines a user-friendly interface with complex knowledge on the Dutch "Bouwstoffenbesluit" (Bsb), laid down in a rule-based knowledge base using decision tables. BOKS allows laymen in the area of the Bsb to adhere to the regulations it contains, without the need of them obtaining deep understanding of the actual Bsb regulations. The system has been developed using an iterative and incremental approach to the development of the knowledge and the application as a whole, with participation of the experts in the modelling process, and with a focus on developing an industrial strength system for a large group of users. BOKS has shown in practice to be a successful and easily-used implementation of the Bsb, with a majority of the intended user base actually employing the system.

References

1. Aalbers, Th. G. *et al*.: Bouwstoffen nader bekeken. Eburon, Delft, (1998).
2. Ministerie van VROM: Bouwstoffenbesluit, parts 1, 2 and 3 (1998).
3. Turban, E.: Decision Support and Expert Systems: Management Support Systems, 2nd ed. MacMillan, New York (1990).
4. Montalbano, M.:Decision Tables. Science Research Associates, Inc, Palo Alto, California (1973).
5. Spronck, P. and S. Peeters, S.: iDoc: Rule-based Knowledge Bases on the Internet. White paper on the website of the authors' employer: www.bouw.tno.nl/bkr (2000).
6. Luger, G.F. and Stubblefield, W.A.: Artificial Intelligence, Structures and Strategies for Complex Problem Solving, 3rd edn. Addison Wesley Longman, Inc., Reading, USA (1997).
7. Spronck, P.H.M. and Schilstra, K.O.: Objectoriëntatie bij kennismodellering. In: Informatie, July/August 1999. Ten Hagen Stam Uitgevers (1999) 21-26. An English version of this paper, titled Object Knowledge Modelling, is found on the website of the authors' employer: www.bouw.tno.nl/bkr.
8. Shortcliffe, E.H.: Computer-based medical consultations: MYCIN. Elsevier, New York (1976).

9. Bachant, J. and McDermott, J.: R1 revisited: Four years in the trenches. AI Magazine 5(3) (1999) 21-32.
10. Buchanon, B., Barstow, D., Bechtel, R. *et al*.: Constructing an Expert System. In: Hayes-Roth, F., Waterman, D. and Lenat, D. (eds.): Building Expert Systems. Addison-Wesley, Reading MA (1983).
11. Schreiber, G., Akkermans, H., Anjewierden, A. *et al*.: Knowledge Engineering and Management. MIT Press, Cambridge (1999).
12. Crofts, A.E., Ciesielski, V.B., Molesworth, M., Smith, T.J., and Lee, R.Y.: Bridging the Gap Between Prototype and Commercial Expert Systems - A Case Study. In: Quinlan, J.R. (ed.): Applications of Expert Systems. Addison-Wesley, Reading MA (1989) 93-105.
13. Stapleton, J.: DSDM: Dynamic Systems Development Method. Addison-Wesley, Reading MA (1997).
14. Jacobson, I., Booch, G. and Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley, Reading MA (1998).

Software

BOKS is freely available from the Dutch Ministry of VROM. It can be downloaded from www.minvenw.nl/rws/dww/boks. A preview version of the KBE, which is used to implement the BOKS knowledge base, is available from www.bouw.tno.nl/bkr.

Using XML as a Language Interface for AI Applications

Said Tabet[1], Prabhakar Bhogaraju[2], and David Ash[3]

[1]Nisus Inc, 180 Turnpike Road, Westborough, MA 01581, USA
Said.Tabet@nisusinc.com
[2]MindBox Inc., 300 Drake's Landing Suite. 155,
Greenbrae, CA 94904, USA
Prabhakar.Bhogaraju@mindbox.com
[3]Intelligent Market Technologies, L.L.C. 330 S. Wells St. Ste. 612
Chicago, IL 60606, USA
david_ash99@yahoo.com

Abstract. One of the key advantages of XML is that it allows developers, through the use of DTD files, to design their own languages for solving different problems. At the same time, one of the biggest challenges to using rule-based AI solutions is that it forces the developer to cast the problem within particular, AI-specific, languages which are difficult to interface with. We demonstrate in this paper how XML changes all that by allowing the development of particular languages suited to particular AI problems and allows a seamless interface with the rules engine. We show that the input and output, and even the rules themselves, from an AI application can be represented as XML files allowing the software engineer to avoid having to invest considerable time and effort in building complex conversion procedures. We illustrate our ideas with an example drawn from the mortgage industry.

1 Introduction

XML (eXtensible Markup Language) is a metalanguage for representing structured data on the Web (World Wide Web Consortium, http://www.w3.org/XML). It is a metalanguage in the sense that it includes a Document Type Declaration (DTD) that is used to declare a specific language for solving particular problems (XML.com, http://www.xml.com/pub/98/10/guide2.html). A DTD allows one to define various markup languages. It is either contained in a separate file or embedded in the XML document.

As a metalanguage, XML can be used to define a variety of different markup languages (MLs) such as the Synchronized Multimedia Integration Language (SMIL), Personal Information Description Language (PIDL), Business Rules Markup Language (BRML), and many others (World Wide Web Consortium 2, http://www.w3.org/TR).

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 103–110, 2001.

Because artificial intelligence has traditionally required specialized languages in order to solve problems (e.g. ART*Enterprise, Blaze Advisor Rule Engine, JESS, JRules, Lisp, Prolog, TIRS, etc.), the availability of a metalanguage suitable for defining multiple specialized languages for solving problems should allow for considerable application in the AI field. Indeed, there have been some applications of XML to AI problems. Hayes and Cunningham proposed Case Based Markup Language (CBML) (Hayes and Cunningham, 1998), an XML application for data represented as cases. CBML was proposed to facilitate knowledge and data markup that could be readily reusable by intelligent agents. Limitations of the CBML approach are discussed by Hayes and Cunningham (1999) in their work on presenting a *case view*, making a case for techniques that can integrate easily with existing mark-up structures.
Another effort is the Artificial Intelligence Markup Language (AIML) (The XML Cover Pages 2, http://www.oasis-open.org/cover/aiml-ALICE.html). This language is an XML-based language used in ALICE, a chat-bot. This proposed markup language offers a simple yet specialized open-source representation alternative for conversational agents. The language offers a minimalist DTD and leverages the use of specific XML tags like patterns and categories. Still another example is DMML (Kambhatla et.al., 2000) which is a markup language designed for intelligent agent communication and applied to online stock trading.
In both these instances, the idea is that chat bot or other intelligent agent implementations across domains/implementations can share the same generic structure, thus making it easier to program such entities. However, such an approach restricts the composition of an XML message to a specific set of tags and attributes. This compromises on the generic appeal of XML and quickly forms the basis for highly specialized variations of the markup language, leading us to the initial problem, that AI applications require specialized and often awkward representation schemes for data input and output.
In this paper we show that XML of itself is an appropriate tool for building AI applications. The onus is on the application architecture in leveraging the strengths of the XML technology to solve a problem using AI measures. Hayes and Cunningham (1999) demonstrated this in their work on CBR applications that use standard XML documents and generate a usable XML *view* of a company's knowledge system. In our case, we are using XML to build rule-based applications for deployment on the Web.

2 Problem Description

Successful e-business solutions require support for internet standards including HTML and HTTP, integration with web application servers, XML support, a robust communications infrastructure, and scalability for web-based demand (Gold-Bernstein, 1999). Currently, the World Wide Web contains millions of html documents that make a massive repository of data. However, it is difficult for an e-business solution to take advantage of that source because of the general chaos that pervades the Web. There is a need in all businesses for quality customer service, and e-businesses are no exception. To provide quality customer service, intelligence is required, and hence AI (artificial intelligence) must be built into such systems.

XML is an appropriate way for representing the semi-structured data that is present on the Internet. Expressing semantics in XML syntax rather than in first-order logic leads to a simpler evaluation function while needing no agreement on the associated ontologies (Glushko et.al., 1999). XML allows for some structure while being a "meta-language" which permits different structures to be used for different problems and for data to be presented in different forms within a single domain (for example, an XML document is much less structured than a table in a relational database). Thus we see XML as a technology whose influence will increase, and hence to realize the goal of intelligent customer service, a robust interface between XML and rule-based systems must be built. We have built just such an interface and the goal of the remainder of this paper is to describe this effort.

2.1 XML and ART*Enterprise®

Towards exploring the usage of XML for rule-based application development, we have researched one commercially available Rule-Based application development product and developed a prototype underwriting application using XML as the choice representation for input and output data. Our choice of software was made based on the availability of the product and its widespread usage in the mortgage industry.

ART*Enterprise®, a product from MindBox[1] Inc., is an integrated knowledge-based application development environment that supports rule-based, case-based, object-oriented and procedural representation and reasoning of domain knowledge. ART*Enterprise offers cross-platform support for most operating systems, windowing systems, and hardware platforms (Watson, 1997). The product allows seamless integration with industry standard programming languages like C/C++ and offers CORBA and Web features that allow the rules engine to communicate with components written in Java or any other language.

2.2 High-Level Architecture

A typical component-based architecture for e-commerce application is usually composed of three tiers. The thin client layer is represented by the user interface implemented using dynamically generated HTML. The user interface runs within popular web browsers (Netscape Navigator, MS Internet Explorer, etc) embedding a Java virtual machine. The middle tier includes the web server, the application server with a servlet engine, the ART*Enterprise rules engine server and the ART*Enterprise-XML parser. A database back-end forms the final layer in this architecture. Figure 1 depicts this architecture.

The application server uses Enterprise Java Beans (EJB) to seamlessly communicate with the back-end process, the XML APIs and other protocols (such as CORBA IIOP). In our design, the server listens for user http requests and delegates ART*Enterprise/XML requests to the specialized application servlet. The servlet processes the HTML request and passes the results on to the ART*Enterprise-XML

[1] Formerly a part of Inference and Brightware corporations.

parser. This component is an implementation of the Document Object Model (DOM) parser based on the XML 1.0 specification, together with ART*Enterprise engine.

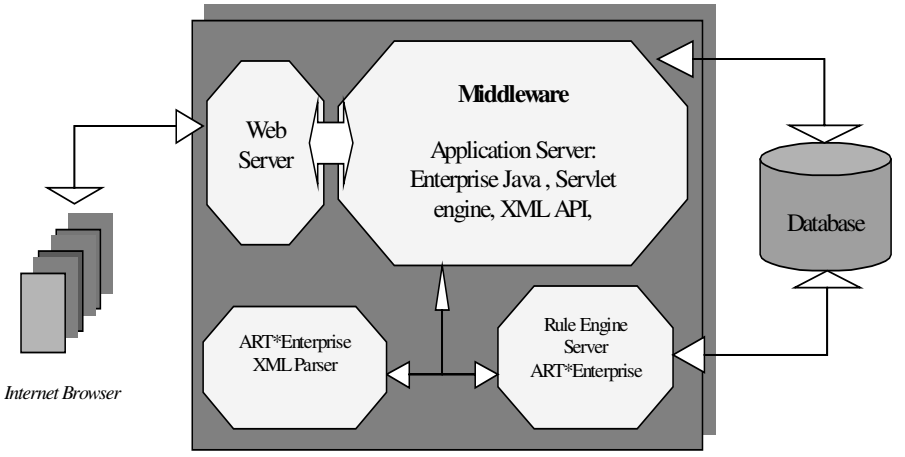


Fig. 1. High Level Architecture

2.3 The XML Parser

There are a variety of XML parsers available free of charge on the Internet. For this application, we elected to build our own parser rather than use one of the ones that are already available. To understand our reason for doing so, we should first explain what we hoped to accomplish with the XML parser. The idea was to take an XML file and convert it into ART*Enterprise objects. We wished, however, to first produce Java objects as an intermediate step towards producing the ART*Enterprise objects. The reason for this was to help ensure compatibility with applications and environments in which Java is a predominant technology. Producing Java objects as a first step would make the XML file available to any Java classes, which might exist. It would also allow preprocessing of the XML files within Java. If, for example, some form of semantic validation on the XML file is needed, this could be done in Java.

Once the Java objects are produced, and validated as needed, the next step is to invoke a method on the top level object to emit the ART*Enterprise code. As a result, we chose to build our own parser in order that the Java objects be structured in such a way as to easily permit building methods to emit ART*Enterprise code. However, if a clear standard XML-to-Java parser were to emerge, which readily permitted adding methods to the generated classes so as to allow the emitting of ART*Enterprise code, that would certainly be a suitable alternative to using our own parser.

The XML-to-Java parser is implemented using JavaCC. This compiler-compiler technology readily permits the building of parsers, which produce Java objects as their output. The compiler produced using JavaCC is itself a Java class and can be

invoked from anywhere within the Java virtual machine, for example from a servlet or a JSP page. The compiler also checks the validity of the XML against the DTD and produces a tree of Java objects. Once the Java objects have been created, a method is invoked on the top-level document object which searches through the tree of Java objects in the document and generates a text file that contains the appropriate ART*Enterprise code. This code is then available to be loaded into the ART*Enterprise application.

3 Example

To demonstrate the feasibility of the proposed architecture, we developed an XML-enabled underwriting application using ART*Enterprise. The underwriting process is simple: loan eligibility is determined based on the front and back ratios. Data input and output are in XML. Figure 2 shows a schematic of the prototype.

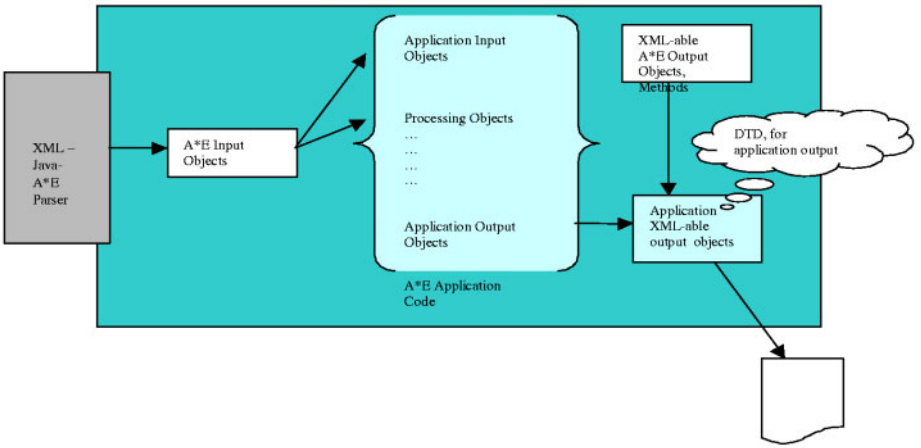


Fig. 2. Application Schematic

3.1 Input

The input form is an HTML user interface accessible using a web browser. The form outlay is simple, with a few key input fields for harvesting user input for a loan application. The form data is submitted to a web browser via the simple yet robust HTTP protocol. The data is then processed by a servlet residing on the web server.
The parser receives the XML document from the servlet, parses the contents and creates ART*Enterprise objects. Listing 1.0 shows input and corresponding application objects.

Example XML code input to the Parser:

```
<INPUT-OBJECT DATA_ID=1001
LOAN_AMOUNT=100000.0
LOAN_TYPE=FRM
```

Example ART*Enterprise code generated by the parser for the XML input above:

```
(define-instance xml:Attribute9-19991215222255155 xml:Attribute
(xml:Has-Name DATA_ID)
(xml:Has-AttValue xml:AttValue11-19991215222255185)
(xml:ownerDocument xml:Document1-19991215222254234)
(xml:value 1001))
(define-instance xml:Attribute12-19991215222255245 xml:Attribute
(xml:Has-Name LOAN_AMOUNT)
(xml:Has-AttValue xml:AttValue14-19991215222255265)
(xml:ownerDocument xml:Document1-19991215222254234)
(xml:value 100000.0))
```

Listing 1. XML code input to the parser

3.2 The Rules Engine

The application loads the data from the parser, made available as schemas in ART*Enterprise. The rules engine first computes the mortgage payment (principal and interest) based on the loan amount, the interest rate and the loan term. The rules engine then determines the eligibility of the case based on two ratios: (i) a ratio of monthly-housing-expenses to monthly income and (ii) a ratio of total-monthly-expenses (housing + other debts/commitments) to monthly income. A simple threshold criteria is applied for determining the eligibility of a loan, the front ratio should be no more than 28% and the back ratio less than 36%. Upon completion of processing, the application would come up with a recommendation and an XML output object data set. This recommendation and the computed ratios are part of the output object data from the application.

The result, an XML document, is now available either for display or for further processing. Listing 2.0 below shows the output from the application.

```
<?xml version="1.0"?>
<OUTPUT_DOCUMENT DATA_ID="1001">
<RECOMMENDATION_SECTION DATA_ID="1001"
RECOMMENDATION="ELIGIBLE">
</RECOMMENDATION_SECTION>

<FILE_ID_SECTION DATA_ID="1001" FILE_TYPE="XML version 1.0"
DATE="Fri Jan 14 13:02:42 2000" AUTHOR="BRIGHTWARE">
</FILE_ID_SECTION>

<BORROWER_SECTION DATA_ID="1001" BORROWER_NAME="Home Buyer One"
BORROWER_SSN="123-45-6789">
</BORROWER_SECTION>

<LOAN_DETAIL_SECTION DATA_ID="1001" LOAN_TERM="30"
LOAN_TYPE="FRM" LOAN_AMOUNT="100000.0" MONTHLY_PI="632.07">
</LOAN_DETAIL_SECTION>
</OUTPUT_DOCUMENT>
```

Listing 2: XML document output from the ART*Enterprise application

4 Conclusion

XML is an appropriate way for representing the semi-structured data that is present on the Internet. One important consideration is the need for standardization of parsers in the AI-XML community. Standardization would facilitate solving problems, rather than selecting, for example, one of a set of parsers and investing considerable resources in a debate over which parser to use.

A similar need for standardization lies in designing specialized markup languages based on XML. Although XML allows for extension by designing other forms of markup, it is not especially desirable if a separate language is developed every time XML is used. However, sizeable vertical industry segments, like the mortgage industry, would benefit from a specialized markup of terms relevant to the mortgage domain.

Usage of XML, however, does not completely eliminate the need for building domain-specific application input layers. In our example, the parser generates ART*Enterprise objects; however, the format of these objects is pretty standard across all domains. If we have specific information about a given domain, then that information can be used to design rules, which are able to map the objects into a format appropriate for the domain. Alternatively, additional methods could be written on the parser's generated Java objects that generate more domain-specific objects. Note that none of this is absolutely necessary: rules can be written using the objects as is, but such an enhancement would make it easier to write rules for a specific domain.

To show the flexibility and the potential of the generated objects, we applied the same approach using the popular Java Expert System Shell (JESS). The mortgage application was rewritten successfully and deployed in the same environment with the same architecture.

5 Future Directions

Organizations implementing e-commerce applications are rapidly adopting XML as their de facto standard for data transfer between applications and partners. Dedicated industry groups like Mortgage Industry Standards Organization (www.mismo.org) have begun standardizing the XML transaction architecture for their clientele. Key players in the technology sector like IBM (IBM, http://www.research.ibm.com/rules/home/html) and Sun (Sun Microsystems, http://jsp.java.sun.com/javaone) are focussing on implementing/integrating rule engines as part of their enterprise e-commerce architecture. These efforts underscore both the need for the XML-rule-based interface described in this paper, as well as the need for a domain specific application input layer.

For knowledge-based systems to be practical and offer effective solutions to the industry, it is imperative that knowledge engineers and the products themselves be extensible and take advantage of the features uniquely brought forward by XML**.** Our experience in applying this approach using both ART*Enterprise and JESS gave us the opportunity to develop a foundation for a shared rule markup language. This work will continue as part of the URML project, an effort to integrate the object oriented rule languages with XML.

References

1. Flynn, P., et.al., University College, Cork, Internet Web Site, http://www.ucc.ie/xml.
2. Glushko, R., Tenenbaum, J., and Meltzer, B., An XML Framework for Agent-Based E-commerce, *Communications of the ACM*, 42:3, March, 1999.
3. Gold-Bernstein, B. 1999. From EAI to e-AI. Application Development Trends, v6 n12.
4. Hayes, C., Cunningham, P., Distributed CBR using XML, *Proceedings of the workshop: Intelligent systems and Electronic Commerce*, Bremen, 1998.
5. Hayes, C., Cunningham, P., Shaping a CBR View with XML, Technical Report TCD-CS-1999-23, Trinity College, Dublin, 1999.
6. Kambhatla, N., Budzikowska, M., Levesque, S., Nicolov, N., Zadrozny, W., Wicha, C., and MacNaught, J., DMML: An XML Language for Interacting with Multi-Modal Dialog Systems, *Proceedings of the Twelfth Conference on Innovative Applications of Artificial Intelligence*, Austin, Texas, August, 2000.
7. Watson, Ian. Applying Case-Based Reasoning: Techniques for Enterprise Systems. Morgan Kaufmann Publishers inc., 1997.
8. IBM, Internet Web Site, http://www.research.ibm.com/rules/home/html.
9. Mortgage Industry Standards Organization, Internet Web Site, http://www.mismo.org
10. Sun Microsystems, Internet Web Site, http://jsp.java.sun.com/javaone
11. World Wide Web Consortium, Internet Web Site, http://www.w3.org/XML
12. World Wide Web Consortium 2, Internet Web Site, http://www.w3.org/TR
13. XML.com, Internet Web Site, http://www.xml.com/pub/98/10/guide2.html.
14. The XML Cover Pages, Internet Web Site, http://www.oasis-open.org/cover.
15. The XML Cover Pages 2, Internet Web Site, http://www.oasis-open.org/cover/AIML-alice.html.

INFOSHOP: A Decision Support Tool for Local Government Regulatory Advice

Ian Watson

Dept. of Computer Science
University of Auckland
Auckland, New Zealand
ian@cs.auckland.ac.nz
www.cs.auckland.ac.nz/~ian/

Abstract. This paper describes the implementation of the INFOSHOP, a unique collaboration between UK central and local government agencies to develop a decision support tool. INFOSHOP helps non-technical front-line reception staff handle complex enquiries on a wide range of government regulations. The paper describes the knowledge engineering and the knowledge level modeling undertaken for the project. The paper describes the design, implementation and architecture of the resulting distributed system that supports local customisation of the knowledge-base in a controlled managed process. The paper concludes by showing that INFOSHOP can be considered a CBR system that uses derivational replay to solve problems rather than the more common retrieval of problem-solution pairs.

1 Introduction

The INFOSHOP project is a one-stop shop, which allows local government front-line staff in the United Kingdom to answer complex queries from the public or businesses on regulations. The INFOSHOP system enables the operator to offer full and consistent advice on a wide range of regulatory issues covering food safety, health and safety, building control and planning regulations.

The INFOSHOP project involves local authority departments working together with central government departments to ensure that accurate information, advice and help, is given to business and the public across a range of services in a cost effective manner. As such it is one of the first examples of the UK Labour government's "*joined up government*" initiatives in action.

INFOSHOP is an intranet application delivered to users through their web-browser. It is based around a set of decision trees provided directly by the central government organisations responsible at a policy level for the regulations about which advice is being given at the local level. The software provides a range of functionality, including fuzzy searching for information, and a suite of enquiry management functions. It is designed so that information content underlying the decision support system can be amended both by central and local government partners, and so that answers to previous enquiries increase the accuracy of future answers.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 111–122, 2001.

2 Background

The Modernising Government White Paper[1], published in March 1999, set out key policies and principles underpinning the UK Government's long-term programme of reform to modernise public services. The programme involves everyone working in public services and everyone who uses them (i.e., the entire nation).

As a response to the White Paper the UK Cabinet Office published an Action Plan setting out some 62 actions being taken to deliver the commitments in the White Paper. INFOSHOP is one such action. In scope, INFOSHOP is a decision support system, designed to be used by front-line staff in Local Government who deal directly with the public. It provides guidance to the user on a range of regulatory matters, based on natural language queries such as: "*Do I need planning permission for a satellite dish*".

There are certain regulatory regimes and issues on which local authorities have to deal with a high volume of enquiries of a comparatively detailed nature. Often enquirers have to be routed to one or more specialists within the regulatory departments involved. Furthermore, enquirers are often only given information relevant to the local authority department contacted and later find out about other regulatory requirements, which are often costly and time consuming because they have been addressed late in the day. This "*pass the enquiry*" process frustrates clients and leads to a number of different contacts being made. Feedback from the People's Panel (a *Service First* initiative of the Government)[2] confirmed this frustration and reports that enquirers want an answer to their query when it is first raised, not to be referred on to other staff.

In 1998 a pilot study was carried out in the London Borough of Bexley. This resulted in a case-based reasoning (CBR) system (implemented using Inference's CBR3 product[3]) [1], which in the area of planning legislation enabled staff to handle 60% of enquiries without referral to a subject expert. Previously the figure had been 30%. It was proposed by a group within the Cabinet Office[4] to build on the success of the Bexley pilot and see whether the same principles underpinning that project could be used in other regulatory areas.

To this end the Cabinet Office bid to the Treasury's *Invest to Save* budget for £100,000 (GBP) to support this work, with approximately matching funds being provided by the Local Authorities involved.

A project group at the Cabinet Office was established along with representatives from:

---

1 http://www.cabinet-office.gov.uk/moderngov/1999/whitepaper/

2 http://www.cabinet-office.gov.uk/servicefirst/index/pphome.htm

3 www.inference.com (now www.egain.com)

4 The Cabinet Office sits at the heart of UK Government, alongside the Prime Minister's Office and the Treasury. It's aim is to ensure that the Government delivers its priorities. It reports directly to the Prime Minister.

- Department for the Environment, Transport & the Regions,
- Department of Health,
- Health & Safety Executive, and
- University of Salford[5],

The following Local Authorities joined the project to pilot the system:

- Barnsley Metropolitan Borough Council
- London Borough of Bexley
- London Borough of Camden
- London Borough of Ealing
- Eden District Council
- Knowsley Metropolitan Borough Council
- Lincolnshire County Council & North Kesteven District Council
- Norfolk County Council
- Reigate & Banstead Borough Council
- Teignbridge District Council
- London Borough of Tower Hamlets
- Thurrock Borough Council
- Vale Royal Borough Council
- London Borough of Waltham Forest

For all involved this was the first time that so many Local Authorities had collaborated on an IT project together, and the first time that several Central Government Departments had worked together with Local Authority partners. It was this collaboration which was the ethos of the Labour Government's "*joined-up government*" initiative.

3 Design Issues

The key design issue was to create a collaborative environment. The purpose of bringing together central government departments and Local Authorities was that a single knowledge-base could be created to cover national regulations. However, this had to be customisable at a local level in two ways.

Firstly, in many instances, and particularly in planning regulation, there are significant local variations in the application of legislation. Central government departments set out policy which local government interprets and applies within their own context. As a consequence regulations differ from one authority to the next. Thus, the centrally provided knowledge base would have to be customisable by the Local Authority in a controlled fashion without the need to employ costly consultant knowledge engineers or programmers.

[5] The author was employed by the University of Salford before moving to the University of Auckland in early 2000 and was a Central Partner of the INFOSHOP project providing advice on knowledge engineering and other technical matters.

Secondly, many enquiries, whilst based on the same legislation will result in different actions in different authorities. For example in one authority a request for planning consent may result in forms being posted to the client, whereas in another details may be taken over the phone, and in a third a case officer may be instructed to visit the property in question. Thus, even where the legislation was being interpreted and applied the same in several authorities the resulting actions may differ. Consequently, actions needed to be locally customisable.

The other main design consideration was one of cost. Local Authorities do not have large IT budgets and have many financial and legal constraints on how revenues can be spent. As a consequence it was essential that INFOSHOP should run on standard PCs and not require expensive user licences.

Linked to this was the budget for the project. £100,000 had been obtained from the Treasury and a contribution of approximately £6,500 was made by each of the Local Authorities giving a maximum budget of nearly £200,000. This sum had to cover all project management expenses, knowledge engineering, implementation, the cost of software licences for the pilot, user training, evaluation and dissemination activities. Given that the money was coming from public funds this budget was fixed and non-negotiable.

4 Implementation

Although the Bexley pilot had used a conversational CBR system [2] the project team thought it wrong to prejudge the technology which might eventually be used to implement the INFOSHOP system. As a consequence the development of the system was split into three distinct phases:

1. *Knowledge acquisition*, which would result in a knowledge level [3] model of the knowledge intended for the system.
2. *Implementation*, which would implement the knowledge in the chosen technology and develop the user interface.
3. *Evaluation*, which would evaluate the pilot system in the field.

Furthermore, in the interests of objectivity it was decided that phases 2 and 3 would not be performed by the same contractor or consultants.

An invitation to tender for phases 1 & 3 (either separately or together) was advertised in early 1999. The consultants PriceWaterhouseCoopers won the bid for both phases 1 and 3.

4.1 Knowledge Acquisition

The project team decided that the INFOSHOP should deal with planning regulations (already partially covered by the Bexley pilot), building control, health and safety and food safety. It was planned that the INFOSHOP should be able to handle the most common 80% of enquiries from the public.

Category: Planning **Sub-category:** Planning permission/consent		**Question No:** P14 **Completed by:** BEXLEY
Client (Who is likely to be asking this question?)		**Underlying requirement** (Why are they asking this question? What are they trying to achieve?)
HOUSEHOLDER - BUSINESS		WANT TO PUT UP SATELLITE DISH FOR PRIVATE OR COMMERCIAL USE
Input(s) (Any information required to answer the question eg reference to local computer systems and databases)		**Source** (Of each input)
LISTED BUILDING CONSERVATION AREAS ARTICLE 4		LOCAL LISTS LOCAL LISTS OR PLANS
Output(s) (Any outputs in addition to simply providing the answer, eg forms supplied, or information passed to other depts/agencies)		**Destination** (Of each output)
LETTERS/FORMS INFO SHEETS – DETR BOOKLET: Householders planning guide for installation of Satellite Television Dishes		ENQUIRER
Applicable legislation (Specify the name of the Regulation/Act. Also, if the answer is derived from an individual regulation (or Act section) specify the regulation number (Act Section)		
TP ACTS GENERAL DEVELOPMENT ORDER		
Keywords/links to related issues (Five keywords/links to categorise/sort this question)		
CONSERVATION AREA SATELLITE DISH/AERIAL LISTED BUILDING		

Fig. 1. A Sample Question Capture Form

PriceWaterhouseCoopers therefore set out to interview each Local Authority to establish what were their most common questions. These were then collated from all the partners to identify the most frequently occurring 80%.

Table 1. No. of Decision Trees per Regulatory Area

Regulatory Area	No. of trees
Planning	51
Building control	35
Food safety	79
Health and Safety	67
Total	232

Knowledge engineering was then undertaken to identify what knowledge was required to answer or handle each question. A form was developed to capture this information (see Figure 1.) and PriceWaterhouseCoopers modelled this information as decision trees, which have been widely used as a concise and readable notation for decision making knowledge [4 & 5]. A total of 232 decision trees were created for the four regulatory areas (note that several question were often subsumed by one decision tree). The knowledge engineering phase took approximately three months.

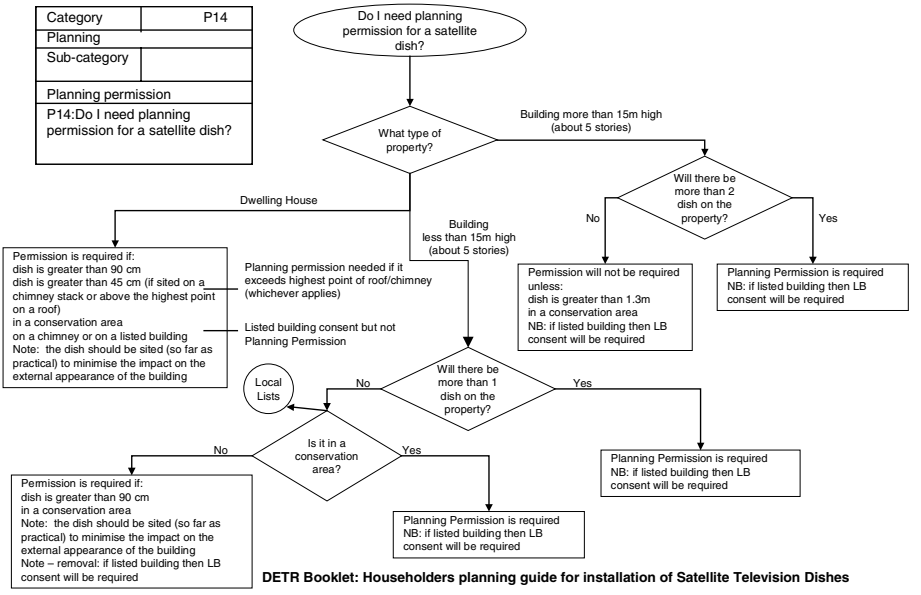


Fig. 2. A Decision Tree from the Planning Domain

4.2 Implementation

An invitation to tender for the implementation of the system was issued in May 1999 and Tagish[6], a company with a strong track record in Local Government IT work, was selected as the contractor. A decision was taken by Tagish to directly implement the decision trees using the flowcharting tool Visio[7]. Visio enables the content of the decision trees (nodes and arcs) to be stored and indexed in a database, which can be searched enabling the correct tree to be retrieved from an initial natural language query. Changes to individual trees can be made using the flowcharting tool, which does not require any programming or knowledge engineering experience. It was hoped that providing proper versioning control was implemented this would satisfy the local customisation requirement for the knowledge base.

Lotus Notes was selected to store the database and decision trees because it provides many features to support collaborative working, versioning, security and through Lotus' Domino Server content is accessible via the Internet or an Intranet. It was recognised that the decision to use Lotus Notes did have a modest licence fee implication for the Local Authorities. However, this was mitigated by the fact that only one Notes licence was needed for the designated tree "author" within each authority plus one Notes Server and Domino Server licence per authority (see Figure 3).

[6] www.tagish.co.uk
[7] www.microsoft.com/office/visio/

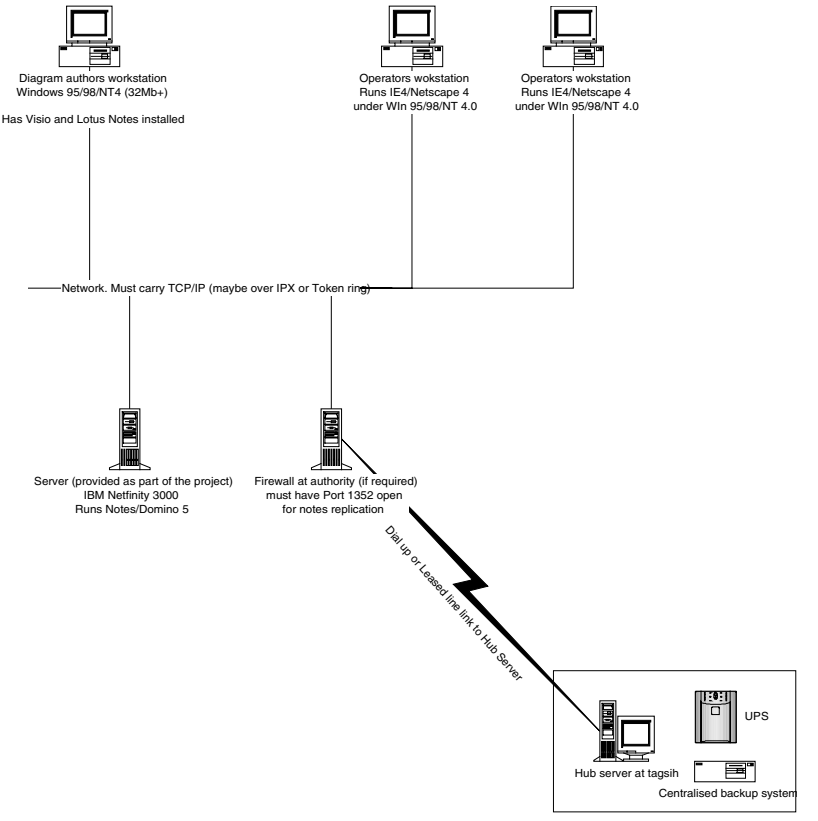


Fig. 3. System Infrastructure

For the pilot Tagish would hold the master copy of the knowledge base (the Master Trees). Each Local Authority would hold a replicated copy on their own server. Local changes to the knowledge-base would be held locally but would also be replicated back to Tagish and stored separately. When national legislation changed the Master Trees would be changed and this change replicated to each Local Authority. The flows of replication and data are illustrated in Figure 4.

A management structure was also put in place for the Local Authorities to periodically meet and review the changes they had independently made to their trees so best practice could be captured and propagated between partners.

4.3 A Consultation

This section takes you through a sample consultation with INFOSHOP [8]. INFOSHOP takes a natural language query and after processing uses it to search the Notes database containing the tree descriptions. Processing the query involves:

[8] A publicly available version of INFOSHOP is available at: http://www.tagish.co.uk/infoshop/online.htm (note some features are disabled)

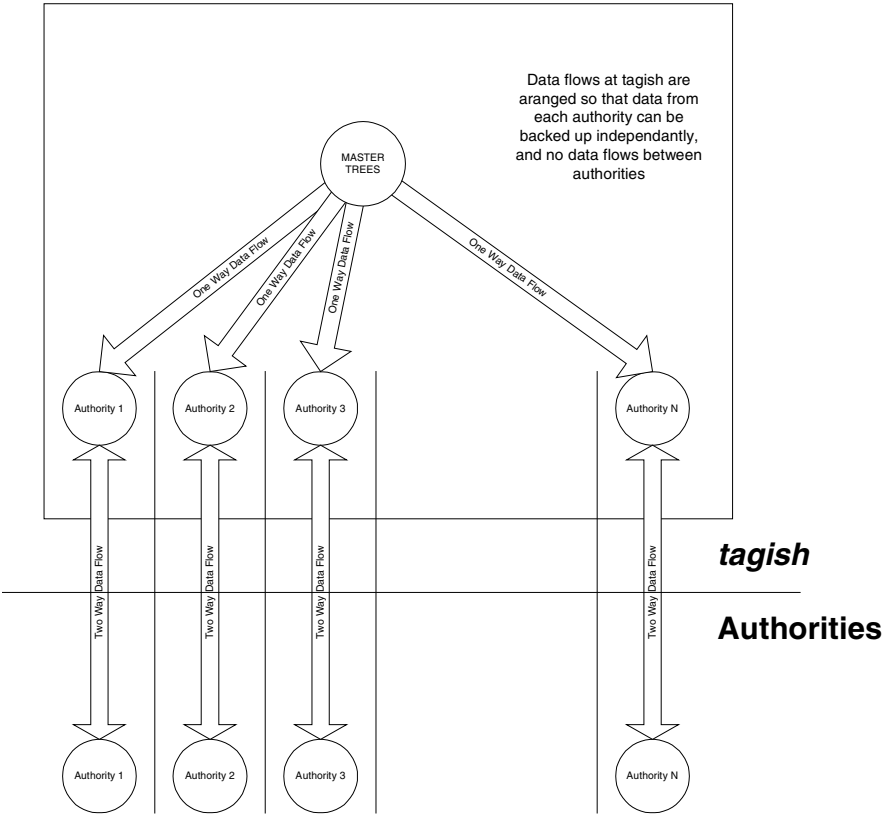


Fig. 4. Replication and Data Flows

- removing noise words (common terms of speech such as prepositions),
- removing plural word forms,
- spell checking to automatically suggest alternatives for unrecognised words
- applying a user defined and locally customisable lexicon to deal with common synonyms and local dialect words specific to one region.

Because the text matching algorithm can not be 100% accurate a set of candidate trees is retrieved with the best match being presented to the user and alternative "relevant inquiry routes" being displayed should the best match prove incorrect (see Figure 5).

Figure 5 shows INFOSHOP's response to the well formed query "*Do I need planning permission for a satellite dish*". INFOSHOP identifies the keywords planning, permission and satellite in the query (these are highlighted in the centre left of the screen). This query causes the retrieval of tree P14. Trees P83, P84, P82 and P80 have also been identified as relevant to this enquiry (these are listed in the bottom right panel of the screen). The retrieval of tree P14 (shown in Figure 2) results in the

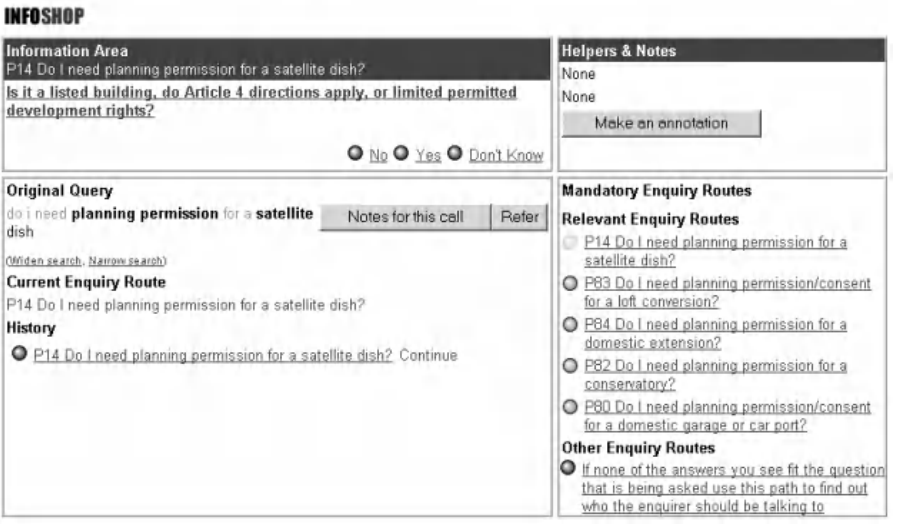


Fig. 5. Screen Dump from the INFOSHOP

question "Is it a listed building…"[9] (shown in the top left panel of the screen – i.e., the region of the screen the user first looks at). If the answer to this question is yes, then Listed Buildings Consent is required regardless of the location of the building the size, position or number of satellite dishes (see Figure 2).

Also of interest in Figure 5. is the "Make an annotation" button. At any time the user can click this and make an annotation. These might be used by to record notes for tree authors to suggest changes or to comment the legislation to improve the ease of future use.

Answering "No" to the question in Figure 5 causes a series of further questions to be asked. Figure 6 shows the conclusion of the consultation. Six more questions have been asked and their answers listed in the "History" panel on the bottom left of the screen.

Arriving at a conclusion (a leaf node in the tree) causes a pop-up window to launch with the result (shown in Figure 7) that no planning permission is required providing it is sited in such a way as to minimise its impact on the external appearance of the building.

Depending on the result required at each node, it can cause documents, forms and standard letters to be retrieved from a document repository system, client data to be entered into booking, logging or tracking systems or faxes and emails to be sent to appropriate people. If a successful solution cannot be obtained the consultation can be logged, appropriate notes can be added to it and it can be referred to an appropriate person to deal with.

[9] Listed buildings in the UK are of historical or architectural significance and are covered by strict regulations.

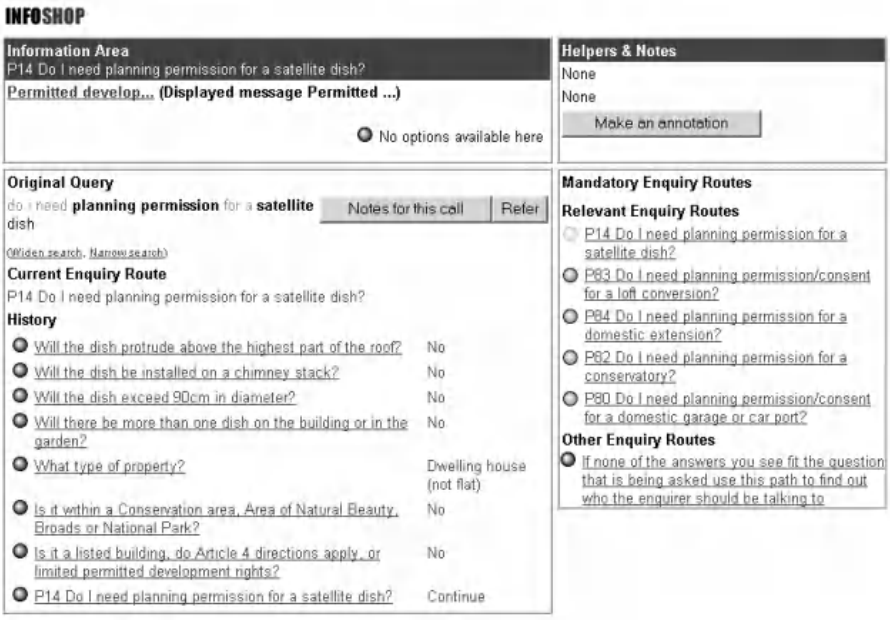


Fig. 6. Screen Dump of a Consultation

5 Conclusion

The final evaluation report from PriceWaterhouseCoopers was completed in June 2000 and results look encouraging. Staff of the Local Authorities are able to author the decision trees as intended, and the usability of the system has been praised by several Local Authorities.

> "*Our reception service is the front line for all services provided by the Environment Department. I am impressed with how user friendly INFOSHOP is. Also, our pilot has made a sound start in beginning to reduce those high frequency standard regulatory calls normally dealt with by Technical Officers. Reception colleagues can now deal with some of these*". [Rudy Bright, Records & Information Manager, London Borough of Camden]

> "*INFOSHOP, fully utilised will be a very powerful tool. Camden decided to go for it, warts and all! I am pleased with the start we have made although we all agree more development work is needed. Camden's INFOSHOP has been adapted to prompt reception colleagues about leaflets and other documents that could also be sent out to the enquirer. Further work will make this into a comprehensive information service."* [Paschal O'Neil, Decision Tree Author, London Borough of Camden]

From an AI perspective INFOSHOP demonstrates that it is not always necessary or appropriate to use an AI tool to develop and deliver AI solutions. The design

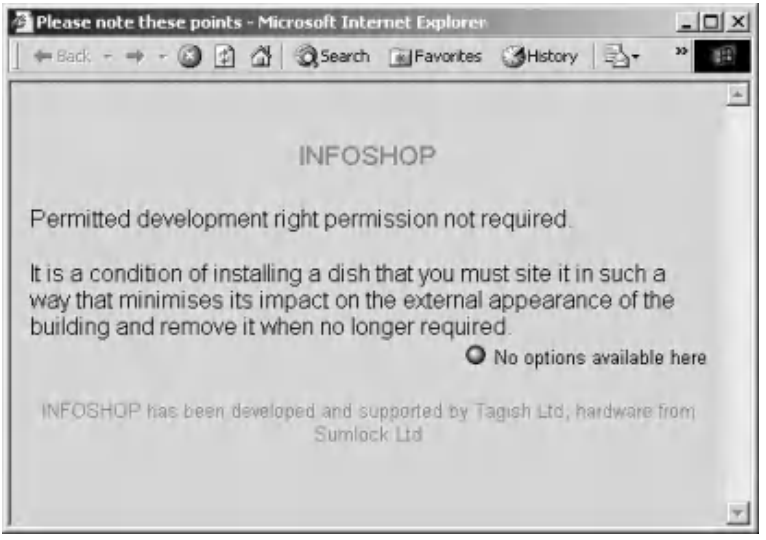


Fig. 7. Result of the Consultation

requirement that Local Authority staff be able to easily customise the knowledge base meant that many more "*sophisticated*" tools were not suitable. However, decision trees as a knowledge level representation [3] were ideal, enabling the authors to worry about documenting legislation and not programming code. Moreover the use of Lotus Note's sophisticated version control, replication and security features made the distributed yet controlled roll out and maintenance of the INFOSHOP feasible.

Although it was decided not to use a CBR tool for the solution, even though it had proved successful in the Bexley pilot, this does not mean that CBR as a problem solving methodology was rejected, quite the contrary. INFOSHOP retrieves the best matching decision tree from its database (i.e., the most similar case) and then uses the retrieved tree to solve the problem.

In CBR terminology this is *derivational replay* [6]. Where cases store problem descriptions and a problem solving method, which can be reused (i.e., replayed) to solve the problem. This is distinct from most CBR systems which store pairs of problem descriptions and their solution. The main advantage of derivational replay is that fewer cases need to be stored since each problem solving method can usually cover a wide range of input criteria. The disadvantage is that you need to understand the domain theory, in order to be able to create problem solving methods.

INFOSHOP also has facilities to acquire new problem solving cases through the customisation and addition of trees (i.e., the revise and retain stages of the CBR-cycle [7]). Thus, INFOSHOP further demonstrates the omnipresence of CBR in problem solving [8] and is a practical demonstration that CBR is a methodology for problem solving not a technology [9], since INFOSHOP doesn't use any of the technologies, such as k-nearest neighbour, so frequently associated with CBR.

The management of INFOSHOP has now passed from the incubator of the Cabinet Office to the Small Business Service[10], an agency of the Department of Trade an Industry that aims to improve the regulatory environment for small businesses, and to ensure that all small businesses have access to world class business support services.

On the 19th April 2000 INFOSHOP won the UK Government Innovation Award 2000[11].

Acknowledgements. The authors would like to acknowledge all the project team at the Cabinet Office: Roger Wiltshire, Angela Evans, Nigel Lockwood and Joe Lloyd, the Local Authority partners (in particular Adrian Cole from Bexley), the Central Government Partners, Nigel Barnes from PricewaterhouseCoopers and all the development team at Tagish.

Disclaimer. This paper does not necessarily represent the views or opinions of any government department, local authority, or the other partners, consultants, contractors or team members of the INFOSHOP project.

References

[1] Watson, I. (1997). Applying Case-Based Reasoning: techniques for enterprise systems. Morgan Kaufmann Inc.

[2] Aha, D.W., Maney, T., & Breslow, L. A. (1998). Supporting dialogue inferencing in conversational case-based reasoning. Fourth European Workshop on Case-Based Reasoning (pp. 262--273). Dublin, Ireland: Springer.

[3] Newell, A. (1982). The knowledge level. Artificial Intelligence, vol. 18, no. 1, Jan. 1982, pp. 87-127.

[4] Longbottom D, Wade G. (1973). An investigation into the application of decision analysis in United Kingdom companies. Omega, vol.1, no.2, April 1973, pp.207-15. UK

[5] Moret B.M.E. (1982). Decision Trees and Diagrams. Computing Surveys, vol.14, no.4, Dec. 1982, pp.593-62.

[6] Mostow, G., & Fisher, G. (1989). Replaying Transformational Derivations of Heuristic Search Algorithms in DIOGENES. In, Proceedings of the DARPA Case-Based Reasoning Workshop, Hammond, K.J. (Ed.), Morgan Kaufmann, Calif., US.

[7] Aamodt, E. & Plaza E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches AICom - Artificial Intelligence Communications, IOS Press, Vol. 7: 1, pp. 39-59.

[8] Aha, D. W. (1998). The Omnipresence of Case-Based Reasoning in Science and Application. Knowledge-Based Systems, 11(5-6), 261-273

[9] Watson, I. (1999). Case-Based Reasoning is a Methodology not a Technology. Knowledge Based Systems Journal Vol. 12 no.5-6, Oct. 1999, pp.303-8. Elsevier, UK.

---

[10] www.dti.gov.uk/sbs/sec1.htm and www.businessadviceonline.org

[11] www.kablenet.com/kable.nsf/KNWhatsNew/1F38BF744C42BE2F802568C60058714E?OpenDocument

Using Boosting to Detect Noisy Data

Virginia Wheway

School of Computer Science and Engineering,
University of New South Wales
Sydney NSW 2052 Australia
virg@cse.unsw.edu.au

Abstract. Noisy data is inherent in many real-life and industrial modelling situations. If prior knowledge of such data was available, it would be a simple process to remove or account for noise and improve model robustness. Unfortunately, in the majority of learning situations, the presence of underlying noise is suspected but difficult to detect.
Ensemble classification techniques such as *bagging*, (Breiman, 1996a), *boosting* (Freund & Schapire, 1997) and *arcing* algorithms (Breiman, 1997) have received much attention in recent literature. Such techniques have been shown to lead to reduced classification error on unseen cases, and this paper demonstrates that they may also be employed as noise detectors. Recently defined diagnostics such as *edge* and *margin* (Breiman, 1997; Freund & Schapire, 1997; Schapire et al., 1998) have been used to explain the improvements made in generalisation error when ensemble classifiers are built. The distributions of these measures are key in the noise detection process introduced in this study.
This paper presents some empirical results on edge distributions which confirm exisiting theories on boosting's tendency to 'balance' error rates. The results are then extended to introduce a methodology whereby boosting may be used to identify noise in training data by examining the changes in edge and margin distributions as boosting proceeds.

1 Introduction

This paper is concerned with the classification problem, whereby a learner is presented with a training set comprising of a series of n labelled training examples of the form $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, with $y_i \in (1, \ldots, k)$. The learner's task is to use these training examples to produce an hypothesis, $h(\mathbf{x})$, which is an estimate of the unknown relationship $y = f(\mathbf{x})$. This 'hypothesis' then allows future prediction of y_i given new input values of $\mathbf{x}$. A classifier built by combining individual $h(\mathbf{x})$'s to form a single classifier is known as an ensemble. Whilst there are many ensemble building methods in existence, this discussion focusses on the method of boosting which is based on a weighted subsampling of the training examples.

Introduced by Freund and Schapire in 1997, boosting is recognised as being one of the most significant recent advances in classification (Freund & Schapire, 1997). Since its introduction, boosting has been the subject of many theoretical and empirical studies (Breiman, 1996b; Quinlan, 1996; Schapire et al., 1998).

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 123–130, 2001.

Empirical studies have shown that ensembles grown from repeatedly applying a learning algorithm over different subsamples of the data result in improved generalisation error.

Boosting is an iterative procedure which trains a learner over the n weighted observations. Boosting begins with all with all training examples being given equal weight (i.e. $\frac{1}{n}$) At the $m+1$-th iteration, examples which were classified incorrectly at the m-th iteration have their weight increased multiplicatively so that the total weight on incorrect observations is equal to 0.5. Hence, the learning algorithm will be given more opportunity to explore areas of the training set which are more difficult to classify. Hypotheses from these parts of the space make fewer mistakes on these areas and play an important role in prediction when all hypotheses are combined via weighted voting. At each iteration, the weighted error is stored and used in the final voting weight when individual classifiers are combined to form the ensemble. Accuracy of the final hypothesis depends on the accuracy of *all* the hypotheses returned at each iteration and the method exploits hypotheses that predict well in more difficult parts of the instance space. An advantage of boosting is that it does not require any background knowledge of the performance of the underlying weak learning algorithm. Refer to Table 1 for details of the boosting algorithm and its weight update methodology.

Table 1. AdaBoost:M1

AdaBoost
Input: n training instances x_i with labels y_i. Maximum trials, M. Base learner, H.
Initialization: All training instances begin with weight $w_i^0 = 1/n$.
Repeat for M trials:

- Induce classifier, h_m, using weighted training data and H.
- ϵ_m = weighted error for h_m on the training data. If $\epsilon_m > 1/2$, discard h_m and stop boosting. (If $\epsilon_m = 0$, then h_m gets infinite weight.)
- Classifier weight, $\beta_m = log\frac{\epsilon_m}{1-\epsilon_m}$
- Re-weight training instances:
  if $h_m(x_i) \neq y_i$ then,
  $w_i^{m+1} = w_i^m/(2\epsilon_m)$
  else, $w_i^{m+1} = w_i^m/2(1-\epsilon_m)$

Unseen instances are classified by voting the ensemble of classifiers h_m with weights w_m.

2 Current Explanations of the Boosting Mechanism

Ensemble classifiers and the reasons for their improved classification accuracy has provided a fertile ground for research in both the machine learning and statistical communities. In theory, as the combined classifier complexity increases,

the gap between training and test set error should increase. However, this is not reflected in empirical studies. There is strong empirical support for the view that overfitting is less of a problem (or perhaps a different problem) when boosting and other resampling methods are used to improve a learner. Some authors have addressed this issue via bias and variance decompostions in an attempt to understand the stability of a learner (Breiman, 1997; Breiman, 1996b; Friedman, 1997).

The following is a summary of the key comments and conclusions made on boosting and ensemble classification to date:

- Breiman (1996b) claims the main effect of the adaptive resampling when building ensembles is to reduce variance, where the reduction comes from adaptive resampling and not the specific form of the ensemble forming algorithm.
- A weighted algorithm in which the classifiers are built via weighted observations performs better than weighted resampling at each iteration, apparently due to removing the randomisation (Friedman, Hastie & Tibshirani, 1998).
- Confidence rated predictions outperform boosting algorithms where a 0/1 loss is applied to incorrect classification (Freund & Schapire, 1996; Schapire & Singer, 1998).
- Successful ensemble classification is due to the non-overlap of errors (Dietterich, 1997) i.e. observations which are classified correctly by one hypothesis are classified incorrectly by others and vice versa.
- Margin and edge analysis are recent explanations (Breiman, 1997; Schapire et al., 1998). More detail on these measures and related studies is provided in the next section.

3 Edge and Margin

Recent explanations as to the success of boosting algorithms have their foundations in margin and edge analysis. These two measures are defined for the ith training observation at trial m as follows:

Assume we have a base learner which produces hypothesis $h_m(\mathbf{x})$ at the m-th iteration, and an error indicator function, $I_m(\mathbf{x}_i) = I(h_m(\mathbf{x}_i) \neq y_i)$. Let c_m represent the vote for the m-th hypothesis with $\sum_m c_m = 1$. Then,

- $edge_i(m, \mathbf{c})$ = total weight assigned to all incorrect classes.
  Breiman (1997) defines the edge as:

$$edge_i(m, \mathbf{c}) = \sum_{j=1}^{m} c_j I_j(\mathbf{x}_i) \tag{1}$$

- $margin_i(m, \mathbf{c})$ = total weight assigned to the correct class minus the maximal weight assigned to any incorrect class.

For the 2 class case $margin_i(m, \mathbf{c}) = 1 - 2\ edge_i(m, \mathbf{c})$ and in general, $margin_i(m, \mathbf{c}) \geq 1 - 2\ edge_i(m, \mathbf{c})$.
Whilst more difficult to compute, the value of the margin is relatively simple to interpret. Margin values will always fall in the range $[-1, 1]$, with high positive margins indicating confidence of correct classification. An example is classified incorrectly if it has a negative margin. The edge on the other hand cannot be used as an indicator variable for correct classification (except in the 2-class case). Whilst the margin is a useful measure due to its interpretability, mathematically it is perhaps not as robust and tractable as the edge.

Schapire et al. (1998) claim that boosting is successful because it creates a higher margin distribution and hence increases the confidence of correct classification. Breiman, however, claims the high margin explanation is incomplete and introduces new ensemble techniques which actively improve margin distributions but do not result in improved generalisation error (Brieman, 1997, 1999).

This study demonstrates the tendency for boosting to 'balance' the edge (or margin) in its quest to classify more difficult observations. Using this property, a method of detecting noise in training data is presented. Methodology for the study is discussed in detail in the next section.

4 Empirical Results

In all experiments, the decision tree learner C4.5 with default values and pruning was used as the base classifier, with a boosted ensemble being built from $M = 50$ iterations. Datasets used are a selection from the UCI[1] Machine Learning Repository. Results from only four of these datasets are presented in this paper. These four datasets were chosen to provide a representative mixture of dataset size and boosting performance previously reported. All edge distribution results are certainly replicable for other UCI datasets. 10-fold crossvalidation was applied whereby the original training data was shuffled randomly and split into 10 equal-sized partitions. Each of the ten partitions was used in turn as a test set for the ensemble generated using the remaining 90% of the original data as a training set.

At each iteration, the values for edge were calculated for each observation in the training set. The average and variance of $edge_i(m, \mathbf{c})$ were calculated as follows:

$$\hat{E}[edge_i(m, \mathbf{c})] = \tfrac{1}{n} \sum_{i=1}^{n} edge_i(m, \mathbf{c})$$

$$\hat{Var}[edge_i(m, \mathbf{c})] = \tfrac{1}{n} \sum_{i=1}^{n} (edge_i(m, \mathbf{c}) - \hat{E}[edge_i(m, \mathbf{c})])^2$$

The graphical results of these trials for the colic, glass and letter datasets appear overleaf.

Refer to Figure 1 to note an apparent exponential decrease in variance perhaps indicating an asymptote of zero or some small value, ϵ. Figure 2 shows

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

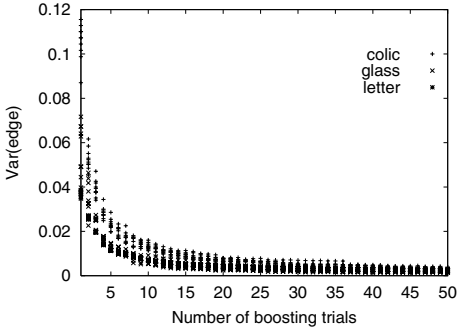


Fig. 1. Var edge vs number of boosting trials

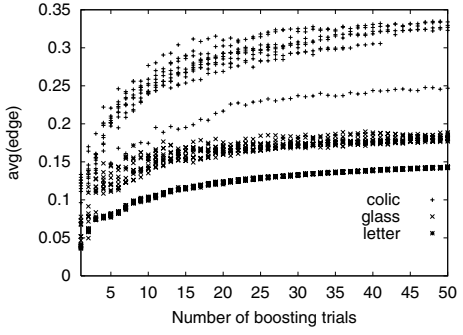


Fig. 2. Average edge vs number of boosting trials

the average edge increasing as the number of boosting trials increases. These results show a homogenisation of the edge, implying that the observation error rate is becoming more uniform. i.e observations which were initially classified correctly are classified incorrectly in later rounds in order to classify 'harder' observations correctly. This results in the percentage of incorrect classifications for each observation becoming more uniform as the number of boosting trials increases. This notion is consistent with margin distribution results presented by Schapire et al. (1998) and edge and margin results by Breiman (1997, 1999).

The most dramatic variance decay is seen in boosting trials $m \leq 5$ i.e. most of the 'hard' work appears to be done in the first few trials. This observation is consistent with several authors noting in earlier published empirical studies that little additional benefit is gained after 10 boosting trials when a relatively strong learner is used as the base learner.

5 A Method for Detecting Noise

The notion of 'balancing' was discussed in the previous section. This property may now be exploited to detect noise in the training data. Since we expect the

distribution of the edge values to become more uniform as the number of boosting trials increases, we may assume deviations from this distribution to be caused by noisy or incorrect data. Noisy data is certainly difficult to classify and some observations may be too difficult to classify correctly in all but a few boosting iterations. Such observations would have edge values which remain high due to persistent misclassification. If these deviations from the overall edge distribution can be detected at say, m=10-20 iterations, then the associated observations may be deemed to be noisy data. Removal of such observations should lead to improved classficiation accuracy on training and test data. The choice of optimal m is still unclear but at m=10-20, computing time is still relatively small and deviations from distributions should already be apparent.

To test this hypothesis, and check whether 'offending' noisy data could be identified, noise was injected into the letter and census datasets by assigning random class labels to 5% of the data. These datasets were chosen because of their size (20,000 and 32,000 observations respectively). To perform this randomisation, the data was shuffled, then the first 5% of observations were assigned a random class label before the data was reshuffled again. This ensured no systematic bias in the noise while still retaining the observation number for later comparison. The edge values were captured after 15 iterations and plotted against observation number below.

Referring to Figure 3 below, a clear distinction between edge values can be seen around observation 1000. For the letter data, observations 0-1000 were deliberately relabelled with random class values to simulate noise. The remainder of the observations were untouched, yet it is still possible that some of these observations may be noisy but as yet undetected in the raw UCI dataset. For the census data, another clear boundary is evident, this time with earlier observations (i.e. observations 0-1500) showing failure to reach lower edge values - note the gap in the plot on the lower left hand corner. Although this example is contrived with known noise being introduced, it demonstrates an important result in being able to identify and eliminate noise. These results indicate that truncating the edge distribution at the top say, 5% of edge values or setting a threshold on edge values and relearning could be an effective way of reducing noise and improving model performance. It would also seem that this technique would be best applied to larger datasets where 5% of observations is not an insignificant number.

Conclusion

This study has presented some interesting results on the variance of the edge when a boosted ensemble is formed. These results confirmed existing research on margin distrubtions. An empirical study on 2 larger datasets demonstrated that is is possible to track deviations in the edge (or margin) distribution when trying to identify the existence or otherwise of noisy data. Although this concept was only tested on known datasets from the UCI repository, initial results are certainly encouraging and further application and research on large scale indus-

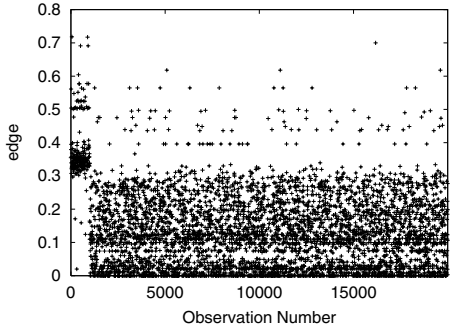


Fig. 3. Edge vs observation number: letter data with class labels randomly assigned for obs. 1-1000

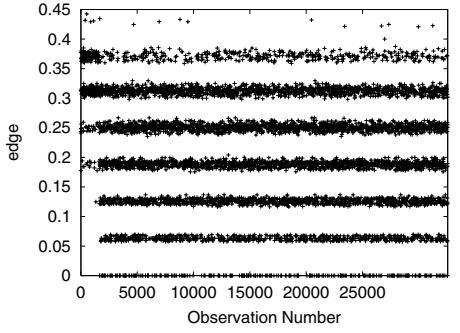


Fig. 4. Edge vs observation number: census data with class labels randomly assigned for obs. 1-1600

trial datasets should lead to higher confidence in the models built from improved training data.

References

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *26(2)*, 123–140.

Breiman, L. (1996b). *Bias, Variance and Arcing Classifiers* (Technical Report 460). Statistics Department, University of California, Berkeley.

Breiman, L. (1997). *Arcing the edge* (Technical Report 486). Statistics Department, University of California, Berkeley.

Breiman, L. (1999). *Random Forests - Random Features* (Technical Report 567). Statistics Department, University of California, Berkeley.

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine*, *18(4)*, 99–137.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148–156). Morgan Kaufmann.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalisation to on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55(1)*, 119–139.

Friedman, J. H. (1997). On bias, variance, 0/1-loss and the curse of dimensionality. *Data Mining and Knowledge Discovery*, *1(1)*, 55–77.

Friedman, J. H. , Hastie, T. & Tibshirani, R.(1998). *Additive logistic regression: a statistical perspective on boosting.* (Technical Report 199). Department of Statistics, Stanford Univeristy.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Quinlan, J. R. (1996). Bagging, boosting and C4.5. *Proceedings of the Thirteenth National Conference on Articifical Intelligence*. (pp. 725–730). Menlo Park California, American Association for Artificial Intelligence.

Schapire, R. E. , Freund, Y., Bartlett, P. & Lee, W. S. (1998). Boosting the margin:a new explanation for the effectiveness of voting methods. *Annals of Statistics*, *26(5)*, 1651–1686.

Schapire, R. E. & Singer, Y. (1998). Improved boosting algorithms using confidence rated predictions. *Proceedings of the Eleventh Computational Learning Theory* (pp.80-91)

Artificial Intelligence in Electronic Commerce

Ryszard Kowalczyk[1] and Maria Lee[2]

CSIRO Mathematical and Information Sciences
[1]723 Swanston Street, Carlton 3053 Vic, Australia
Ph: +61-3-8341 8216, Fax: +61-3-8341 8222
ryszard.kowalczyk@cmis.csiro.au
[2]Locked Bag 17, North Ryde 2113 NSW, Australia
Ph: +61-2-9325 3159, Fax: +61-2- 9325 3200
maria.lee@cmis.csiro.au

Abstract. Following the success of the first Australian Workshop on Artificial Intelligence in Electronic Commerce (AIEC'1999) the second AIEC'2000 workshop was held in conjunction with the 6th Pacific Rim International Conference on Artificial Intelligence (PRICAI'2000) on 29 August 2000. The goal of the AIEC'2000 workshop was to bring together leading practitioners, researchers and developers to explore the challenging technical issues and new intelligent solutions in the rapidly growing area of e-commerce.

1 Introduction

An unprecedented growth of electronic commerce has been driven by a new channel provided by Internet to connect buyers and sellers quickly, efficiently and at a very low cost. The first wave of e-commerce has progressed from web publishing, through basic shopping carts to web application integration with ERP. The second wave focuses on inter-enterprise integration in open marketplaces where businesses can build a more tightly aligned value chain and business relationships that extend across an even broader network of trade and service partners.

At the same time the growing complexity of the e-commerce environment characterized by the increasing velocity, scope and change of information drives the businesses to examine new technologies for more advanced on-line solutions including:

- Customer profiling and need identification
- Product and merchant brokering
- Auctions, tendering and negotiations
- Automated shopping and trading
- Formation of coalitions and virtual enterprises
- Payment and delivery
- Infrastructure, languages, protocols
- Human computer interaction
- Information exchange and communication

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 133–134, 2001.

- Supply chain management
- Security and legal support
- Intelligent assistants, brokers and agents
- Intelligent search and filtering
- Electronic delivery of goods and services

The objective of the AIEC'2000 workshop was to address some of those topics from the perspective of artificial intelligence technology. The one-day workshop consisted of an invited talk from industry, research paper presentations and a panel discussion. The papers selected for this volume represent most of the workshop papers in the original or extended form. They cover a range of topics including intelligent agents, reasoning and ontology for facilitating and automating e-commerce transactions. In particular Wong and Lau propose intelligent payment agents that use possibilistic logic to make payment decisions in the presence of uncertain and incomplete market information. Zhang and Wong present a web-based negotiation agent that applies Case-based Reasoning to capture and re-use previously successful negotiation experiences. Rahwan et al. discuss the use of the BDI software agent paradigm for designing and operations of virtual enterprises. Sinnappan et al. present an agent-based architecture for Internet marketing. Kayed and Colomb present the conceptual graph structures in building tendering ontologies.

2 Program Committee

Acknowledgments. We would like to thank all authors, participants and the program committee members for making the AIEC'2000 workshop successful. In particular we would like to thank Mr Ben Hosken, the Chief Technologist and co-founder of AgentArts Inc for presenting a very stimulating invited talk. Finally, we appreciate the support from the CSIRO Mathematical and Information Sciences, PRICAI'2000 organizers and Springer editors.

Conceptual Structures for Tendering Ontology

Ahmad Kayed and Robert M. Colomb

Computer Science and Electric Engineering,
University of Queensland
Brisbane, Australia
kayed, colomb@csee.uq.edu.au

Abstract. The aim of method presented in this paper is to define the role of conceptual structures in building tendering ontology. More precise, how to use Conceptual Graph to build tendering ontology. We construct our ontologies based on three components: the concepts, the structures, and the contexts. This decomposition facilitates the process of ontology building and reusing. It also helps us to define different types of matching.

1 Introduction

Using Internet as underlying platform for tendering automation involves many problems. Examples of these problems are: security, authentication, heterogeneity, interoperability, and ontology problems. Internet users need tools to search for information across heterogeneous systems and to match potential data sources with their needs. Consumers also need to search for information using terms from domains they are familiar with (Ontologies) [1]. Automating any business process needs to reuse and distribute information among different parties. This raises two problems: the need of a common vocabulary (ontology) and the need of common protocol and management model (standards).

This paper contributes in this direction. We are building online tendering system focusing on helping buyers to write their tender, solving the ontological problems, and giving more potential to e-mediators. We aim at defining the ontological structures needed for tendering process. Using natural language to model tendering makes any process associated with tendering automation extremely difficult. We need to define structures and their contents that will store the tendering information. Since we are interested in storing the information in a knowledge base, the structures should be modeled in logical or formal way. These structures should contain what is called a context [28] [18], which will help in matching and reasoning. The ontology contains abstract concepts that will form the primitives to construct a tender or a bid. This will make it easy to build tools to transfer from a friendly user interface (like the Web) to a logical structure (knowledge base).

To give the flavor of these structures we first give a brief view of our system then we detail these structures. The reader encouraged to read [21] [20] or visit [19] for more details.

Section two identifies the roles of ontological and conceptual structures in automating the tendering process. Section three introduces the ontological conceptual

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 135–146, 2001.

structures for tendering system. Section four demonstrates our solution and section five concludes the paper.

2 The Roles of Ontology in Tendering Automation

The term “Ontology” has its roots in philosophy which has been defined as a particular theory about the nature of being or the kinds of existence [22]. In the computer science community, ontology has become an important issue. Many research areas study and use ontology in various ways, fields such as Artificial Intelligence (AI), knowledge-based systems, language engineering, multi-database systems, agent-based systems, information systems, etc [14]. Guarino [25] defines the ontology as an explicit, partial account of a conceptualization. Gruber [13] defines it as an explicit specification of a conceptualization.

In the tendering process domain, ontology is needed to solve many heterogeneity problems [21]. Buyers/sellers in a specific domain can use a common ontology to describe their needs/offers. The huge amount of information on the Internet prevents buyers and sellers from finding relevant information that meets their needs. If the tendering information is committed to a common ontology, this will make easy to find the relevant information for all parties.

In the tendering domain, different activities may need different types of matching. As an example, in the invitation activity the buyers advertise general information about the tender, while the seller profiles are more specific. Conversely, when a seller is looking for "good" tenders, the tenders are more specific than the sellers' profiles. Different activities need different types of matching. Formal ontologies define the roles, the concepts, and the relations between concepts. This allows us to define many compatibility measures to check the similarity between concepts. This will facilitate different type of matching among buyers and sellers. A formal ontology with different type of relation can solve the different level of abstractions in the call for bid invitation.

The procurement process requires mediation between buyers and sellers. In the e-commerce environment, software agents will play this role. In a community governed by software agents, ontologies become a central issue for the development of any agent-based systems. Ontological-based tendering system will help in testing the feasibility of the ontological approach, which will contribute in building a new generation of business to business EC.

2.1 Conceptual Structures

Conceptual Graphs (CGs) are a method of knowledge representation developed by Sowa [27] based on Charles Peirce's Existential Graphs and semantic networks of artificial intelligence [28]. According to Sowa [27], CGs have a direct mapping to and from natural language and a graphic notation designed for human readability. Conceptual graphs have all the expressive power of logic but more intuitive and readable. CGs are semantically equivalent graphic representation for first order logic (FOL) like Knowledge Interchange Format (KIF).

In our project, we have used CGs as our implementation language. We are aware of some problems of CGs to model ontology. Mineau [23] argues that CG can be used easily to represent an ontology. He argues that with some work in CG it is more suitable than Ontolingua [12]. Ontolingua is a formal ontology description language. Ontolingua consists of a KIF parser, tools for analyzing ontologies, and a set of translators for converting Ontolingua sources into forms acceptable to implemented knowledge representation systems.

The one-to-one mapping between KIF and CG makes it possible to implement ontology in CGs. Ontology can be implanted in CG by using a second order CG. The type definition in CGs can be used to replace the Define-class construct in Ontolingua. The second order Graph, the lambda expression, and the canonical basis are sufficient constructs to implement ontologies in CGs.

Three things influence us to use CGs. First the CG components (the relation and concepts catalogs, the type hierarchy, the canonical basis) are suitable for decomposing the ontology to those components. Second the context (situation where the CG is assertion) is good to represent the lifting axioms and to define different type of matching. Third the one-to-one mapping between CG and KIF (KIF was the original Ontolingua language).

In the following subsection we will introduce the tendering ontologies and show how we used the CGs to build these ontologies.

3 Tendering Ontological Structures

In our project, we divided the ontology into three parts: collections of concepts, collections of conceptual structures, and collections of formal contexts. These all form our ontology (see figure 1). The collections of concepts help us to build tools for translation and integration from one domain to another. The Concepts part consist of three sub-parts. Those are: the catalog vocabularies, the relation vocabularies, and the hierarchical relation between concepts (the type function in CG). The Conceptual Structures (CS) represents the basic element for the tendering system. Software agents use these CSs to communicate and interact. Buyers, sellers, and mediator used these structures to describe their need, offers, responses, or queries (the canonical basis in CGs). The formal context will provide the mechanisms of defining the similarities between concepts. The formal contexts contain three parts: the intentions graph (the graph in which the graph will be asserted), the lifting axioms, and the relation (type-of, is-a, part-of, etc.). The lifting axioms help us in reusing the ontologies and knowledge.

Our approach of decomposing the ontological constructions into three parts helps us building the tendering ontologies. Building ontology is still an art, not a science [8], despite some attempts to define a methodology for ontology construction [11] [9] [29].

Decomposition of the ontology facilitates the implementation of large-scale knowledge bases. It maintains efficiency for high-performance knowledge representation systems. [17] divide their ontology into structural assertion (e.g. Is-a)

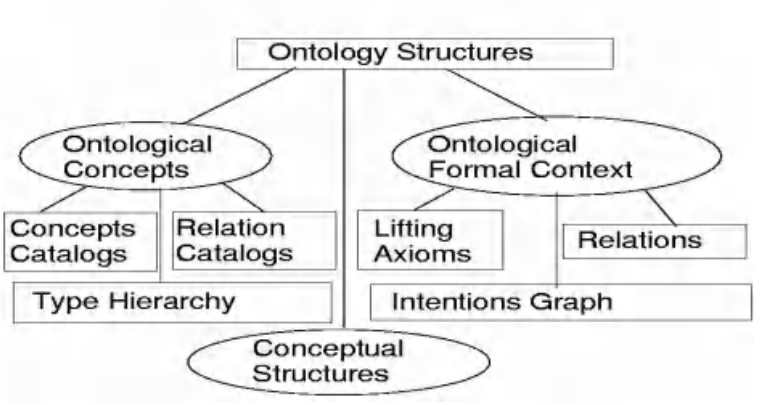


Fig. 1. Ontology Structure

and non-structural assertion (all other concepts in the KB). They claim that the number of structural assertions is less that the number of non- structural assertions. Depending on that they kept the structural assertions in a cache memory where the non-structural assertions are kept in secondary memory.

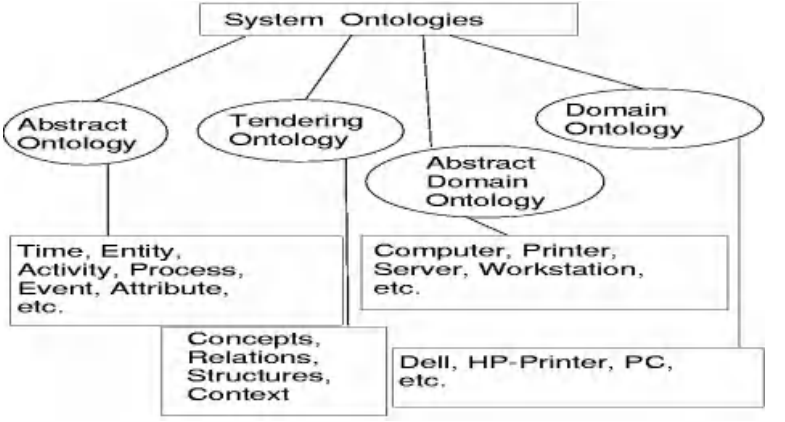


Fig. 2. System Ontologies

Following our framework [21], we need four types of ontologies: meta-ontology, abstract domain ontology, domain ontology, and tendering ontology (see figure 2). In the following we will describe each one.

The Meta-Ontology defines (describes) very general concepts for other ontologies. The meta-Ontology helps to query the domain ontologies and to translate from and to the domain ontologies. This is a very abstract ontology and we build its components from other generic ontologies like [7], [3], and [30]. We reuse the definition of time (Date, Days, Years, Hours) from the ontology server [7]. We take the basic unit

measures from Cyc ontology server [3]. Cyc is common-sense knowledge base ontology used to relate concepts from different models. We also redefine some organizational concepts like Entity, Buyer, Seller, Agent, Activity, Process, etc. from the Enterprise ontology [30].

The Abstract Domain Ontology contains *Classes* which are abstract description of objects in a domain. The class has Class-ID, Class-Properties, Class-Synonyms, Class-Type, Relation, Sub-Class, and Axioms. A relation is a link between classes, and axioms are rules that govern the behavior of the classes. The abstract domain ontology represents a container of abstract data types for sellers' catalogs. In this sense we should distinguish between the catalogs and the ontology. Ontology may contain a PC as a concept, which has RAM and CPU as other concepts. Catalog may contain Pentium 3 with 32 MB RAM. In CGs sense, this can be translated to

[PC:Dell]→(Part-Of)→[CPU-Type: Pentium 3]→(part-Of)→[Memory: RAM] ← (measure)←[Memory-Unit: 32 MB]

The Domain Ontology is a collection of vocabularies mapped to concepts in the Abstract Domain Ontology. Since the Abstract Domain Ontology (ADO) is a schema for the sellers' catalogs, we should define mapping between these abstract concepts (in the ADO) and the catalog values. This is how we know that Dell computer is a PC concept. We can keep this ontology in the mediator side or each seller can create it locally. Normally, this ontology is huge and constructed from the catalogs. If there are some values are not mapped to the abstract domain ontology, the mediator may add new concepts, which are relevant to that value. Mediator also can provide [unknown] concept, which mapped these values, then later add new concepts for these values.

The Tendering Ontology represents the core ontology in our system. The basic part of it is the Tendering Conceptual Structures (TCSs). We divide them into three models: buyer, seller, and mediator models. The buyer model is divided into advertising model, query model, and policy model. The advertising model again can be divided into tender invitation, terms, objects (services), specification, and returned forms. It is beyond the scope of this paper to describe everything in this ontology. Figure 3 shows the hierarchy of this ontology. Appendix 1 summarizes some examples of the catalogs concepts and relations. For the sake of space, we illustrate our tendering ontologies by detailing two of the most important TCSs. Those are the Tendering Invitation Structure (TIS) and the sellers' profile structure (SPS).

Tendering Invitation Structure. The tender invitation structure (TIS) is to inform the tenderers of the scope of the procurement. The tender invitation provides basic information on the procurement and guidance to the tenderers on the participation. We derived the component of TIS from UN EDIFACT ISO 9735 request for quote message [2].

We define TIS as a nested CG, containing information about the scope of the procurement, the address and conditions of contracting entity, nature of contract, duration/completion of contract, eligibility, award criteria, rules on participation, and objects specifications.

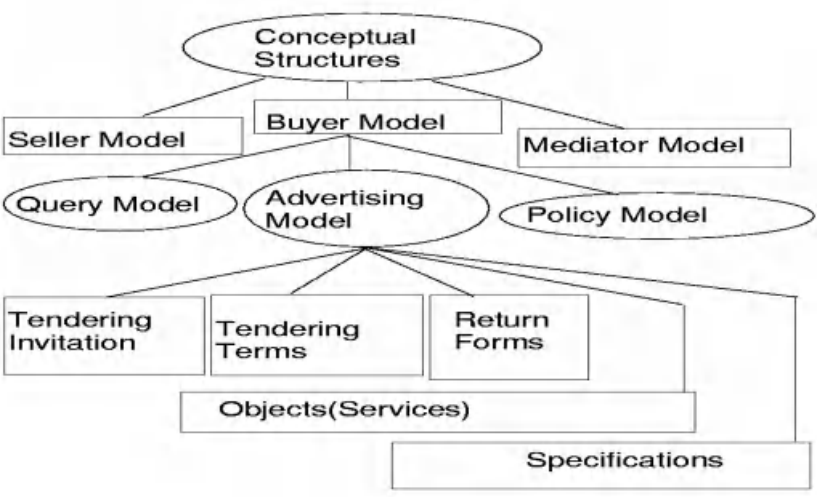


Fig. 3. Tendering Conceptual Structures

Formally TIS is defined using the type function of CG as follows:
Type TIS(x) is
[TCS:*x] -
(ATTR) → [Contracting Entity]-
(ATTR)→[Address]
(ATTR)→[Name]
(ATTR)→[Contracting Duration]
(ATTR)→[Eligibility]
(ATTR)→[Award Criteria]
(ATTR)→[Participation Rules]
(ATTR)→[Services])-
(ATTR)← [Identification]
(ATTR)← [Description]
(ATTR)← [Quantity]
(Measure)← [Unit]
(Relevant Date)← [Production Date]
(Relevant Date)← [Expiry date]
... etc.

Sellers register with a certain mediator by sending their profiles. We define the SPS as a nested CG, containing information about sellers, their domain of interests, the services they offer, the value-added services such as extended warranties, brand reputation, fast delivery times, etc. Formally SPS is defined in CG as follows:

Type SPS(x) is a
[TCS:*x]-
(ATTR) → [Contracting Entity]-
(ATTR) → [Address]

(ATTR) → [Name]

(ATTR) → [Domain Interests]

(ATTR) → [Services]

(ATTR) → [Value Added Services: *{extended

warranties, brand reputation, fast delivery times }]

In this example the concepts are not normalized i.e.(concept can be divided into more primitives concept like the address can be divided into country, street, etc.). The Tendering Conceptual Structure (TCS) is a primitive concept defined in the concepts catalog. In this case the TCS is a super type of TIS (TIS < TCS). The type definition in C.G declares a new concept from primitive and predefined concepts and relations. Type definition contains two parts: the genus (the body of the definition) and differentiae (the label associated with that definition) [27]. Each concept in the body of a type definition should be defined in the concept catalog or in previous type definitions; also each relation should be defined in the same manner. To illustrate this, the contracting duration (CD) has been defined by the type definition as follows:

Type CD(x) is

[Time:*x]-

(Part-Of) ← [Tendering Date]

(Part-Of) ← [Start Date]

(Part-Of) ← [End Date]

4 Motivating Example

E-mediator receives users' structures (buyers or sellers) and checks their semantics and correctness. To check that, E-mediator checks if the structures are canonically derived from the ontologies. Here we apply the algorithm of [24]. This algorithm decides whether a conceptual graph is canonical relative to a given canonical basis. The complexity of this algorithm is polynomial related to the complexity of computing a projection between two conceptual graphs. When the canonical basis is a set of trees, it is polynomial. We use the editor proposed in [26] to edit the canonical basis graphs.

Buyer agent contacts the mediator through formal structures that are committed to the ontology which should be defined in previous step. Mediator checks the profile repository, and depending on the buyers' strategies determines the address of sellers that match their needs. When mediator receives the user's structure (e.g. TIS), mediator tries to find a projection between the ontological structure and the user structure.

To illustrate this, suppose Mr. K. wants to buy 600 PC. A seller wants to add his profile to the mediator repository. Mr. K. wants to advertise his need with a certain mediator. He consults mediator ontology; he browses the structure ontology and finds that TIS suits his needs. He fills the TIS as follows

[TCS: B1] -

(ATTR)→ [Contracting Entity]-

(ATTR)→ [Name: Mr. K.]

(ATTR)→ [Contracting Duration]-

(Part-Of)← [Start Date:31-10-2000]
(Part-Of)← Lambda(T)[End Date:31-12-2000]
(ATTR)→ [Services])-
(Part-Of) ← [Service1]-
(ATTR) ← [Identification: PC] ←(CTX) ← [I_4]
(ATTR) ← [Quantity:600]
(ATTR) ← [Description]-
(ATTR) ← [Warranty]←(> =)← [Year:@3]
(Do)→ [installation]
(Do)→ [Delivery]← (Before)← Delivery Time] ← (< =) ← [[T]+ [Month: @3]]
(ATTR)→ [General Terms]← (Part-Of)← [Experience]← (>)← [Year:@3]
The seller S1 can submit his profile as follows
[TCS: S1]-
(ATTR) → [Contracting Entity]-
(ATTR) → [Name: Seller1]
(ATTR) → [Domain Interests: {[Electronic], [Selling], [Installation]}]
(ATTR) → [Services: {Computers, Printers }]
(ATTR) → [Value Added Services: {[Extended warranties: @4 years] , [Delivery: @[Day: @60]]}]

All terms in square brackets are defined in concepts catalog, previous type or prototype definition. We assume that these structures are semantically correct. After that e-mediator tries to match the TIS with a suitable profile. By applying the formulation rules, we can find out that the buyer TIS matches the seller profile. The matching here is in the sense of generalization and specialization. That means there is a projection from the buyers TIS onto the seller SPS. Think of e-mediator as a theorem prover who tries to prove the buyers' TIS with sellers' PSPs repository. Mediator may use some lifting axioms to prove any assertion in the TIS graph. As example, the assertion [Day: @60] < [Month: @3] can be easily proved by using the lifting axiom [Month: 1]→(=)→ [Day: 30]. For more technical details about CG matching and indexing, readers encouraged to read Ellis work in [5] [4] [6].

When buyers submit their TIS they should define contexts, which help the mediator, to find the appropriate sellers. In the CG sense the context is a situation where the graph is true. In our system we define the context as concepts measures labeled from I_1 to I_{12} (see table 1, some measures were adopted from [15]). We associate the context with a sub-graph via the CXT relationship. In the previous example, under I_4 (Kind-of) the buyer accepts any seller who sells computer since the [PC] is a kind-of [Computer]. In other situation (i.e. under I_1 (Identical)) this is not true.

We apply the context measures in graph level or in the concept level. In the concept level, the concept is true if and only if there is a relation in the ontology catalogs relates the two concepts. This relation must be equivalent or dominated the context relation (i.e. I_1 to I_{12}). In the graph level, the context is true if and only if the context is true for each concept.

Table 1: Intention CG Contexts

Similarity Type	Meaning
(I_1) -Identical	Two concepts are the same
(I_2) -Equal	Two concepts are equivalent
(I_3) -Compatible	Two concepts are transformable
(I_4) -Kind-Of	Specialization of a concept
(I_5) -Association	Positive association between two concepts
(I_6) -Collection-Of	Collection of related concepts
(I_7) -Component-Object	Component part of a concept
(I_8) -Portion	A mass portion of a concept
(I_9) -Instance-Of	Instance of a concept
(I_{10})-Common	Common characteristics of a collection or concept
(I_{11})-Feature	Descriptive feature of a concept
(I_{12})-Has	Property belonging to instances or concepts

5 Conclusions and Future Work

We have defined the role of ontology in automating the tendering process. We have constructed our ontologies based on three components: the concepts, the structures, and the contexts. This decomposition facilitates the process of ontology building and reusing. We have described our system of tendering automation focusing on the role of ontology. We clarified how the ontology would help in defining semantic matching. We have shown how the expressive power of CG helps in building ontologies and conceptual structures.

We introduced the concepts of layered ontologies. At the top level we used very abstract ontology which contains abstract data types for the domain ontology. Defining variant levels of abstractions facilitates the transform of catalog to ontology. One of the exciting areas here is to define the relation between the catalogs, standards, and ontologies. Catalogs are not interoperable. Standard catalogs are but lack flexibility. The ontology is more flexible and provides interoperability between partners.

The lack of agreement of the definition of ontology in each domain, where each application organizes it to suit themselves, causes a problem in building a common ontology. Our approach in decomposing the ontology into abstract and domain ontology helps to solve this problem.

In future work, we will use the context to implement what we call soft-matching [21]. Soft-matching depends on the multi-attribute utility theory (MAUT) [16]. The tender is divided into classes, which may contain sub-classes. The buyer provides a factor of importance (utility function) for each concept in each class in each level in the tender (the sum in each level should =100%). A context like I_1 is not totally true, it is true with the importance factor specified by the buyer. So the context will have the

form [I_i, Percentage]. This fuzziness will capture the buyers' policies that will direct the agent in finding buyers' needs.

Since we are thinking of agent oriented design, the mediator represents a collection of software agents. Agency means that the agent can behave on behalf of the user, but this behavior is controlled by a given strategies or policies. In our future work, we will use the CG formalism to define how the agent could work autonomously, how the policies can be adaptable, whether the agent can change its own policy by learning, etc.

The CG literatures provide some tools to check if a CG is canonically derived from a canon basis or not (e.g. Mugnier et. al algorithm [24] and their CoGITaNT project [10]). The reverse engineering of this process is not supported. At first glance, the idea of given CGs then create canon basis seems absurd. The idea of reverse engineering of CSs to canon basis makes sense in the ontology building process. If we think of ontology as enhanced step in knowledge representation, then the ontology will represent abstract knowledge that can be reused. In this case the reverse engineering of knowledge structure will be the basis for the ontology building process. An algorithm to build a minimal canon basis from given CGs is needed. We are building tools to reverse engineering of CSs to canon basis.

Acknowledgment. The authors acknowledge the department of CSEE at the University of Queensland for financial support for this project. Also we acknowledge anonymous referees who contributed to improving the ideas and readability of this paper.

References

[1] Nabil Adam, Oktay Dogramaci, Aryya Gangopadhyay, and Yelena Yesha. *Electronic Commerce: Technical, Business, and Legal Issues*. Prentice Hall (ISBN: 0-13-949082-5, August 1998.

[2] Peder Blomberg and S"ren Lennartsson. *Technical assistance in Electronic Tendering Devlopment-FINAL REPORT Technical assistance in electronic procurement to EDI - EEG 12 Sub-group 1*. http://simaptest.infeurope.lu/EN/pub/src/main6.htm, June 1997.

[3] C Elkan and R. Greiner. Book review of building large knowledge-based systems: Representation and inference in the cyc project (D.B. Lenat and R.V. Guha). *Artificial Intelligence*, 61(1):41-52, May 1993.

[4] G. Ellis. Efficient retrieval from hierarchies of objects using lattice operations. *Lecture Notes in Computer Science*, 699:274-284, 1993.

[5] Gerard Ellis. *Ph.D. Thesis: Managing Complex Objects*. Computer Science Department, University of Queensland, 1995.

[6] Gerard Ellis, Robert A. Levinson, and Peter J. Robinson. Managing complex objects in peirce. *International Journal of Human-Computer Studies*, 41(1,2):109-148, 1994.

[7] Adam Farquhar, Richard Fikes, and James Rice. The ontolingua server: A tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707-727, 1997.

[8] M. Fernandez, A. Gomez-Perez, and N. Juristo. *Methodology: From Ontological Art Towards Ontological Engineering*. Workshop on Ontological Engineering. ECAI'96 PP 41-51, 1996.

[9] M. Fernandez, A. Gomez-Perez, and J. Sierra. Building a chemical ontology using methodology and the ontology design environment. *IEEE Intelligent Systems*, 14,1:37-46, 1999.

[10] D. Genest and E. Salvat. A platform allowing typed nested graphs: How CoGITo became CoGITaNT. *Lecture Notes in Computer Science*, 1453:154-164, 1998.

[11] A. Gomez-Perez and D. Rojas-Amaya. Ontological reengineering for reuse. *Lecture Notes in Computer Science*, 1621:139-149, 1999.

[12] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199-220, 1993.

[13] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5,6):907-928, 1995.

[14] N. Guarino. Formal ontology and information systems. In *Proc. of the 1st International Conference, Trento, Italy*, 6-8 June 1998.

[15] Joachim Hammer and Dennis McLeod. An approach to resolving semantic heterogeneity in a federation of autonomous, heterogeneous database systems. *Journal for Intelligent and Cooperative Information Systems*, 2(1):51-83, 1993.

[16] Z. Hatush. *Ph.D. Thesis: Contractor selection using multi-attribute utility theory*. School of Construction Management and Property, Queensland University of Technology-Australia, 1996.

[17] James Hendler and Kilian Stoffel. Back-end technology for high-perfoemance knowledge-representation systems. *IEEE Intelligent Systems*, 14,3:63-69, 1999.

[18] Vipul Kashyap and Amit P. Sheth. Semantic and schematic similarities between database objects: A context-based approach. *The VLDB Journal*, 5(4):276-304, December 1996.

[19] Ahmad Kayed. *Home Page*. http://www.csee.uq.edu.au/ ~ kayed/, 2000.

[20] Ahmad Kayed. *Building Online Tendering System*. Doctoral Consortium,2000 Pacific Asia Conference on Information Systems (PACIS2000), Hong Kong, June 2000.

[21] Ahmad Kayed and Robert M. Colomb. Infrastructure for electronic tendering interoperability. In *The Australian Workshop on AI in Electronic Commerce, conjunction with the Australian Joint Conference on Artificial Intelligence (AI'99) Sydney, Australia, ISBN 0643065520*, pages 87-102, Dec. 1999.

[22] Incorporated Merriam-Webster. *WWWebster Dictionary*. http://www.m-w.com/dictionary, 1999.

[23] G. W. Mineau. The term definition operators of ontolingua and of the conceptual graph formalism: a comparison. In John F. Mineau, Guy W.; Moulin, Bernard; Sowa, editor, *Proceedings of the First International Conference on Conceptual Structures for Knowledge Representation (ICCS '93)*, volume 699 of *LNAI*, pages 90-105, Quebec City, Canada, August 1993. Springer Verlag.

[24] M. L. Mugnier and M. Chein. Characterization and algorithmic recognition of canonical conceptual graphs. *Lecture Notes in Computer Science*, 699:294-304, 1993.

[25] Guarino N. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. *In M. T. Pazienza (ed.) Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, Springer Verlag*, To appear.

[26] S. Pollitt, A. Burrow, and P. W. Eklund. WebKB-GE - A visual editor for canonical conceptual graphs. *Lecture Notes in Computer Science*, 1453:111-118, 1998.

[27] J. F. Sowa. *Conceptual Structures: Information Processing in Minds and Machines*. Addison-Wesley, Reading, Mass., 1984.

[28] J. F. Sowa. Syntax, semantics, and pragmatics of contexts. *Lecture Notes in Computer Science*, 954:1-39, 1995.

[29] Mike Uschold. Building ontologies: Towards a unified methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK, 1996.
[30] Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13 (Special Issue on Putting Ontologies to Use), 1998.

Appendix 1. Entity, Contracting Entity, Buyers, Sellers, Process, Activity, Name, Address, Street, Service, Items, Contracting entity Classification, Classical, Government entity, Name, Address, Organisation, Organisation code, Postal code, Place, City, Country, Official registration number, Registration number, VAT number, Contact information, Contact person, Telephone, Telefax, Electronic mail, Nature of contract, Purchase, Rent, Lease, Hire, Duration, completion of contract, Contract date, Contract start date, Contract end date, Number of months, Weeks, Days, Eligibility, Lowest price, Most economically advantageous, tender Rules, Final date for receipt of tenders, etc.

Virtual Enterprise Design – BDI Agents vs. Objects

Iyad Rahwan[1], Ryszard Kowalczyk[2], and Yun Yang[1]

[1] School of Information Technology, Swinburne University of Technology
P.O.Box 218, Hawthorn, VIC 3122, Australia
{iyad, yun}@it.swin.edu.au
[2] CSIRO Mathematical and Information Sciences
723 Swanston St, Carlton, VIC 3054, Australia
Ryszard.Kowalczyk@cmis.csiro.au

Abstract. Current research identifying architectures for a virtual enterprise has moved from information modelling to role modelling. Thus, a high level of autonomy results from the distribution of responsibilities, capabilities, and knowledge among different business units in the virtual enterprise at the design stage. Current trends tend towards using object-oriented technology as an effective abstract system design and implementation methodology. We argue that applying the software agent paradigm to the virtual enterprise provides various advantages on both the design and operational levels. We further show that the Belief Desire Intention agent architecture has additional abilities of mapping real world business unit autonomy and interaction. We also introduce the Belief Desire Intention agent paradigm capability of facilitating highly flexible (agile) enterprise design and implementation.

1 Introduction

A significant enhancement of inter-enterprise transactions has been provided by the emerging information technologies (IT). This has raised a need for establishing a new paradigm defining architectures, standards, and design and implementation policies for inter-enterprise systems, which realize the capabilities of IT and maximize their utilization in all possible ways. The Virtual Enterprise (VE) paradigm emerged in order to address this need and produce efficient procedures for dealing with the above requirements. Although there is no clear standardized definition of the VE, there is a general agreement between currently proposed definitions. For example, the NIIIP project [1] defines the VE to be "a temporary consortium or alliance of companies formed to share costs and skills and exploit fast-changing market opportunities". While for Walton and Whicker [2] "the Virtual Enterprise consists of a series of co-operating 'nodes' of core competence, which form into a supply chain in order to address a specific opportunity in the market place". In other words, the virtual enterprise is an organization of enterprises sharing resources and working cooperatively towards the achievement of mutual benefits. Terms such as e-commerce, supply chain management, and virtual corporation correspond to closely related concepts. In a VE, there is no centralized control, neither is there a hierarchy of enterprise management levels. Instead, the cooperation of independent, self-interested units results in convergence towards an overall welfare.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 147–157, 2001.

In the next section, we outline the problem with current virtual enterprise design, and hence implementation approaches. Then, in section 3, we present an overview of agent technology and different agent architectures. Section 4 discusses the problem with current approaches towards designing the virtual enterprise, particularly the object-oriented approach. Section 5 outlines the major advantages of the agent approach. In Section 6, we introduce the specific capabilities of Belief Desire Intention (BDI) agents in modelling and implementing virtual enterprises. Next, we show how the BDI approach corresponds well to a set of well-known agile (flexible) enterprise design guidelines in section 7. Finally, a number of conclusions are drawn, and future research is outlined.

2 Virtual Enterprise Problems

There is a great diversity in the way different organizations do business. Different companies have different priorities, business process definitions, ontologies representing business documents and procedures, and different tools for modelling their overall strategies and plans. Aspects such as the explicit representation of coordination strategies (by adoption of workflow technologies), the execution of the coordination strategies (workflow engine), and cooperative system organization (negotiation, coalition formation) represent major issues to be resolved [3]. The behaviour of each company regarding its participation in a VE can be explicitly configured and stated through a *plan* and other general *profile characteristics* [3]. The Workflow Process Definition Language following the workflow reference architecture proposed by the Workflow Management Coalition [4] is one approach to represent dynamic behaviour. It is based on explicit representation of the workflow of business processes as well as all possible exceptions and the way each one of them should be handled. Another method to represent the dynamic behaviour of each VE node is the use of Petri Nets [3].

The problem with using such approaches down to the detailed level is that an explicit representation covering all possible cases introduces a scalability problem. This makes exception handling module design an extremely complex task, especially within the VE context. In this paper, we propose the incorporation of agent technology in order to create dynamic VE systems while reducing the complexity of the configuration process. Instead of explicitly specifying how each situation must be handled at the elementary design stage, an overall process design is implemented, and responsibilities are given to different autonomous units which are capable of solving their own internal problems. The different units are then provided with a coordination mechanism, allowing for the dynamic nature of the system to emerge during the system implementation, deployment and operation phases. We do not assume static, pre-negotiated intra- and inter-organizational workflow. In contrast, as proposed in [5], we view the establishment of a VE as a problem of dynamically expanding and integrating workflows in decentralized, autonomous and interacting workflow systems. Workflow techniques are best suited for implementing an abstract business process model which describes the overall process steps that have to be performed to achieve a specific business goal according to well defined business rules and the respective responsibilities of process participants [6]. The various process steps can

then be realized by *intelligent* business components (implemented by intelligent software agents) that perform specific business transactions.

3 Software Agents for Modelling

Agent-based computing has been considered 'the new revolution in software' [7]. The following definition is adapted from a definition proposed by [8]: "an agent is a software system (or system component) that is *situated* in an environment, which it can *perceive*, and that is capable of *autonomous* action in this environment in order to meet its design objectives."

Jennings and Wooldridge [9] proposed that an *intelligent* agent is an agent that is capable of *flexible* autonomous behaviour, where flexible means:

- ***Responsive***: able to perceive the environment and respond in a timely fashion.
- ***Proactive***: exhibit goal-directed behaviour and take initiative when appropriate.
- ***Social***: able to interact/communicate when appropriate with humans and other agents.

From this point and on, we will use the term agent to refer to intelligent agent.

One of the interesting and most important aspects of agents is that they facilitate *cognitive* modelling (based on behavioural aspects fulfilling the purpose), as opposed to *role* modelling (based solely on purpose). If we are able to define a framework that best describes an agent, and how it interacts with its environment and with other agents, we will have achieved a significant contribution towards the abstraction of system design, and gain better mapping of real world problem solving into our computer systems. Moreover, the autonomy and pro-activity of agents facilitate a system in which entities take action without the need for centralized control. More advantages of using the agent approach are discussed in section 3.

A number of proposals have been made describing different internal architectures of agents and their implications on the performance of agent systems.

- **Reactive Agents.** In a reactive agent, the behaviour modules are finite state machines. Behaviours are implemented as rules of the form: situation → action [8].
- **Deliberative Agents.** A deliberative agent has explicit goals. It reasons about which, when, and how to achieve them [8]. A deliberative agent has an internal state, allowing flexible, history-sensitive, non-deterministic behaviour.
- **Hybrid Agents.** Hybrid agents combine the best of reactive and deliberative features. BDI Agents, discussed next, are one example of Hybrid Agents.

The Belief Desire Intention (BDI) model combines reactive and deliberative approaches in a uniform way. BDI agents have internal *mental* states [10]. A mental state is comprised of three concepts:

- **Beliefs**: The information the agent currently believes true. This information could be about the agent internal state or about the environment. We should emphasize the difference between *knowledge* and *belief*. While knowledge is assumed to be always *true*, beliefs are considered to be *true* but possibly *false*. This captures the dynamic nature of the agent's information about itself and the environment.
- **Desires (goals)**: Desired future states.
- **Intentions**: Commitments to action. The notion of intentions is closely related, but not identical, to the notion of plans. In a certain situation an agent might have a

number of possible plans. The selected plan, and the commitment to taking action become an intention. In other words, an intention is an instance of a plan.

The control cycle of BDI agents is described in *Figure 1*. First an event or goal is selected for processing. The agent then finds plans that are applicable to the current situation. Appropriate plans are chosen, resulting in the creation of intentions. The agent then executes the enabled intention, starting with the first step. This may result in an action being executed, or an event being posted, hence invoking sub-plans, etc.

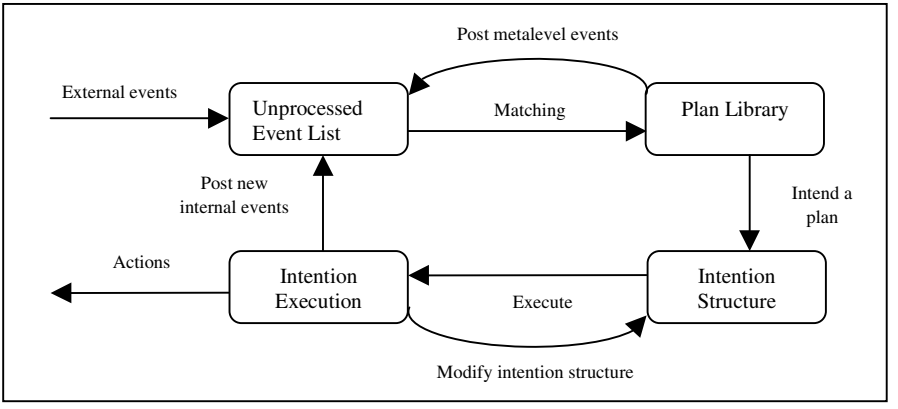


Fig. 1. BDI Control Cycle

Due to its highly abstract philosophical origins, the BDI approach has proved valuable for the design of agents that operate in a dynamic environment, and that respond flexibly to changing circumstances despite incomplete information about the state of the world and other agents in it [11]. The BDI reasoning model resembles the kind of reasoning that we appear to use in our everyday lives [8]. Another major attraction of this model is that it provides a clear functional decomposition, similar to, but more uniform than that provided by hybrid layered architectures.

There is an ongoing argument around the best way to represent and generate plans in different agent models. For example, some argue that rather than explicitly providing a library of plans and choosing from it, a rule-based agent has an implicit representation of different plans, and plans emerge at run time through following those rules. The BDI cycle "provides for a spectrum of behaviours ranging from purely deliberative, to highly reactive, depending on the structure of the plans provided and on decisions such as how often to take account of changes in the environment while executing plans" [11]. In other words, the BDI model provides an effective, highly flexible paradigm for developing intelligent software agents that can be configured on various levels of intelligence and deliberation to accommodate the nature of the problem.

Since this paper is in the context of facilitating VE, there is an inherent necessity for having multiple agents in the systems. Agents, in this case, will not only be able to reason autonomously and effectively about achieving local objectives, but they will also be able to gain the benefits of interacting with other agents in the VE. *Interaction protocols* are "recipes" for structured communication between agents [12]. Ultimately, an interaction protocol consists of: 1) a set of messages encoded in an

Agent Communication Language *ACL*; and 2) a set of rules governing conversations (sequences of messages).

4 Approaching the Virtual Enterprise Problem

Enabling the VE requires the adoption of a new technical infrastructure, as well as an effective design methodology. As systems scale up, and the complexity increases, there is a need for a design methodology, which is highly abstract yet, has the ability to be directly implemented. The most dominant methodologies currently used adapt the object-oriented (OO) system design principles. These methodologies provide a higher level of abstraction of the problem. In the object-oriented paradigm, the system is comprised of several, possibly distributed, objects interacting with each other through predefined interfaces. The interface describes the object's services which can be invoked by other objects. In other words, an object is a *passive* system component which is able to respond to predefined requests and react accordingly. An object system is usually governed by a central process/object responsible for the coordination between objects and the flow of control between them towards the achievement of the desired result.

The first drawback of objects is related to the level of abstraction they provide. Business *objects* make a major contribution to modelling *information* in the enterprise. While business *agents* extend this capability to model *behaviour* in the enterprise. This fact means that the behaviour of agents can be modified *dynamically*, due to learning or influence of other agents or the environment. Moreover, agents can dynamically cooperate to solve problems.

Another important difference between the object and agent paradigms is related to the level of autonomy. The object paradigm uses the notion of *encapsulation* to give objects control over their own internal states. An object can declare that a certain instance variable (or method) is private (only internally accessible) or public (accessible both internally and by other objects). While an object may exhibit autonomy over its *state*, by making instance variables accessible only via certain methods (controlled channels); it does not exhibit autonomy over its *behaviour* [8]. That is because once a method has been declared public (at design time), the object has no control over whether that method is executed. Any other object could *invoke* this method if it wants to. In a multi-agent system, an agent *requests* an action to be performed by another agent rather than *invoking* this action. And this request could be accepted or refused by the serving agent (i.e. the decision lies in the hands of the serving agent rather than the client. Decision could depend on various factors such as the current domain state or the identity of the client). Objects are considered to provide good mapping of real-world problem solving. But that limitation affects this mapping capability dramatically, since the actual problem description needs to be conceptually modified to accommodate the limited capabilities objects offer.

The major reason for the apparent failure of object orientation to deliver on reuse is insufficient attention to the issue of domain understanding and the representation of this understanding in an unambiguous and precise way [13]. The OO paradigm is intended towards the implementation of software systems rather than rich business concept representation (such as rules, constraints, goals and responsibilities). Goal seeking behaviour, policies and trust are all enterprise concepts which do not naturally

fit into a computational object model. Based on this argument, A. Wood et al [14] showed that conventional object modelling constructs are not sufficient for modelling enterprises.

As mentioned in the previous section, the passiveness of objects requires the availability of "controller objects" which are responsible for the coordination of control among other objects. As the size of an OO system grows, "the number of messages between objects grows non-linearly and controller objects themselves need to be coordinated or they can become performance bottlenecks" [13].

It is important to denote that many of the advantages of agents mentioned above could indeed be implemented using object tools. But our argument is based on the fact that those attributes are not components of the basic object-oriented model.

5 The General Agent Approach

Following the definition of the VE mentioned earlier, the VE creation could be viewed as a Cooperative System design problem. A Cooperative System is "a system in which a set of autonomous agents (computational and human) interact with each other through sharing their information, decision making capabilities and other resources, and distributing the corresponding workload among themselves, in order to achieve common or complementary goals" [3]. There is an apparent similarity between the definition of the VE and that of a Cooperative Agent System from two perspectives: the problem addressed, and the approach adopted towards solving it.

Both object and agent paradigms address *change* and *complexity* but to different levels. We are not proposing the disposal of the widely used distributed object foundation. Instead, agent technology can be built on top of the distributed object infrastructure to provide a natural merging of object orientation and knowledge-based technologies [13]. Agents' ability to provide reasoning capability within the primitive application component logic facilitates direct mapping and encapsulation of business rules within the organization.

The motivations behind the agent solution could be summarized by the following points adapted from [15]:

1. The need for a higher-level design approach, which is capable of mapping effectively to real world problem solving capabilities as opposed to approaches where the actual problem description might need to be conceptually modified to accommodate the limited capabilities of the approach.
2. The domain involves an inherent distribution of data, problem solving capabilities, and responsibilities.
3. The need to maintain the autonomy of organizational business units and sub-components.
4. The need for accommodating complex interactions, such as negotiation, coordination, cooperation, and information sharing. This calls for more sophisticated social skills provided by agents.
5. The fact that a detailed solution to a problem cannot be designed from start to finish, as the business environment is highly unpredictable and a rigid one-time design does not accommodate such changes. Instead, a general workflow definition of business processes could be implemented, while autonomous, adaptive components deal with specific business transactions and their internal problems.

6 Advantages of the BDI Agent Approach

In addition to the advantages of using the agent paradigm mentioned in the previous section, there are BDI Agent-specific features that are central to our argument. We will show that each element of the BDI model offers an advantage in the scope of capturing significant aspects of virtual enterprises.

Beliefs: Going back to the object-oriented model, there is an overall agreement concerning the benefits of the encapsulation of information within enterprise objects. This advantage is still realized by the BDI Agent model through the agent's beliefs (its *current* knowledge about itself, its environment, and other agents). Furthermore, the BDI architecture offers a higher level of abstraction by explicitly allowing beliefs to have a direct impact upon the agent's behaviour. This allows for the agent's behaviour to be dynamically modified as its knowledge about the domain changes. For example, one of the enterprise concepts that do not naturally fit into a computational object model is the specification of *policies* that govern the behaviour of enterprise objects [14]. An agent's set of beliefs, on the other hand, could include specifications of these policies, as well as having the ability to accommodate *changes* in such policies as part of its nature.

Desires: There is a rising need for capturing the goal directed behaviour in enterprise business units. For example, if the coordinating process in an enterprise were based on precise specifications of sub-process activities, sub-processes would not be allowed to dynamically change their activities, as this will cause coordination failure. Capture of intentional information concerning higher level business objectives of processes and the mapping of specific activities to those objectives allows dynamically changing systems to maintain coordination across a useful range of situations [16]. This way, activities and strategies that constitute the business unit task are allowed to autonomously evolve as required by changes in their local domain.

Intentions: Intentions reflect the reasoning ability which an agent pursues before it takes decisions about its actions (i.e. about what plan to commit next). The ability to choose from different possible plans maps well to a business unit's different strategies for achieving its task. If one plan fails, another could be tried until no other plans are available. This way, no error reporting to the higher-level coordination process or service requestor is required until all possible strategies are consumed without success.

7 Agile Enterprise Design

In the previous sections, we showed that the BDI agent architecture has interesting capabilities of modelling the behaviour of business units. In this section, we will take a further step into showing how the BDI agent paradigm may provide powerful facilitation of highly adaptable (*agile*) enterprise design. We will use a set of agile enterprise system design principles in order to show how the BDI agent paradigm could support such an enterprise through attributes central to its description.

An agile enterprise is one that is broadly change-proficient [17]. In other words, an agile enterprise manages and applies knowledge in order to accommodate and cause

change to its own advantage. Regardless of the strategies chosen, effective implementations of such an enterprise employ a common set of principles that promote proficiency at change. Designing agile systems, be they entire enterprises or any of their critical elements like business practices, operating procedures, supply-chain strategies, and production processes, require designing a sustainable proficiency at change into the very nature of the system [18].

R. Dove [18] identified ten key design principles which are responsible for the high adaptability in a number of industrial applications. These principles have emerged from observations of both natural and man-made systems. *Table 1* shows these principles and the corresponding BDI agent architecture features which facilitate the design and application of each principle.

Table 1. Correspondence between the BDI agent model and agile enterprise design principles

Design Principles	Corresponding BDI Agent Model Features
Self Contained Units: System of separable self-sufficient units not intimately integrated. Internal workings not important externally.	An agent is an autonomous, self-sufficient unit capable of performing a task proactively and independently. It can interact with other agents without central control. An agent encapsulates capability implementations resulting in service abstraction.
Plug Compatibility: System units share common interaction and interface standards, and are easily inserted or removed.	Agents use an *interaction protocol* implemented using an agent communication language (ACL) such as KQML to enforce message syntax. Message semantics could be realized by ad-hoc or industry standards such as EDIFACT.
Facilitated Re-use: Unit inventory management, modification tools, and designated maintenance responsibilities.	Agents are modular, and belong to classes from which any number of agents could be instantiated. Modification could be done by either changing the agent's state (beliefs) or by replacing the agent by another with more sophisticated model to support extra or more efficient functionality.
Non-Hierarchical Interaction: Non-hierarchical direct negotiation, communication, and interfacing among system units.	Agents interact with each other without centralized control through direct messaging and negotiation. Bidding for internal jobs could be done among groups of agents, providing granularity of interactions. Because BDI agents introduce the notion of planning, sophisticated negotiation and collaboration techniques could be incorporated within the business unit itself.
Deferred Commitment: Relationships are transient when possible; fixed binding is postponed until immediately necessary.	Individual business unit agents are assigned job fulfilment in real time rather than pre-specifying a complete detailed workflow of system processes. This could be done through interaction with other agents as imposed by the situation. Since there is no heavily centralized control, new agents can be easily added to the system to facilitate unsupported functionalities, allowing the system to grow dynamically. Another level of deferred commitment is present in the fact that agents perform online planning according to situations.

Distributed Control and Information: Units respond to objectives; decisions made at point of knowledge; data retained locally but accessible globally.	Each agent has a private representation of its own objectives (desires), which are directed towards the overall system performance. This enables agents to *decide*, locally and in real time, what to do next. Information distribution advantages are similar to those proposed by object models (encapsulation), but the beliefs notion of BDI agents allows for more flexible knowledge representation (for example, allowing for true and false beliefs to be included).
Self-Organizing Relationships: Dynamic unit alliances and scheduling; open bidding; and other self-adapting behaviours.	Since a BDI agent has its own cognitive model, modifications in its beliefs can cause change in behaviour. Plan generation, and hence decision-making, are dependant on the agents own dynamic model of the environment. Automated coalition formation [19] allows both static and dynamic formation of BDI teams. Coalition formation could be done in many different ways to enable the achievement of mutual goals or the exchange of benefits.
Flexible Capacity: Unrestricted unit populations that allow large increases and decreases in total unit population.	An instantiation of any number of agents is possible as needed. Agents could be added to perform new business functionalities or represent business units. Agents could reside on different machines or be mobile in order to achieve scalability.
Unit Redundancy: Duplicate unit types or capabilities to provide capacity fluctuation options and fault tolerance.	Agent systems allow easy recovery. If after consuming all possible strategies, an agent fails to achieve its task, it could report this to another agent which is capable of dealing with such situation by either finding another way of performing the task or choosing an alternative task.
Evolving Standards: Evolving, open system framework, capable of accommodating legacy, common, and completely new units.	Agents could be designed so that they interact with legacy systems by using technologies such as those used to integrate object systems (eg. CORBA). Moreover, it is possible to upgrade an agent to a version with more functionality, enabling the system to evolve.

8 Conclusions and Further Research

Automating the virtual enterprise is the next step beyond today's e-commerce. The most effective way of designing and implementing the virtual enterprise is that which offers capabilities for direct mapping to the behavioural nature of various business units. This is not currently fulfilled by existing paradigms, such as the object-oriented paradigm, which impose a need for mutating the problem in order to fit into the limited design capabilities the paradigm offers. Software agents can offer a significant advantage to the design and implementation of flexible, adaptive, and scalable virtual enterprises. Furthermore, Belief Desire Intention agent architectures can naturally accommodate a rich representation of various business units' knowledge, goals, and strategic plans. They can facilitate a highly adaptive (agile) enterprise design through attributes that are central to this particular agent architecture.

This paper is a step towards incorporating agent technologies into electronic commerce and virtual enterprises. There is a need for further investigation of agents' capability to offer additional features from the functional point of view, such as negotiation and dynamic planning capabilities. Among different proposed planning, negotiation and collaboration models, effective choices must be made which effectively model real world business practices. More work also needs to be done towards methodologies for the design and analysis of agent systems.

Acknowledgement. Special thanks to David Kinny for his valuable contributions made in various discussions surrounding software agents and to referees for their comments.

References

1. The NIIIP Reference Architecture, 1996, www. niiip.org.
2. Walton, J., Whicker, L.: Virtual Enterprise: Myth & Reality. J. Control (1996).
3. Camarinha-Matos, L.M., Afsarmanesh, H.: Cooperative Systems Challenges in Virtual Enterprises. Proceedings of the 2nd IMACS International Multiconference on Computational Engineering in Systems Applications (in CD), CESA'98, Nabeul-Hammamet, Tunisia, April (1998).
4. Workflow Management Coalition: Workflow Management Coalition, The Workflow Reference Model - Document Number TC00 - 1003, Issue 1.1, Brussels, Nov (1994).
5. Chrysanthis, P.K., Znati, T., Banerjee, S., Chang, S.K.: Establishing Virtual Enterprises by means of Mobile Agents. RIDE99, Sydney, Australia, (1999) 116-123.
6. Schmidt, M.T.: Building Workflow Business Objects. Proceedings of the OOPSLA 98. Workshop on Business Object Component Design and Implementation. Vancouver, (1998).
7. Ovum Report. Intelligent agents: the new revolution in software (1994).
8. Wooldridge, M.: Intelligent Agents. In: Grehard Weiss (ed.): Multiagent Systems. MIT Press (1999).
9. Jennings, N.R., Wooldridge, M.: Applications of Intelligent Agents. In: Jennings, N.R., Wooldridge (eds.): Agent Technology: Foundations, Applications, and Markets (1998) 3-28.
10. Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), San Fransisco, USA, June, (1995).
11. Dignum, F., Morley, D., Sonenberg, E.A., Cavedon, L.: Towards socially sophisticated BDI agents. (To appear) In Proceedings of the Fourth International Conference on MultiAgent Systems, Boston, USA (ICMAS 2000).
12. d'Inverno, M., Kinny, D., Luck, M.: Interaction Protocols in Boa. Proceedings of ICMAS (1998).
13. Farhoodi, F., Fingar, P.: *Competing for the Future with Intelligent Agents*. Distributed Object Computing, "DOC" Magazine, Part 1: Oct 1997, Part 2: Nov (1997).
14. Wood, A., Milosevic, Z., Aagedal, J.O.: Describing Virtual Enterprises: the Object of Roles and the Role of Objects. Proceedings of the OOPSLA 98. Workshop on Objects, Components and the Virtual Enterprise. Vancouver (1998).
15. Jennings, N.R., Norman, T.J., Faratin, P.: ADEPT: An Agent-Based Approach to Business Process Management. SIGMOND Record 27(4), (1998) 32-39.

16. Burg, B.K.B.: Using Intentional Information to Coordinate Interoperating Workflows. Proceedings of the OOPSLA 98. Workshop on Business Object Component Design and Implementation. Vancouver, (1998).
17. Dove, R., Hartman, S., Benson, S.: An Agile Enterprise Reference Model with a Case Study of Remmele Engineering. Agiligy Forum, December (1996), Report AR96-04.
18. Dove, R.: Design Principles for Highly Adaptable Business Systems, With Tangible Manufacturing Examples. Maryland's Industrial Handbook. McGraw Hill, (1999).
19. Sandholm, T., Lesser, V.: Coalition Formation Among Bounded Rational Agents. 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Canada, (1995) pp. 662-669.

Agent Based Architecture for Internet Marketing

Sukunesan Sinnappan, Mary-Anne Williams, and Siva Muthaly

Centre of Electronic Commerce for Global Business
University of Newcastle, 2308 NSW
mgssi@cc.newcastle.edu.au
maryanne@ebusiness.newcastle.edu.au
smuthaly@mail.newcastle.edu.au

Abstract. In the current highly competitive era of e-commerce, many firms are focusing their attention on forging and nurturing customer relationships. Businesses not only need to discover and capitalize actionable marketing intelligence but they also need to manipulate this valuable information to improve their relationship with their customer base. In this paper we describe an agent architecture paired with web mining techniques that provides personalized web sessions to nurture customer relationship by:

- *Merging information from heterogeneous sources.*
- *Dealing with change in a dynamic environment.*
- *Handling decisions that need to be made with information that is incomplete and/or uncertain.*

Our agents are based on the BDI[1] framework implemented using JACK[2]. and Java Server Pages. We envisage that our intelligent software agents inhabit a market space where they work for the benefit of businesses engaged in customer-to-business and business-to-business electronic commerce. The key contribution and focus of this paper is the development of an agent framework that ensures successful customer relationship management.

Keywords: Personalized web sessions, Customer Relationship Management, BDI Agent, Web Mining, Server logs

1 Introduction

In the current increasingly efficient business environment, customers have instant access to information and a wider array of product choice, a situation leading to intensive comparison shopping [6]. E-commerce facilitates conditions for *perfect competition* and forces a knowledge shift from retailers to customers [23]. This massive power shift occurs as customers gain advantage through rapid access to quality information [6,17]. Since sustaining competitive advantage in this highly competitive era of e-commerce is an uphill task, many firms are focusing their attention on forging and nurturing relationships with their customers via intelligent systems. In a survey done by Ernst and Young it was found that the spending for customer relationship management (CRM) has increased by 33 per cent whilst overall technology spending remained at 8 per cent. The most effective retailers have always

[1] Based on practical reasoning adopted method by human. BDI agents are characterized by a mental state with three components: beliefs, desires, and intentions.

[2] JACK Intelligent Agent is a third-generation agent framework, designed as a set of lightweight component with high performance and strong data typing.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 158–169, 2001.

been those who achieved a deep understanding of their customers and were in a position to fulfill and anticipate their needs [6]. Nevertheless, many e-commerce businesses have made attempts to increase the scale and efficiency of their CRM by employing mass customization techniques but failed due to the shallowness of strategy and technology employed.

With the development of smart websites retailers are now able to track customer-browsing activity down to a mouse click and learn their customer access patterns by using server logs. Realistically, it is impossible to know exactly what a visitor's intentions are. But the very fact that he chooses to view certain pages portrays his interest in the attributes that the particular page carries. Designing an ideal website is difficult. First, different visitors carry different intention and are attracted to the different elements at the website. Second, different visitors perceive the same site differently and may use it differently against the designer's expectations. Third, the same visitor may log in to the same site with a changed flavor with new intentions.

The challenge lies in the identification of relevant information that is actually needed at that time, for the customer and to be put forth in the most appealing and informative form [29]. This highlights the need for modeling and categorization of Web objects (pages and etc.) and user, matching objects and/or subjects (across and between), and finally, the determination of a set of actions to be recommended for personalization [16]. Many AI solutions have come about in response to this challenge. All too often these approaches are based on human interaction-input (user profile) such as Firefly [25]. These static inputs feed the system to produce the same non-dynamic personalized presentation, as the profile ages. We believe that by agent technology based on the BDI model paired with web mining techniques will yield more favorable results.

A number of approaches have been developed which focus on automatic personalization based on user-web site interaction. For instance Schechter et al [24] came up with the technique using user path profiles to predict further web usage, which can be used for network and proxy caching. Other researchers, Perkowitz and Etzioni [20] suggested the idea of optimizing on the Web site based co-occurrence page patterns. Clustering of user sessions to predict future user behavior were used by Shahabi et al [26] and Nasraoui et al [18]. Mobasher, Cooley and Srivastava [16] provided several techniques in which the user preferences are automatically learned from the web usage data to keep a constant rejuvenated user profile. Similarly, Spiliopoulou et al [28], Cooley et al [4], and Buchner and Mulvenna [3] have applied data mining techniques to extract usage pattern from web server logs. Perkowitz and Etzioni [21] again, proposed a cluster based mining algorithm together with data mining algorithm to analyze log servers in attempt to synthesize the index page.

In this paper we describe an approach to personalize web sessions, which takes into account the full spectrum of agent programming based on the BDI architecture using JACK, Java Server Pages (JSP) paired with web mining techniques. Our approach is described by the architecture shown in Fig. 1, which heavily relies on agent technology, thus bringing the personalization process not only automatic but dynamic too.

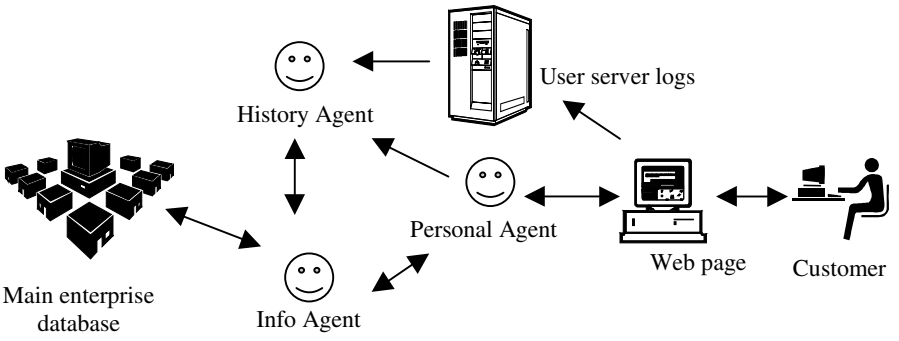


Fig. 1. System's Architecture

2 Agent Technology: BDI Agents

Researchers involved in agent technology have offered a variety of definitions, each carrying their own meanings of the word "agent."

According to Wooldridge and Jennings [Wooldridge and Jennings 95, page 2] *agent is "... a hardware or (more usually) software-based computer system that enjoys the following properties:*

- *autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;*
- *social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;*
- *reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;*
- *pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative."*

Traditional software systems are often unable to model rational behavior and experience a great deal of limitation in a *real time environment*, which encompasses *incomplete information* and *rapid change*. It thrives on the need for applications to exhibit rational, human like behavior in their respective problem domains. These properties give agents the edge to play a pivotal role in the information and process rich environment of e-commerce, which creates the need for agent based programming as the solution for current and future business needs [10]. The agent-oriented approach shares the same nature as object-oriented approach, except that agent are autonomous, this allows program development to be enhanced with the embedding of the desired action in modular units, operating individually thus giving

more flexibility with better performance. Although still in the phase of development, agent-oriented programming has already established much faith and shown promise in a variety of real-time applications such as air traffic management, air simulation, precision automation and distributed business systems, and e-commerce.

One agent-oriented framework, the BDI model, is widely recognized in philosophical and AI literature [1, 2, 9], and is regarded to be the best studied model of practical reasoning to the Agent Theories Architecture and Languages (ATAL) community [10]. BDI agents are characterized by three components: beliefs, desires, and intentions. The beliefs of an agent are its model of the domain, its desires provide some sort of ordering between states, and its intentions are the things it has decided to do. An agent's practical reasoning involves constant updating of beliefs obtained from its the environment, deciding what options are available, filtering these options to justify and determine new intentions, and acting on the basis of these intentions. A BDI agent gradually refines its intentions to primitive actions that can be executed as shown in Fig. 2 below [22].

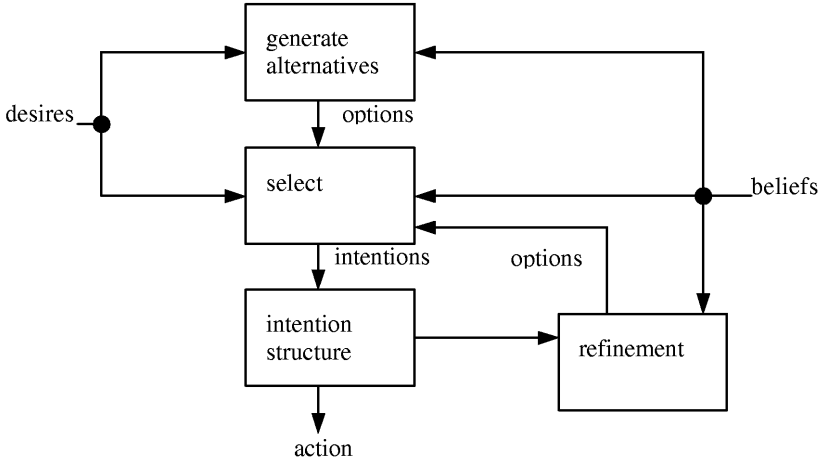


Fig. 2. BDI agent framework
Source : Jokelainen, 1998

The BDI architecture is increasingly being seen as an appropriate model to built intelligent autonomous agents. However researchers often focus on the specification or characterization of these rational agents and their behaviors are confined to specific environments [22]. And "it seems unlikely that there will be a single BDI logic that will be suitable for all applications" [Jokelainen 98, page 8]. Several programming languages have been developed based on this famous BDI framework. After an extensive evaluation we selected JACK, which provides BDI extensions to Java and this, would definitely comply with JSP.

3 JACK Agent : An Implementation

JACK is a third-generation agent programming language, designed as a set of light-weight components with high performance and strong data typing. JACK is actually made up by three components i.e., namely i) JACK Agent Language which is the actual programming language used to describe an agent-oriented software system. The JACK Agent Language is a super-set of Java – built from Java syntax while extending it with constructs to represent agent-oriented features. ii) JACK Agent Compiler pre-processes JACK Agent Language source files and converts them into Java codes. These codes can then be compiled into Java virtual machine code to run on the target system. iii) Finally the JACK Agent Kernel which is the runtime engine for programs written in the JACK Agent Language. It provides a set of classes that give JACK Agent Language programs their agent-oriented functionality. Most of these classes run behind the scenes and implement the underlying infrastructure and functionality that agents require, while others are used explicitly in JACK Agent Language programs.

The agents developed using JACK are *intelligent agents***,** i.e., they model reasoning behavior according to the theoretical BDI model. Adopting the BDI model, JACK intelligent agents are autonomous software components that have explicit goals to achieve or events to handle (*desires*). In other words, agents are proactive and reactive, and both behaviors are enacted through plans. Each plan describes how to achieve a goal under varying circumstances. A JACK agent remains idle until it is given a goal to pursue, or until it is has to respond to an event. When triggered, the agent pursues its given goals (*desires*), adopting the appropriate plans (*intentions*) according to its current set of data (*beliefs*) about the state of the world. This combination of desires and beliefs initiating context-sensitive intended behavior is part of what characterizes a BDI agent.

As mentioned, below are the ways of how JACK extends Java to support agent-oriented programming;

i. It defines new base classes, interfaces and methods.
ii. It provides extensions to the Java syntax to support new agent-oriented classes, definitions and statements, and
iii. It provides semantic extensions (runtime differences) to support the execution model required by an agent-oriented software system.

All the language extensions are implemented as Java plug-ins. This makes JACK as extensible and flexible as possible [13].

JACK BDI agents can assemble a plan set for a given event, apply sophisticated heuristics for plan choice and act intelligently on plan failure. At least one of the following characteristics applies to each type of BDI event in default:

i. meta-level reasoning - a technique for writing plans about how to select plans. This can help in refining plan selection to allow for selection of the most appropriate plan in a given situation; If the meta-level reasoning plan fails and does not select a plan for execution, then the default plan selection method is invoked.
ii. reconsidering alternative plans on plan failure - if a course of action (plan) fails, this technique allows an agent system to consider other

courses of action to achieve the goal that has been set; and it is this property that allows agents to avoid many of the pitfalls of more primitive reasoning models. Rather than having "one try" at achieving a goal, JACK agent can try a number of approaches to solve the problem by attempting any number of applicable plans.

iii. re-calculating the applicable plan set - it is possible when acting on plan failure to assemble a new plan set which excludes any failed plans by keeping track of the initially applicable instances.

Further, JACK allows the customization of the BDI behavior control by setting the behavior attributes that determines what happens with respect to plan reconsideration, plan set recalculation and meta-level reasoning. Also, embedded in JACK is a construct called capability, which is a technique to simplify agent system design, allows code reuse and encapsulation of agent functionality. Capabilities represent functional aspects of an agent that can be plugged in as when required. This *capability as component* approach allows an agent system architect to build up a library of capabilities over time. These components can then be used to add selected functionality to an agent. Additionally, as in object-oriented programming, capabilities can be structured so that a number of sub-capabilities can be combined to provide complex functionality in a parent capability. This capability can in turn be added to JACK agent to give it the desired functionality.

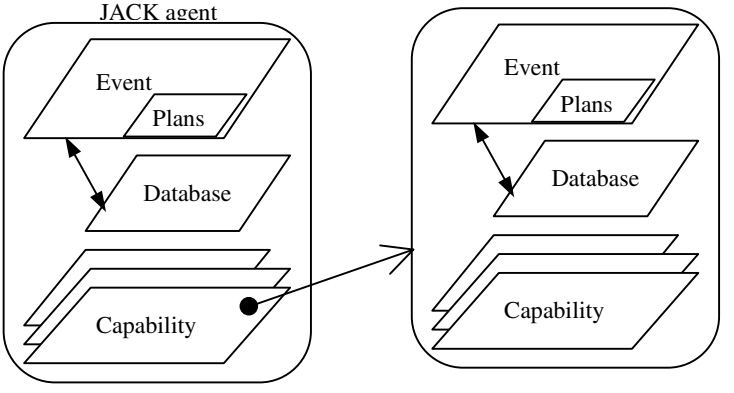


Fig. 3. Structure of a JACK agent showing event, plan, database and capability.

Figure 3 above describes the structure of a basic JACK agent. As shown, a JACK agent can handle a given set of events, in which each event is associated to one or more plans, at some instance accessing database. Capabilities are used to package together either events, plans or databases, or all if necessary to achieve a given goal. Capabilities are seen as a convenient way to share a common behavior between agents and it can also contain some sub-capabilities, which allow a tight control of the granularity of the shared behavior.

4 Multiagent Environment

Interaction is one of the most important features of an agent [19]. In other words, agents recurrently interact to share information and to perform tasks to achieve their goals. JACK provides the infrastructure for agent communication with its default communication protocol UDP/IP (user datagram protocol / internet protocol). JACK network communication, i.e., dci network allows communication among agents running from different processors. This system increases the flexibility and allows external communication. Since, any other than the default communication protocol could be used, we chose Knowledge Query and Manipulation Language (KQML) for the following reasons [8]:

i. KQML messages are opaque to the content they carry. KQML messages do not merely communicate sentences in some language, but rather communicate an attitude about the content (assertion, request, query).
ii. The language's primitive, called performatives. As the term suggests, the concept is related to speech act theory. Performatives define the permissible actions (operation) that agents may attempt in communicating with each other.
iii. An environment of KQML speaking agents can be enriched by facilitators i.e., special agent to facilitate with the functions (registration of database, communication services, etc.) of the other agents.

The performatives used in our proposed system are: tell, untell, ask, advertise, register, unregister, reply, achieve, standby, forward, and broadcast. JACK agents must meet the following basic requirements before any inter-agent communication can take place: i) the agents must know one another's name, ii) the sender (source agent) needs to able to send a message event, and iii) the receiver must be able to handle this message event.

5 System's Architecture

In our system each customer is assigned to a JACK agent. The agent determines the information to be presented at any given time, and should a customer return then the same agent that served them at a previous visit could be assigned to them. Our agents are particularly suited to personalizing the presentation of information based on customer preference and access behavior. As depicted in the Fig. 1 three autonomous agents handle the online task. The Personal agent is basically responsible for presenting the personalized information to the customer via the web page. The History agent communicates the pattern of web usage (extracted and analyzed from the server logs after every user session) to the Info agent to be stored. Here only the Info agent has direct access to the main enterprise database.

The actual process of Web personalization in our system can be divided into two parts, namely (i) on-line data transferring and processing, and (ii) off-line data management. As shown in this system is made up of three types of autonomous agents i.e., History agent, Info agent and Personal agent supported by the main enterprise database and the server logs. The off-line component of our system bears the task of data preparation and organizing, including database filtering, cleaning and updating,

uploading of new information ready to be published, and removal of redundant information. Web mining makes up the other part of the offline component. Data collected from the server log is coupled to form as web log data hypercubes [3] as shown in Fig. 4, which is later analyzed using OLAP techniques. The derived information (marketing intelligence) is later used to form marketing decisions such as segmentation, target marketing and product differentiation - customization of product.

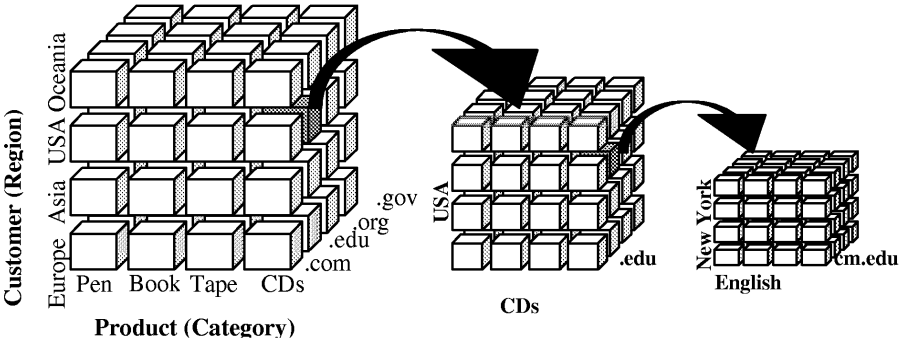


Fig. 4. Three dimensional Web log data hypercube used for Segmentation and Target Marketing. **Source: Adapted from** Buchner, and Mulvenna, 1998.

Many marketers fail to gain repeated customer visits because they fail to identify customer desires and intentions. Such information is essential to increase sales [30, 11, 15] and will play a pivotal role in understanding consumer on-line shopping behavior. To make things more difficult customer needs and preferences change over which calls for constant monitoring and updating marketing intelligence.

Customization of web objects and content is accomplished using JSP linked to Jack BDI agents. Apart from monitoring and storing customer selection from the web page, Jack agent matches the previous access and the marketing decisions before displaying the information using JSP. Different customers may receive the same information in different style of presentation, which is, deduced from their on-line behavior. The main advantage of using JSP is in tracking online access behavior using beans[3]. Jack agents can be placed as beans in the *scope* attribute to track customer movement in the web page as shown below. In the example below the Jack agent (Bean) is 'alive' – active (continuously monitoring) throughout the lifespan of the system no matter how many times the particular user logs in and out.

Our agent framework uses the following techniques to handle merging, change, incompleteness and uncertainty that have all been implemented in Java and will be embedded into the agents using the JACK Intelligent Agent infrastructure;

[3] Called also as Java Beans. It is a set of reusable java codes ready to be embedded into other Java applications.

i. *Theory extraction* is used for fusing information from multiple sources [33],
ii. *Belief revision* is used to model change in a dynamic environment and to support decision making in the face of incomplete information [31,32] and
iii. *Possibilistic reasoning* is used to handle uncertain information [5].

The marketspace is becoming more complex and more competitive in every industry sector. Change, incompleteness and uncertainty are prominent elements of that marketspace. Software agents must be able to cope with change effectively because change is an omnipresent feature of the current market space.

6 The Agents

6.1 The Personal Agent

Each customer is assigned a personal agent instantiation. This agent is solely responsible for personalizing information for each customer based on his access pattern. To be effective this agent must deal appropriately with incomplete information. It can make requests for additional information (such as its beliefs) from the Info agent, and it can closely monitor the user to capture the browsing activity. The Personal agent carries the burden of performing the personalization process, hence given the priority in plan execution. The more frequent a customer uses the site the more personalized the information presentation becomes. When the generated plan fails the Personal agent will execute the default plan i.e., to allow the customer to access the default page. Using JACK's and it records the failed and discards the plan when recalculating plans again.

6.2 The History Agent

This agent is responsible for performing data mining on the information stored earlier from the Personal agent and from the server logs. Once the analysis is done the information is passed to the Info agent. The goal of this agent is to identify patterns amongst customer usage sessions and to update the Info agent regarding these patterns. Once a customer's logs are received (from the Personal agent), referring to the customer's identification, History agent requests the previous pattern from the Info agent and re-organizes the access pattern.

6.3 The Info Agent

This is the agent that has access to the main enterprise database – the main database. The immediate goal of this agent includes providing cached information (frequently requested by Personal agent). The Info agent does most of the off-line task of the system and will frequently be fed information about the website by the human administrators. This agent often carries out requests for the History agent, and actively communicates with the Personal agent in an attempt to satisfy and provide enough assistance to personalize a web session.

7 Conclusion

In summary we have proposed an intelligent agent architecture, which will support the decision-making requirements of e-commerce applications, which have continuously evolving needs. In particular, the key contribution of this paper is the development of an intelligent agent framework that provides the infrastructure to nurture customer relationships. This is achieved by providing personalized web sessions and by handling decision making in an uncertain dynamic environment where information from heterogeneous sources may need to be integrated with the support of sophisticated intelligent processing.

In this paper we described an agent-oriented architecture based on the BDI, which is based on practical reasoning portraying human mental state of Beliefs, Desires and Intentions. Our architecture is under construction using JACK that is an agent-oriented development environment. It supports the BDI framework and is built on top of the Java development environment. It not only provides the entire necessary agent infrastructure for our architecture, but it also allows us to use previously developed Java modules such as [33] in an agent environment.

The main advantage of using an agent-based approach is that the agents interact autonomously and power is devolved to the agents, i.e. there is no need for a super-agent to oversee the communication and interaction. This leads to better performance and a more personalized web browsing session. This research will be extended to build a dynamic database containing each customer profile, which could be updated after each user session. This customer database will be a valuable aid to marketing decisions and help to support the firm's wider success.

References

1. Bratman M. E.. 1987. Intentions, Plans, and Practical Reason. Havard University Press, Cambridge, M.A..
2. Bratman, M. E, Isreal, D., and Pollack, M. E.. 1988. Plans and resource-bounded practical reasoning. Computational Intelligence, 4:439-355.
3. Buchner, A. and Mulvenna, M. D., Discovering internet marketing intelligence through online analytical Web usage mining. *SIGMOD Record*, (4) 27, 1999.
4. Cooley, R., Mobasher, B., and Srivastava, J., Data preparation for mining World Wide Web browsing patterns. *Journal of Knowledge and Information Systems*, (1) 1, 1999.
5. Dubois, D. and Prade, H., Possibilistic Logic. *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume3, Nonmonotonic Reasoning and Uncertain Reasoning, Gabbay, D., Hogger, C., and Robinson, J. (eds), Claredon Press, Oxford. (1994).
6. Ernst and Young. 1999. Special Report Technology in Financial Services. E-commerce. Customer Relationship Management. [Online] http://www.ey.com/
7. Field. C., 1996. 'Retailing on the Internet, the future for online commerce", Financial Times Management Reports, London.
8. Finin, T., Yannis Labrou, and James Mayfield. 1997. KQML as an agent communication language, in Jeff Bradshaw (Ed.), ``Software Agents", MIT Press, Cambridge.
9. Georgeff M. P. and F. F. Ingrand. 1989. Decision-making in an embedded reasoning system. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, MI.

10. Georgeff M. P, Pell B., Pollack M., Tambe M. and. Wooldridge M. 1999. The Belief-Desire-Intention Model of Agency. *Proceedings of the 5th International workshop ATAL 98*
11. Hagel III, J., Rayport, J. *The coming battle for customer information* The McKinsey Quarterly, 1997 Number 3, pp. 65—76. (1997)
12. IMR Worldwide. WebMeasure technical notes. WebMeasure release June 1999
13. Jack Intelligent Agents manual. 2000. Agent Oriented Software [Online] http://www.agent-software.com/
14. Jokelainen, A.S 1998. Autonomous Agent in Artificial Intelligence Research. Object oriented software and intelligent agents in future automation systems - Seminar Autumn 1998 [Online] http://www.automation.hut.fi/edu/as84360/autumn98/jokelainen/jokelainen.htm
15. Kierzkowksi, A., Mcquade, S., Waitman, R., Zeisser, M. Marketing to the digital consumer The McKinsey Quaterly, 1996 Number 2, pp. 180-183. (1996)
16. Mobasher, B., Cooley, R., and Srivastava, J., 1999. Automatic Personalization Through Web Usage Mining, Technical Report TR99-010, Department of Computer Science, Depaul University.
17. Morrisette S,. April 1998 The Retail Power Shift Forrester Report [Online. http://www.forrester.com/ER/Research/Report/Excerpt/0,1358,3861,FF.html
18. Nasraoui, O., Frigui, H., Joshi, A., Krishnapuram, R., Mining Web access logs using relational competitive fuzzy clustering. To appear in *the Proceedings of the Eight International Fuzzy Systems Association World Congress*, August 1999.
19. Nwana, H.S., 1996. Software Agents: An Overview. In: The Knowledge Engineering Review, October/November, Volume 11, Number 3, pages 205-244.
20. Perkowitz, M. and Etzioni, O., 1998. Adaptive Web sites: automaticlly synthesizing Web pages. In *Proceedings of Fifteenth National Conference on Artificial Intelligence*, Madison, WI.
21. Perkowitz, M. and Etzioni, O., Towards Adaptive Web sites: C0onceptual Framework and Case StudyArtificial Intelligence Journal, v 188, n1/2, 2000.
22. Rao, A.S. 1996. Decision Procedures for Propositional Linear-Time Belief-Desire-Intention Logics, in Wooldridge, M. Müller, J.P. Tambe, M. (Eds.), Intelligent Agents II: Agent Theories, Architectures, and Languages, Springer-Verlag, Berlin and Heidelberg, Germany, pp.33-48
23. Robert Frances Group. 1999. A Knowledge Shift Becomes a Power Shift – Anticipate it or Vanish [Online]. www.rfgonline.com/unprotectedarchive/research/060399nt.html.
24. Schechter, S., Krishnan, M., and Smith, M. D., Using path profiles to predict HTTP requests. In *Proceedings of 7th International World Wide Web Conference*, Brisbane, Australia, 1998.
25. Shardanand, U., Maes, P., 1995. Social information filtering: algorithms for automating "word of mouth." In *Proceedings of the ACM CHI Conference*.
26. Shahabi, C., Zarkesh, A. M., Adibi, J., and Shah, V., 1997. Knowledge discovery from users Web-page navigation. In *Proceedings of Workshop on Research Issues in Data Engineering*, Birmingham, England.
27. Shern, S and Crawford, F. 1999. The 2nd Annual Ernst &Young. Internet Shopping Study. The Digital Channel Continuous to Gather Steam. [Online]. http://www.ey.com.au
28. Spiliopoulou, M. and Faulstich, L. C., 1999. WUM: A Web Utilization Miner. *In Proceedings of EDBT Workshop WebDB98*, Valencia, Spain, LNCS 1590, Springer Verlag.
29. Terpsidis, I. S., Moukas, A., Pergioudakis, B., Doukidis, G., and Maes, P., 1997. The potential of Electronic Commerce in Re-engineering-retailer relationships through Intelligent agents. *Proceedings of the European Conference on Multimedia and E-Commerce*, Florence, Italy 1997

30. Webley, M., Coates, A., King, S., 1999. Knowing your customer. A research report into customer information. KPMG Consulting. [Online. http://www.kpmg.com.uk (1999, Aug. 17)
31. Williams, M.A., 1997. *Anytime Belief Revision*, in the Proceedings of the International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 74-80.
32. Williams, M. A., Applications of Belief Revision, in Transactions and Change in Logic Databases, LNAI 1472, 285 - 314, Springer Verlag, 1998.
33. Williams, M.A and Sims, A., 2000. SATEN: an Object-Oriented Web-Based Revision and Extraction Engine, in the online *Proceedings of the International Workshop on Nonmotonic Reasoning, 2000*. [Online] Computer Science, abstract cs.AI/0003059 http://arvix.org/abs/cs.AI/0003059
34. Wooldridge, M and Jennings, R., J. 1995. Agent Theories, Architectures, and Languages: a Survey, "in Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1-22.

Possibilistic Reasoning for Intelligent Payment Agents

On Wong[1] and Raymond Lau[2]

[1] School of Computing Science
Queensland University of Technology
GPO Box 2434, Brisbane, Qld 4001, Australia
o.wong@qut.edu.au
[2] Cooperative Information Systems Research Centre
Queensland University of Technology
GPO Box 2434, Brisbane, Qld 4001, Australia
raymond@icis.qut.edu.au

Abstract. With the rapid growth of electronic commerce on the Internet, it becomes increasingly important to have effective, secure, and low-cost methods of handling the monetary aspect of the on-line transactions. Recently, a number of electronic payment services have been introduced to the Web. An agent-based approach for electronic payment services is appealing since payment agents can *proactively* monitor the latest market information and *autonomously* select the best settlement option on behalf of their customers. However, because of the intrinsically dynamic nature of the Internet, these payment agents are faced with the challenges of making good decisions based on uncertain and incomplete market information. Possibilistic logic provides an expressive language to capture these uncertainties, and a robust and powerful reasoning method to make sound decisions by considering the uncertainties related to the activities in payment settlements. In addition, possibilistic deduction can be used to explain and justify an agent's decisions. Enhanced explanatory capability promotes users' trust and satisfaction, and this is essential in agent-mediated electronic commerce. This paper proposes an agent-based electronic payment service. In particular, how possibilistic logic can be applied to the development of intelligent payment agents is discussed.

1 Introduction

The exponential growth of the Internet has rapidly changed the way businesses are performed, and the way consumers do their shopping [12,15,16,17]. Recently, a number of electronic payment services have emerged on the World Wide Web (Web) [18]. No doubt, electronic payment services will grow very rapidly and several large online organizations like Yahoo, AOL have already joined in the race. There are several advantages for the consumers to make use of electronic payment services on the Web. Firstly, these payment services autonomously and accurately process the incoming bills on behalf of the consumers. Therefore,

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 170–182, 2001.

consumers do not need to worry about keeping track of their bills and paying penalties for late payments. Secondly, there could be financial advantage of using electronic payment services. For example, the total transaction cost may be reduced by having a single bulk payment from the payment service rather than having several individual micro payments settled between a consumer and their biller. Security is also an important issue for electronic commerce. Payment service providers take care of all the security issues with other payment service providers, or the ultimate financial institutes where bill settlements take place. Therefore, a consumer's risk of conducting on-line shopping is reduced to a minimum. In fact, similar advantages are also brought to the virtual stores or utility companies. For example, they do not need to spend lots of money to set up their own payment services on the Web. Moreover, they are guaranteed that payments will be received once the services or goods are delivered to their clients.

The market of Internet-based electronic payment services is highly competitive, and there are new payment services brought to the Web everyday. In order to make profit and survive under such a keen competition, the payment service providers must be able to select the best settlement options (e.g. cheap, secured, and reliable settlement services) according to the latest market information such as the transaction costs and the service qualities of the external payment/settlement service providers. An agent-based electronic payment service is appealing because payment agents, which are a kind of business agents, can *proactively* monitor the latest market information on the Internet. Moreover, they can *autonomously* keep track of bill payments on behalf of each registered client and make sensible decisions regarding optimum settlement options available in a payment cycle. Business agents can also learn the consumers' shopping requirements and recommend appropriate products for purchasing [14,16]. Nevertheless, one difficulty that payment agents need to deal with is that market information (e.g. prices and service qualities of settlement services) available on the Internet is highly volatile. For instance, even though a settlement service was up and running few hours' ago, it may be out of service at the current payment processing cycle. Its transaction cost may vary quite frequently. This trend has already been revealed in nowadays telecommunication market. If a settlement services is just introduced to the Web, it is difficult to obtain information about its service characteristic (e.g. reliability) at all.

Therefore, payment agents are faced with the challenge of making good settlement decisions based on uncertain and incomplete information. Though sophisticated quantitative methods have been develop to model the decision making processes [6,11], there are weaknesses of these methods. For example, those axioms underlying a particular decision theory may not be valid under all decision making situations [6]. Bayesian network has been widely used for uncertainty analysis. However, it is difficult to obtain all the conditional probabilities required in a large network model. Even if assumptions of independency between events simplify the calculations, it may not reflect the realities of a problem domain. In the context of agent mediated electronic commerce, it is desirable for business agents to explain and justify their decisions so that consumer trust and

satisfaction can be obtained [8,9]. It is not easy to explain an agent's decisions purely based on a quantitative model where the relationships between various decision factors are buried by numerous conditional probabilities or weight vectors.

This paper reports a preliminary study of applying possibilistic logic to payment agents so that uncertain and incomplete market information can explicitly be represented and reasoned about during agents' decision making. Moreover, under such an integrated framework of quantitative and symbolic approaches, the explanatory power of payment agents can also be enhanced because the relationships between a decision and its justifications can be represented by more comprehensible causal rules. Enhanced explanatory capabilities of payment agents will subsequently lead to improved consumer trust, and the deployment of these agents to Web-based electronic commerce. Section 2 gives an overview of agent-based electronic payment service. Preliminaries of possibilistic logic are provided in Section 3. The formulation of a payment agent's decision rules, and the representation of uncertain market information are discussed in Section 4. Section 5 illustrates how possibilistic reasoning can be applied to the payment agents for making settlement decisions. Finally, future work is proposed, and a conclusion summarising our findings is given.

2 System Overview of ABPS

An overview of the proposed Agent-based Bill Payment Service (ABPS) is depicted in Figure 1. The Web site housing this system is certified digitally so that all the related parties can verify its identification. Moreover, encrypted transmissions are used to exchange information with external agents (e.g. customers, billers, banks, external payment services, etc.).

A consumer must first register with ABPS by providing their personal information such as their digital certification and bank details. After acquiring utility services (e.g. electricity) or conducting on-line shopping (i.e. step 1 in figure 1), registered customers or their agents will authorise ABPS to pay the related parties (i.e. step 2). The payment agent of ABPS can then handshake with the retailers or their agents to obtain settlement instructions (i.e. step 3). It is assumed that extensible markup language (XML) [7] is used to facilitate the exchange of such information. Finally, the payment agent settle the bills via appropriate financial institutes (e.g. banks) or external payment services (i.e. step 4). Communications with these external agents can be conducted via dedicated lines, Internet connection with Secure Sockets Layer (SSL) or Secure Electronic Transaction (SET). These external financial institutes or their agents may pass the payments through the settlement chain until the payments can finally reach the bankers of the retailers or the utility companies. For the discussion in this paper, these external financial institutes or payment services are collectedly referred to as settlement services or settlement agents. The focus of this paper is step 4 from which the payment agent select the optimum settlement options according to the service characteristics of pertaining settlement services dur-

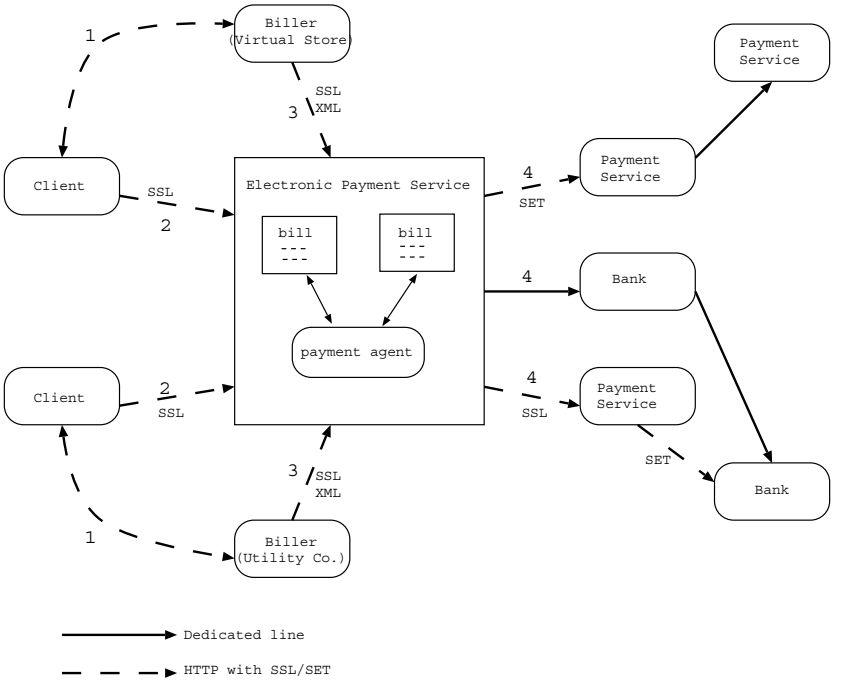


Fig. 1. An Overview of Agent-based Bill Payment Service

ing a payment processing cycle. Because of the dynamic nature of the Internet, service characteristics of these settlement services are uncertain or incomplete. Therefore, a formalism that allows explicit representation of uncertainties and sound reasoning about these uncertainties is required for the development of the payment agents.

Since all the billing and payment records are stored in ABPS, a registered customer can review their billing and payment histories. Moreover, they can specify regular payment requirements (e.g. direct debits) and set maximum payment limit. In each payment processing cycle, the payment agent will scan through the billing database and extract billing items which are going to be due. Billing items are then grouped into bulk payments according to the settlement instructions pertaining to the payments. Then, the payment agent needs to decide which settlement service is the best (e.g. being able to meet the payment deadline, cheap and secured) to settle each bulk payment because there could be more than one settlement options available during a payment processing cycle.

3 Preliminaries of Possibilistic Logic

This paper refers to necessity-valued possibilistic logic [1,3]. Let $\mathcal{L}$ be a classical first-order language and Ω be the set of classical interpretations for $\mathcal{L}$. A

possibilistic formula is of the form (α, m), where $\alpha \in \mathcal{L}$ is a closed formula and $m \in (0, 1]$ is a positive number. The intuitively meaning of the formula (α, m) is that the truth of α is certain at least to the degree m (i.e. $N(\alpha) \geq m$, where N is the necessity measure of α). The semantics of a set of classical formulae Δ is defined by a subset $\mathcal{M}(\Delta) \subset \Omega$ that satisfies all formulae in Δ. Each interpretation $\omega \in \mathcal{M}(\Delta)$ is called a model. For a set of possibilistic formulae $F = \{(\alpha_i, m_i), i = 1, \ldots, n\}$, a *possibility distribution* π over Ω that characterises the *fuzzy set of models* $\mathcal{M}(F)$ is introduced. In other words, F induces a preference ordering over Ω via π. The necessity measure N induced by π is then defined as [1]:

$$\forall \alpha \in \mathcal{L},\ N(\alpha) = \inf\{1 - \pi(\omega), \omega \models \neg\alpha\}$$

In other words, the necessity measure (i.e. the lower bound certainty) of a formula α equals to the complement of the highest *possibility* attached to an interpretation ω where $\neg\alpha$ is classically satisfied. A possibility distribution π on Ω is said to satisfy the necessity-valued formula (α, m) (i.e. $\pi \models (\alpha, m)$), iff $N(\alpha) \geq m$. $\pi \models F$, iff $\forall i = 1, \ldots, n,\ \pi \models (\alpha_i, m_i)$. If $\forall \pi, \pi \models (\alpha, m)$, (α, m) is said to be valid and is denoted $\models (\alpha, m)$. A possibilistic formula Φ is a logical consequence of a set of possibilistic formulae F (i.e. $F \models \Phi$), iff $\forall \pi, \pi \models F$ implying $\pi \models \Phi$. The consistency of F (i.e. $Cons(F)$) is a measure of the degree that there is at least one completely possible interpretation for F and is defined by $Cons(F) = sup_{\omega \in \Omega} \pi_F(\omega)$. Then, the inconsistency degree of F is defined by: $Incons(F) = 1 - Cons(F) = 1 - sup_{\omega \in \Omega} \pi_F(\omega)$.

The deduction problem in possibilisticlogic is taken as finding the best valuation m (i.e. $Val(\beta, F)$) such that $F \models (\beta, m)$. Based on the classical resolution, a sound and complete proof method has been developed [1,4] as follows:

1. A possibilistic knowledge base $F = \{(\alpha_i, m_i), i = 1, \ldots, n\}$ is converted to possibilistic clausal form (i.e. conjunctive normal form) $C = \bigwedge_{i,j}\{(c_{i,j}, m_i)\}$ where $c_{i,j}$ is a universally quantified classical first-order clause.
2. The negation of a first-order formula β, which is to be deduced from F, is also converted to clausal form $c_1, \ldots, c_n$.
3. Let $C' = C \cup \{(c_1, 1), \ldots, (c_n, 1)\}$.
4. Search for the empty clause $(\perp, \bar{m})$ by applying the *possibilistic resolution rule*:
$$(c_1, m_1), (c_2, m_2) \vdash (R(c_1, c_2), \min(m_1, m_2)) \quad (1)$$
from C' repeatedly until the maximal $\bar{m}$ is found.$R(c_1, c_2)$ is the classical resolvent of c_1 and c_2.
5. Let $Val(\beta, F) = \bar{m}$.

In addition,it has been shown that the inconsistency degree $Incons(F)$ of any possibilistic knowledge base F is the least certain formula $\min\{m_1, \ldots, m_n\}$ involved in the strongest contradiction (i.e. $(\perp, \bar{m})$). Based on these propositions, *nontrivial possibilistic deduction* ($\mid\!\sim$) is defined as [1]:

$$F \mid\!\sim (\beta, m) \text{ iff } F \models (\beta, m) \text{ and } m > Incons(F)$$

The intuition behind nontrivial possibilistic deduction is that the proof of a formula β is restricted to the subset $F^{'}$ of a partially inconsistent possibilistic knowledge base F where the information is more or less certain. For example, $\forall(\alpha_i, m_i) \in F^{'} : m_i > Incons(F)$. In other words, the certainties attached to the formulae in $F^{'}$ must be strictly higher than the formula in F that causes the inconsistency. Therefore, β is proved with the best knowledge (i.e. the most certain piece of knowledge) from the knowledge base.

4 Knowledge Representation

It is believed that the first step towards decision making is to produce a quantitative description of the problem domain [10] (e.g. identifying decision variables and estimating uncertainties). This is also applied to a possibilistic framework of knowledge representation and reasoning. For the examples discussed in this paper, the decision variables are assumed to be: processing time, availability, risk, and transaction cost. Processing time and availability are considered as *hard constraints*, whereas transaction cost and risk are treated as *soft constraints* [10]. For example, even though a particular settlement service is the most expensive one, the payment agent may still consider it because of its ability to meet the payment deadline. Table 1 is an example of four settlement services with corresponding service characteristics.

Table 1. Characteristics of Settlement Services

Services	Processing time	Availability	Risk	Cost
A	2 hours	Running	Low	60 cents/tran.
B	2 hours	Running	Medium	60 cents/tran.
C	1 hour	Running	Low	90 cents/tran.
D	1 hour	Likely Running	High	30 cents/tran.

To transform the service characteristics depicted in Table 1 to possibilistic formulae, the certainty value associated with each characteristic (i.e. service value) pertaining to a settlement service is computed. For example, even though a settlement service announces that its processing time for a transaction is 2 hours on average, there is uncertainty related to this quoted figure because of the dynamic of the Internet (e.g. certain sites along the settlement chain are down). Similarly, even though the payment agent can be "handshake" with a settlement service (e.g. by using the "ping" command), the service may go down when a transaction is finally routed there.

Table 2 depicts the possibilistic formulation of the service characteristics for these settlement services. Under each service attribute (e.g. processing time), the left hand column is the predicate formulae of corresponding service values, and the right hand column is the associated certainty values. It is assumed

that a history file is kept for each settlement service. If a settlement service is brand new, default values and associated certainties could be applied. Based on the history file, if a settlement service A announces an average processing time of 2 hours, and nine out of ten times that a payment can be settled within 2 hours, the certainty attached to the predicate $Proctime(A, 2)$ will be 0.9. The predicate $Proctime(x, y)$ means that the average processing time of the settlement service x is y hour. Availability of a settlement service is modelled by the predicate $Down(x)$, which indicates if a service is down or not. If a settlement service does not respond to the initial "handshake" command, the certainty of $Down(x)$ is assumed 1.0. In other words, a pessimistic estimation is used in ABPS. However, even though the remote system responds, it is still possible that it will be out of service when a transaction arrives. The certainty m of $Down(x)$ is then computed based on the history records (e.g. frequency of out of service given that it responds at the "handshake" stage). It is assumed that routing payments to external settlement services is risky in nature. The certainty of the predicate $Risk(x)$ is derived subjectively according to certain basic features of the settlement service x. For example, it can be assessed based on whether it is a certified site, transaction history, kinds of secured transmission protocols supported (SSL/SET), etc.. If a settlement service A is originated from a certified site and it supports highly secured transmission protocol, it is not as risky as another agent D which is not certified at all. The cost factor is modelled as a *fuzzy predicate* $Expensive(x)$ of a settlement service x. Expensive is a relative term for the evaluation of settlement services. There is not a definite value which will be considered as expensive. Among a group of settlement services, the one with the highest transaction cost will be considered as expensive with certainty 1.0, whereas the cheapest agent will be assigned a certainty of 0.0. The rest of the settlement services can be evaluated according to a *fuzzy membership function* [19] μ. For the discussion in this paper, it is assumed that $\mu_{Expensive} = \frac{x-lowest}{highest-lowest}$, where *highest* and *lowest* represent the highest and the lowest transaction costs as quoted by the settlement services, and x is the transaction cost of a particular settlement service.

Table 2. Reformulation of Service Characteristics

Services	Processing Time		Availability		Risk		Cost	
A	$Proctime(A, 2)$	0.9	$Down(A)$	0.2	$Risky(A)$	0.3	$Expensive(A)$	0.5
B	$Proctime(B, 2)$	0.8	$Down(B)$	0.3	$Risky(B)$	0.4	$Expensive(B)$	0.5
C	$Proctime(C, 1)$	0.8	$Down(C)$	0.3	$Risky(C)$	0.3	$Expensive(C)$	1.0
D	$Proctime(D, 1)$	0.5	$Down(D)$	0.5	$Risky(D)$	0.7	$Expensive(D)$	0.0

The payment agent of ABPS will only select a settlement service x as a candidate $Cand(x)$ for payment processing if it is likely to satisfy the hard constraint such as processing time. The payment agent prefers settlement services which are cheap and less risky. The following formulae represent the payment agent's

expectations of a good candidate service $Cand(x)$ and a bad candidate service $\neg Cand(x)$:

$$Proctime(x, y) \wedge Reqtime(z) \wedge lq(y, z) \rightarrow Cand(x), 1.0$$
$$Down(x) \rightarrow \neg Cand(x), 1.0$$
$$Risky(x) \rightarrow \neg Cand(x), 0.6$$
$$Expensive(x) \rightarrow \neg Cand(x), 0.8$$
$$Expensive(x) \rightarrow \neg Risky(x), 0.4$$
$$Reqtime(2), 1.0$$

The first formula means that if a settlement service x's processing time is y hour and it is less than or equal to (i.e. predicate $lq(y, z)$) the required processing time z of the current processing cycle, it is a candidate service (i.e. $Cand(x)$) with certainty 1.0; If a service x is unavailable (i.e. $down(x)$), it is not a candidate service (i.e. $\neg Cand(x)$) with certainty 1.0; A risky service is not a candidate service with certainty 0.6; An expensive settlement service is not a candidate service with certainty 0.8; An expensive service implies that it is not risky with certainty 0.4, a low confidence level. It is assumed that the deadline of processing the bulk payments in the current processing cycle is 2 hours, and so $Reqtime(2), 1.0$ is added to the payment agent's knowledge base K.

5 Possibilistic Reasoning for Smart Payment Processing

The payment agent in ABPS employs possibilistic deduction to infer the best settlement option(s) according to its preference (e.g. cheap, less risky, meeting deadline, etc.) and the characteristics of the settlement services. When a settlement service is evaluated, its characteristics are encoded as possibilistic formulae and added to the payment agent's knowledge base as *assumptions*. If the payment agent's knowledge base K entails a settlement service x as a candidate service $Cand(x)$, x will be added to the set S of feasible solutions. This scenario is similar to the symbolic approach of finding a feasible plan or schedule through *assumption based reasoning* in MXFRMS [13], where each possible plan is added to the agent's knowledge base as assumption. If the plan is consistent with the knowledge base where the constraints about the planning process is stored, the plan is considered feasible. However, for the payment agent, some of the characteristics of a service x may indicate that it should be a candidate service, whereas other service qualities may exclude x from consideration. This is a general problem in multiple criteria decision making [20]. In logical term, inconsistency $\perp$ may exist in the resulting possibilistic knowledge base K. Therefore, the nontrivial possibilistic deduction $\mid\sim$ is used so that the payment agent can draw sensible conclusions based on the most certain part of K even though some of its elements are contradictory to each other. To choose the optimal settlement option(s) from S, the valuation $Val(Cand(x), K)$ will be compared for each $x \in S$. A candidate settlement agent that receives the strongest valuation will be chosen as the settlement service since the payment agent is most certain that it is a candidate service.

With reference to our example, the predicates representing the characteristics of each settlement service will be added to the payment agent's knowledge base K as assumptions. This process creates K_A, K_B, K_C, and K_D respectively. By possibilistic resolution, the inconsistency degree of each revised knowledge base can be computed (e.g. $Incons(K_A)$). The second step is to compute the valuation $Val(Cand(x), K_x)$. According to the definition of nontrivial possibilistic deduction $\mid\sim$ described in Section 3, if $Val(Cand(x), K_x) > Incons(K_x)$ is true, $K_x \mid\sim Cand(x)$ can be established. To conduct possibilistic resolution, all the formulae must be converted to their clausal form first. The following is a snapshot of K_A after the characteristics of settlement service A are added to K:

$$\neg Proctime(x, y) \vee \neg Reqtime(z) \vee \neg lq(y, z) \vee Cand(x), 1.0$$
$$\neg Down(x) \vee \neg Cand(x), 1.0$$
$$\neg Risky(x) \vee \neg Cand(x), 0.6$$
$$\neg Expensive(x) \vee \neg Cand(x), 0.8$$
$$\neg Expensive(x) \vee \neg Risky(x), 0.4$$
$$Reqtime(2), 1.0$$

$$Proctime(A, 2), 0.9$$
$$Down(A), 0.2$$
$$Risky(A), 0.3$$
$$Expensive(A), 0.5$$

It can easily be observed that K_A is partially inconsistent (e.g. $Cand(A) \in K_A$, $\neg Cand(A) \in K_A$). Therefore, $Incons(K_A) > 0$ is derived. In addition, there are several valuations of $Val(\bot, K_A)$. The maximal valuation $\bar{m} = 0.5$ is derived from the resolution path depicted in Figure 2. It is assumed that standard unification procedure supported by Prolog and the possibilistic resolution procedure described in Section 3 are used to derive the refutation. As can be seen, $Incons(K_A) = 0.5$ is derived. To find an optimal refutation without traversing a large number of nodes along the resolution paths, efficient resolution strategies have been proposed [2]. To see how strong the payment agent's knowledge base K_A entails $Cand(A)$ (i.e. how strong the payment agent believes that service A is a candidate for processing payments), the valuation $Val(Cand(A), K_A)$ is computed. According to the resolution procedure described in Section 3, $(\neg Cand(A), 1)$ is added to K_A, several refutations can be found again. Figure 3 depicts the resolution path with the strongest refutation. Therefore, $Val(Cand(A), K_A) = 0.9$ and $K_A \models (Cand(A), 0.9)$ are established. According to the definition of nontrivial possibilistic deduction $\mid\sim$ described in Section 3, $K_A \mid\sim Cand(A)$ since $Val(Cand(A), K_A) = 0.9 > Incons(K_A) = 0.5$. In other words, settlement service A is a feasible solution.

Similarly, service characteristics such as $(Proctime(B, 2), 0.8)$, $(Down(B), 0.3)$, $(Risky(B), 0.4)$, and $(Expensive(B), 0.5)$ are also added to the possibilistic knowledge base K as assumptions. By means of possibilistic resolution, $Incons(K_B) = 0.5$ and $Val(Cand(B), K_B) = 0.8$ are derived. Therefore, the payment agent can also deduce that $K_B \mid\sim Cand(B)$. Consequently, settlement service B is also a potential solution. According to the characteristics of the

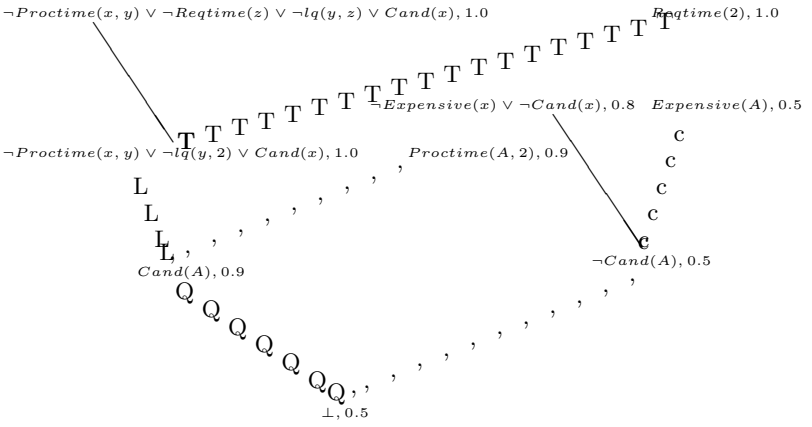


Fig. 2. Resolution Path for the maximal $Val(\bot, K_A)$

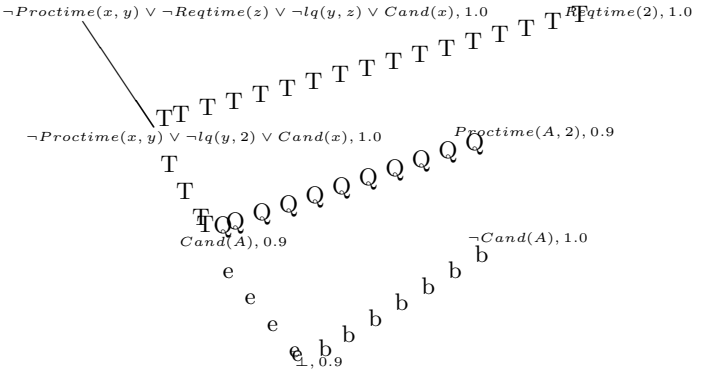


Fig. 3. Resolution Path for the maximal $Val(Cand(A), K_A)$

settlement services described in Table 2, $(Proctime(C, 1), 0.8)$, $(Down(C), 0.3)$, $(Risky(C), 0.3)$, and $(Expensive(C), 1.0)$ are also added to K as assumptions. By means of possibilistic resolution, $Incons(K_C) = 0.8$ and $Val(Cand(C), K_C) = 0.8$ are derived. The payment agent deduces that $K_C \mid\!\sim Cand(C)$ based on the definition of nontrivial possibilistic deduction. Therefore, settlement service C will not be considered for payment processing because the payment agent is not certain that C should be a candidate service (e.g. it is an expensive service). To evaluate the settlement service D, the assumptions of $(Proctime(D, 1), 0.5)$, $(Down(D), 0.5)$, $(Risky(D), 0.7)$, and $(Expensive(D), 0.0)$ are used to derive K_D. Accordingly, $Incons(K_D) = 0.5$ and $Val(Cand(D), K_D) = 0.5$ are computed. Therefore, $K_D \mid\!\sim Cand(D)$ is also established. In other words, it is uncertain that settlement service D can fulfill the requirements of being a candidate service because of its frequent down time and relatively high risk. In summary,

the payment agent draws the following conclusions based on nontrivial possibilistic deduction:

$$K_A \mid\sim Cand(A)$$
$$K_B \mid\sim Cand(B)$$
$$K_C \mid\approx Cand(C)$$
$$K_D \mid\approx Cand(D)$$

Since $Val(Cand(A), K_A)$ is higher than $Val(Cand(B), K_B)$, the payment agent is more certain that settlement service A is a candidate service for payment processing. The intrinsic nature of the Internet is very dynamic, and so is the Web-based electronic commerce. A full set of service attributes pertaining to a particular settlement service may not be available for decision making. Therefore, it may be desirable for the payment agent to be able to draw sensible conclusions with incomplete information. Possibilistic reasoning is also useful under such circumstance. For instance, if the only information available for service A is processing time and availability, $K_A \mid\sim Cand(A)$ is still maintained according to the given evidence. So, A is still considered as a feasible solution because of its likelihood to meet the payment deadline. However, further investigation is required to study the expected rational behavior of payment agents when they are faced with uncertain or missing market information. According to the above example, it also sheds light on the explanation facilities of payment agents. For example, the reason why settlement service A is considered as a candidate $Cand(A)$ is that there is high certainty of it being able to meet the processing deadline. Comparatively speaking, it is unlikely that A should not be a candidate service $\neg Cand(A)$. The reason is that there is low certainty for out of service (i.e.$Down(A)$), risk (i.e. $Risky(A)$), and high cost (i.e. $Expensive(A)$). Possibilistic logic provides an explicit representation of these causal relationships and a gradated assessment of the likelihood of these relationships given the available evidence.

6 Conclusions and Future Work

This paper proposes a theoretical framework for an agent-based electronic payment system. However, it is important to implement and evaluate such a framework. Particularly, effectiveness and computational efficiency of payment agents, which are underpinned by possibilistic logic, need to be examined further. One of the possible ways to conduct a quantitative evaluation for payment agents is to compare their performance (i.e. accuracy of decision, latency of making a decision) with other agents developed based on the traditional quantitative approaches such as multiple criteria decision making [11]. Initially, simulation of the agent-based electronic payment environment will be conducted in an Intranet environment. For example, several artificial sites can be set up as settlement services. Each of these services will be equipped with various service characteristics. By comparing the results of agents' decision making processes which are based

on possibilistic reasoning with that based on traditional quantitative approaches, the effectiveness of the possibilistic payment agents can be assessed.

The possibilistic resolution method illustrated in this paper is for non-fuzzy predicates. Since the payment agents also deal with fuzzy predicates such as $Expensive(x)$, further study is required to examine the feasibility of applying the possibilistic resolution method for fuzzy predicates [5] to the payment agents. Apart from intelligently choosing the optimum settlement service, a step further towards the development of payment agents is to allow them to negotiate with the settlement agents for a better deal (e.g. cheaper cost per transaction for larger amounts of payments). Whether it is technically feasible to integrate existing negotiation protocols with possibilistic reasoning in payment agents is another interesting research question.

The proposed agent-based electronic payment system sheds light on a formal approach towards electronic payment on the Internet. Because of the intrinsic dynamics of the Web-based electronic commerce, payment agents are faced with the challenge of making good decisions based on uncertain or incomplete market information. Possibilistic logic provides an expressive language to formalise uncertainties that are often associated with on-line transactions. Moreover, non-trivial possibilistic deduction allows a payment agent to make sound decisions based on the most certain and reliable information held in its knowledge base. The explicit modelling of the causal relationships between a decision and its justifications also facilitates the explanation of an agent's decision making process. From our initial study, it is technically feasible to apply possibilistic knowledge representation and reasoning to develop payment agents for electronic commerce.

References

1. D. Dubois, J. Lang, and H. Parade. Possibilistic Logic. In D. M. Gabbay, C. J. Hogger, J. A. Robinson, and D. Nute, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 439–513. Oxford University Press, Oxford, UK, 1993.
2. D. Dubois, J. Lang, and H. Prade. Theorem proving under uncertainty - a possibility theory based approach. In John McDermott, editor, *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 984–986, San Francisco, CA, August 1987. Morgan Kaufmann Publishers Inc.
3. D. Dubois, J. Lang, and H. Prade. A brief overview of possibilistic logic. In Rudolf Krause and Pierre Siegel, editors, *Proceedings of Symbolic and Quantitative Approaches to Uncertainty (ECSQAU '91)*, volume 548 of *LNCS*, pages 53–57, Berlin, Germany, October 1991. Springer.
4. D. Dubois, J. Lang, and H. Prade. Automated Reasoning using Possibilistic Logic: Semantics, Belief Revision, and Variable Certainty Weights. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):64–71, February 1994.
5. D. Dubois and H. Prade. Resolution principles in possibilistic logic. *International Journal of Approximate Reasoning*, 4(1):1–21, 1990.
6. W. Edwards. *Utility Theories: Measurements and Applications*. Kluwer Academic Publishers, Norwell, Massachusetts, 1992.

7. R. Glushko, J. Tenenbaum, and B. Meltzer. An XML Framework for Agent-Based E-commerce. *Communications of the ACM*, 42(3):106–114, March 1999.
8. R. Guttman and P. Maes. Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce. In *Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98)*, volume 1435 of *Lecture Notes in Artificial Intelligence*, pages 135–147, Berlin, 1998. Springer.
9. D. Hoffman, T. Novak, and M. Peralta. Building consumer trust online. *Communications of the ACM*, 42(4):80–85, April 1999.
10. S. Holtzman. *Intelligent Decision Systems.* Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.
11. R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* Cambridge University Press, Cambridge, UK, 1993.
12. S. Ketchpel, H. Garcia-Molina, and A. Paepcke. Shopping models: a flexible architecture for information commerce. In Robert B. Allen and Edie Rasmussen, editors, *Proceedings of the 2nd ACM International Conference on Digital Libraries*, pages 65–74, New York, July 23–26 1997. ACM Press.
13. G. Kraetzschmar. Distributed reason maintenance for multiagent systems. In *PhD Thesis, School of Engineering, University of Erlangen-Nürnberg, Erlangen, Germany*, volume 1229 of *Lecture Notes in Artificial Intelligence*, pages 1–261, Berlin, Germany, 1997. Springer.
14. R. Lau, A. H. M. Hofstede, and P. D. Bruza. Adaptive Profiling Agents for Electronic Commerce. In *Proceedings of the 4th CollECTeR Conference on Electronic Commerce*, Breckenridge, Colorado, April 2000.
15. G. Lohse and P. Spiller. Electronic shopping. *Communications of the ACM*, 41(7):81–87, July 1998.
16. P. Maes, R. Guttman, and A. Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–91, March 1999.
17. J. Müller and M. Pischel. Doing business in the information marketplace: A case study. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99)*, pages 139–146, New York, May 1–5, 1999. ACM Press.
18. P. Panurach. Money in electronic commerce:Digital cash, electronic fund transfer, and ecash. *Communications of the ACM*, 39(6):45–50, June 1996.
19. L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy sets and systems*, 1:3–28, 1978.
20. M. Zeleny. *Multiple Criteria Decision Making.* McGraw-Hill, New York, New York, 1982.

A Web-Based Negotiation Agent Using CBR

Dong Mei Zhang and Wai Yat Wong

CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde NSW 2113 Australia

Abstract. This paper presents a negotiation agent that applies Case-Based Reasoning (CBR) techniques to capture and re-use previously successful negotiation experiences in the multi-agent negotiation context. This negotiation agent provides adaptive negotiation strategies that can be generated dynamically and are context-sensitive. We demonstrate this negotiation agent in the context of used-car trading and a web-based application of negotiation agent for used car trading is developed. Relevant issues are discussed, including formulation of negotiation episodes, negotiation experience representation, negotiation experience matching/selection and previous negotiation experience adaptation/reuse.

1 Introduction

Electronic commerce systems are increasingly being developed to support trading on the Internet. Automated negotiation mechanisms play an integral and important role in these systems. In the trading environment, the multi-agent negotiation process can be described as a process of constructing an agreement based on a series of offer/counter-offers. The agents start with own goals that may be far apart and whose distance has to be narrowed gradually to zero, e.g. the buyer's price meets the seller's offer or vice versa, during the course of negotiation. Usually it involves a number of decision-making steps, i.e., negotiation episodes.

In real world negotiation, trading parts need to manage their own negotiation strategies during the whole negotiation process. This paper presents an automated negotiation agent that can provide adaptive negotiation strategies due to changing negotiation context. Previous negotiation experiences are represented and reused to suggest a new negotiation strategy in the used car trading context. Such an adaptive negotiation agent is based on acquiring and learning the negotiation strategies from previous negotiation experiences using Case-Based Reasoning (CBR) [6], [9] techniques.

Current EC trading systems which looks at electronic negotiation facilities usually use predefined and non-adaptive negotiation mechanisms [8]. Most systems use a number of predefined negotiation strategies to generate counter-offers [4]. They typically are based on non-adaptive strategies that are fixed for all offer-counter-offer episodes during negotiation. For example, negotiation in Kashah [8], [10] (MIT media Lab's) allows the agents to use three predefined strategies, anxious, cool-headed and

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 183–195, 2001.

frugal corresponding to a linear, quadratic and exponential functions in the generation of proposals/counter-proposals. The users need to decide which strategy the agents will follow during negotiation. A good automated negotiation mechanism should be able to deal with inherent complexity and changing world information of the real-word transactions. Researchers are now exploring various Artificial Intelligence based techniques to provide adaptive behavior to the negotiation engine. The use of Bayesian learning to learn negotiation strategy is one example [13], [18].

The paper first describes concepts used in the negotiation agent, negotiation episodes, episode strategy and negotiation protocol. It then presents the software framework of the web-based negotiation agent for used car trading. Issues of representation, selection and reuse of negotiation experience on car trading are respectively discussed. Sections 4 briefly describe the development environment for the negotiation agent software. Section 5 draws conclusions.

2 Negotiation Episode, Strategy, and Protocol

Three fundamental concepts in the negotiation agent are presented in this section, including negotiation episode, episode strategy and negotiation protocol. A negotiation episode represents a process unit in the negotiation, which consists of processes related to evaluation and generation of offers/counter-offers. In each episode, an episode strategy has to be defined for the generation of counter-offers. The negotiation protocol in our negotiation agent is characterized by defining adaptive episode strategies using CBR.

Negotiation Episode

Negotiation is an iterative process [14] where the trading agents start by having goals that may be far apart and whose distance has to be narrowed gradually. A successful negotiation occurs when the two opposing goals meet, e.g., when the buyer's price meets the seller's offer or vice versa. This negotiation process can be seen as a sequential decision making model studied by Cyert & DeGroot [3] and Bertsekas [1] and used in the Bazaar model by Zeng [18]. So, the negotiation process consists of a number of decision-making episodes. Each episode is characterized by evaluating an offer, determining an episodic strategy and generating a counteroffer, as illustrated in Fig 1.

Episode Strategy

In each negotiation episode, the negotiation agent needs to determine a strategy to be followed in the generation of counter-offers. Such a strategy is regarded as an episode strategy. The negotiation strategy of overall negotiation process is a composite of all these episode strategies. The negotiation agent can change its episode strategy from episode to episode due to changing world information. However, information related to strategies is hidden from each other. In the previous negotiation experiences, a series of offers/counter-offers reflects information related to episode strategies. The

Fig. 1. Negotiation episodes in the multi-agent negotiation.

changes from one offer/counter-offer to another offer/counter-offer show variation of episode strategies.

We use concession between offers/counter-offers to capture and reflect episode strategies [11]. Given a series of offers, (O1, O2, … , O5), the concession C(i+1) applied in the episode strategy S(i+1) is formulated as a percentage based on O(i) and O(i+1):

$$C(i+1) = [O(i+1) - O(i)] / O(i) * 100\%; \quad (1)$$

In this way, a series of concessions can be used to capture the implicit episode strategies used in the previous negotiation. The concession, as a percentage tries to represent context-independent information on episode strategies, which facilitates reuse of previous strategies in similar negotiation contexts.

Negotiation Protocol

The negotiation process consists of a number of negotiation episodes. In each episode an offer/counter-offer is generated based upon a certain strategy, which is viewed as an episode strategy as described above. Basically, the negotiation protocol in our negotiation agent involves the following actions:

- evaluating the offer from other agent,
- defining an adaptive concession by CBR for the current episode, and
- generating a counter-offer based on proposed concession.

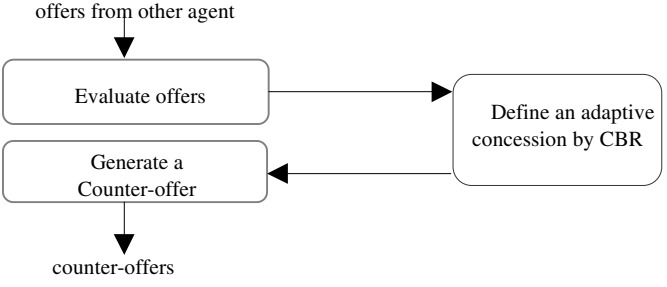


Fig. 2. Three actions in the negotiation protocol.

Fig. 2 shows actions in the negotiation protocol. Given an offer by other agent, if the negotiation agent evaluates it and decides not to accept it, the negotiation agent needs to determine what strategy (e.g., concession) to follow by reusing/adapting previous episode strategies. A counter-offer is then produced using the proposed concession.

3 A Web-Based Negotiation Agent for Used Car Trading

Fig 3 shows the framework of the web-based negotiation agent for our used car trading. This used car trading software prototype contains several functional components, as shown in Fig 3., a Case-Based Negotiator, a Negotiation Case Browser, a Descriptive Statistics component and a Case Maintenance component:

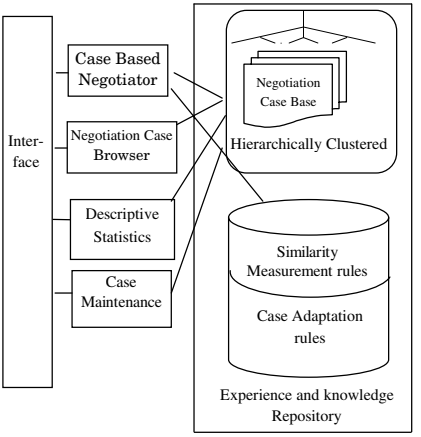


Fig. 3. Framework of the web-based negotiation agent for used car trading.

- The Case-Based Negotiator (CBN) assists the users to negotiate with opponent agents on used car trading price. It evaluates the offer from the opponent agent and an provides an appropriate concession to be used in the generation of an counter-offer if the offer is rejected. To determine the appropriate concession, it matches current negotiating scenario with previous successful negotiation experiences, selects best-matched negotiation experience and adapts previous concession to the current negotiation context. Resultant successful negotiation can be automatically updated into the experience and knowledge repository.
- The Negotiation Case Browser allows users to browse a previous negotiation case repository using various types of queries.
- The Descriptive Statistics component supplies several useful descriptive statistics on negotiation case repository.
- The Case Maintenance component allows negotiation experts to moderate, maintain and to update the case repository. To condense potential great number of similar stored cases, generalization based /concept learning techniques can be

used to induce generalized case. This will allow efficient, faster and perhaps better case matching and reuse in the future.

The prototype is initiated by a user's choice of functional components. For negotiation function, the user starts by selecting choice of used car to buy/sell. The user then input profile information, e.g. negotiation focus in terms of 'good- deal', 'best-price', 'must-buy/sell', budget, gender, age, etc. the case-based negotiator then starts negotiation with the opponent agent.

A number of previous negotiation experiences on car trading is stored and represented in the experience and knowledge repository, which provides detailed negotiation context and decisions made in previous negotiations. A hierarchy is used as an organizational structure for storing negotiation experiences, which enables an efficient searching through cases. It enables the retrieval of relevant cases to the given negotiation information. A set of measurement rules and adaptation rules is also represented in the experience and knowledge repository. The measurement rules are applied to measure the similarities between retrieved cases and the given negotiation information. It is based on rule-based reasoning approach. Adaptation rules are applied in reusing and adapting previous cases.

The rest of the section focuses on the negotiation process in the Case-Based Negotiator and presents our approaches to representation of previous used car negotiation experiences and generation of episode concessions

3.1 Representation of Car-Trading Experiences

Previous car-trading experiences are represented as negotiation cases in the software. A negotiation case for a car trading scenario represents information related to a specific car trading agent (e.g., sell or buyer) in a previous negotiation and captures contextual information and negotiation experience available to the agent. A negotiation case for car trading can contain the following information:

- Car buyer's profile,
- Car seller's profile,
- Used car's profile,
- Offers made from opponent agent and concessions used in episode strategies,
- Counter-offers made by the agent and concessions used in episode strategies,
- Performance information about the feedback of negotiation results.

The negotiation context for car trading is defined by profiles of the car buyer, the car seller and the user car, which provides the following information:

1. Car seller's and car buyer's profiles including name, age, buyer/seller's desire, expected negotiation duration, issues of negotiation (e.g. price, warranty, trade-in, etc), constraints (e.g., budget) and preferences.
2. Profile of negotiated used-car including car-size, car-make, car-age.

The negotiation experiences on car trading can be captured by the received offers, the generated counter-offers and the concessions used in each negotiation episode. The performance information gives description of outcomes of negotiations such as success/failure, number of episodes and the final mutually agreed terms of a deal. The following Fig. 4 summarizes contents for a car buyer agent's case.

Buyer's profile		Seller's profile		Profile of used-car
Episode No.	**Seller's offer**	**Seller's concession**	**Buyer's counter-offer**	**Buyer's concession**
1	O-1		CO-1	
2	O-2	S%-1	CO-2	B%-1
3	O-3	S%-2	CO-3	B%-2
4	O-4	S%-3	CO-4	B%-3
...				
Performance information				

Fig. 4. A summary of a car buyer agent's case.

In the software, the contents of a negotiation case for car trading can be displayed, including information on car-trading context, performance and concessions and offers/counter-offers. Fig. 5 gives a screen display for car buyer's Case'5' in the negotiation software.

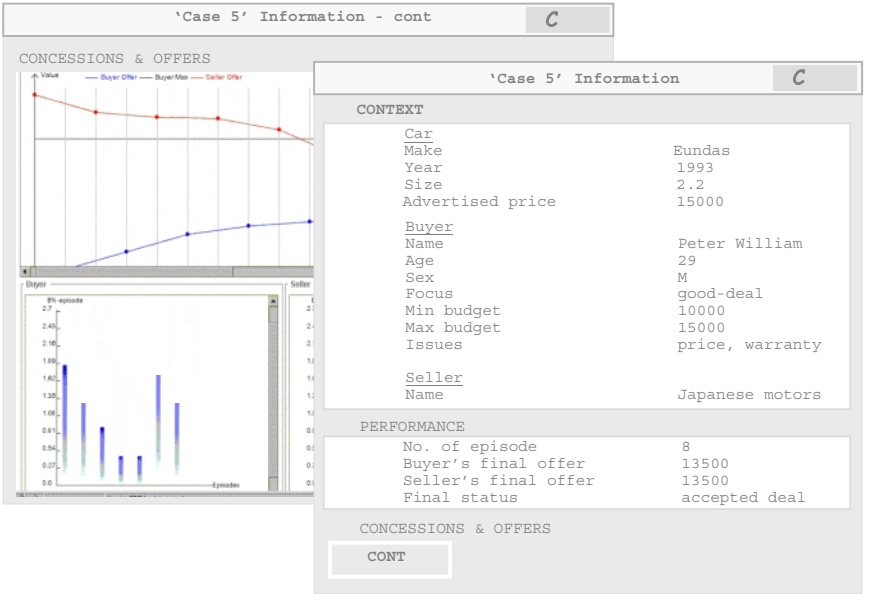


Fig. 5. A screen display for car buyer's 'Case 5' in the software.

3.2 Generation of Episode Concessions

The case-based negotiator generates a proper concession for a negotiation episode based on retrieval, selection and reuse of relevant previous experience. As shown in Fig. 6, this process is composed of three main processes:

- retrieving relevant previous negotiation experience cases,
- matching/selecting the most matched case and
- reusing the strategy information in the selected negotiation experience case.

In each negotiation episode, the case based negotiator has the information of the current profiles of the negotiating agents, current car profile, and current most up-to-date episode-history-lists of offers & corresponding concessions and counter-offers & corresponding concessions. Based on the current profiles of the agents and car, the negotiator first retrieves a set of previous negotiation cases using the contextual case organization hierarchy. It then performs similarity assessment to select the most similar cases from this group of relevant cases using information from all profiles and the current episode-history lists of offers, concessions, counter-offers, & counter-concessions. Information from the most similar case will be re-used to adapt the next concession used to generate the next offer. This sub-section will describe briefly the retrieval process, selection using the similarity measures and the reuse of the best-matched case(s) from the viewpoint of a buyer agent.

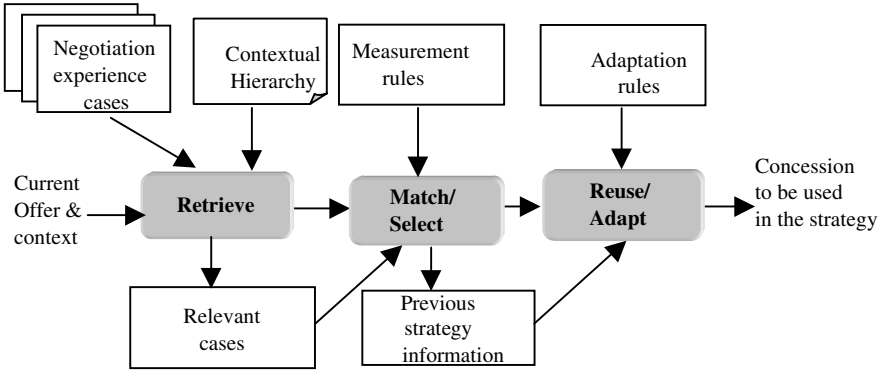


Fig. 6. The processes of defining a concession by CBR.

3.2.1 Retrieving Negotiation Experiences

A contextual hierarchy is used as an organization structure for grouping and categorizing the car trading cases. A number of salient features are selected to index cases based on their possible values. For example, because we believe that different negotiation behavior arises primarily from the negotiation focus of the buying agent, the feature “focus” in the buyer agent’s profile are used as primary index for different groups of buyer negotiation cases. The possible values of "focus" include “must-buy”, “good-deal” and “best-price”. The feature "focus" thus groups cases into three catego-

ries. Other salient features used for further sub-categorization include the age and the engine-size of the car. The Case organizational hierarchy is used to is used to retrieve relevant negotiation cases by searching through case memory. If no case is found based on the case organization hierarchy, the buyer agent can fall back using some default negotiation strategies selected by the user from a set of pre-defined strategies (e.g. linear, quadratic or exponential).

Fig. 7 shows a contextual hierarchy for car buyer agent's cases. Three contextual features, buyer's focus, car's age and car's size are selected and used to index cases based on their values. At the top of hierarchy, values of the feature "buyer's focus" in the buyer's profile are used as indices for different groups of cases. For each 'buyer's focus' branch, other features (car-age and car-size) are used to further classify the cases. At the bottom, a group buyer's cases, e.g., C1, C5, C7, … is indexed by three features with three particular values, e.g., buyer's focus: good-deal; car-age: <=4 years and car-size: <= 2 liter.

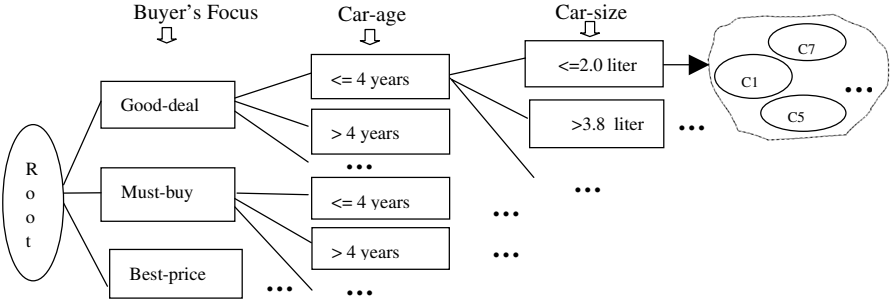


Fig. 7. The contextual hierarchy of buyer's cases.

3.2.2 Selecting Similar Negotiation Experience

Similarity measurement rules are used to select out the best-matched case from the retrieved relevant set of cases. The rules are applied to all relevant cases retrieved based on contextual hierarchy. We illustrate here the "good-deal" scenario and will show briefly the difference of "best-price" and "must-buy" scenarios.

In the "good-deal" scenario, all the retrieved relevant cases are passed first through a concession-match filter. The concession-match tries to find "sub-string" matches between the concessions & counter-concessions of the previous negotiation cases and the concessions & counter-concessions of the current negotiation process (Fig. 8).

Concession matching is done based on measurement strategies with some allowance of tolerance. It enables partial matching of concession sequences between cases and input information, for example, in the above example, the concession sequence "10, 5, 5" is a partial matching with input concessions "9.7, 5.2, 4.8" based on a tolerance of certain percentage of difference. This tolerance is regarded as an adjustable threshold.

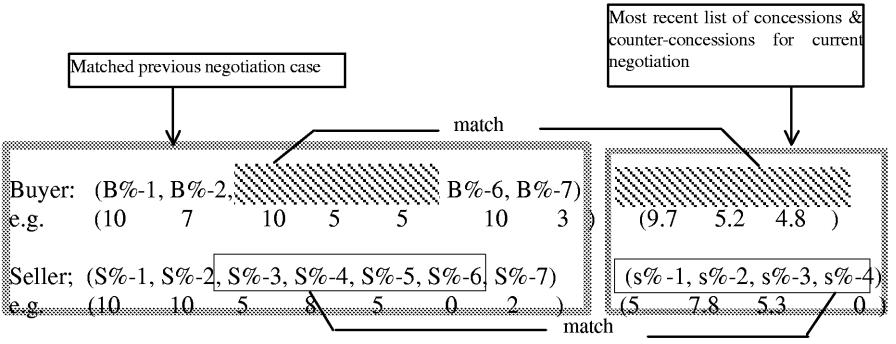


Fig. 8. The concession matching in the case-based negotiator.

When more than one case are selected based on concession matching, the further measurement is done based on the profile information. Dependent on the number of the filtered cases, they are then passed through an ordered series of profile matching filters. A series of profile matching filters is applied, including the starting-offer-match filter, starting-position-match filter, profile-age-match filter, and profile-gender-match filter.

In our current prototype implementation, two quantities match if their difference falls within either a percentage tolerance threshold or absolute tolerance threshold. As described earlier, negotiation strategy for a case is reflected by the overall composite of the episodic concessions/counter-concessions during the negotiation process. The concession-match thus captures the negotiation processes/cases with the same strategies or sub-strategies, i.e. they are of the similar functional/sub-functional and behavior/sub-behavior forms. The starting-offer gives preference to cases with similar initial buying offer. This filter thus restricts the magnitude difference of those cases with similar strategies or sub-strategies, i.e. this filter gives preference to negotiation cases/sub-cases which are similar in offers/counter-offers – a stronger requirement. The starting-position match filter gives preference to the previous case whose concession matches from the beginning. This filter thus restricts the translational difference. It throws away the sub-strategy functional match and sub-case match and prefers full-strategy/ full-case match. The age and gender filters respectively match cases with the same age range and gender. They are used to put some preference on the matching cases if the two restrictive (starting-offer-match and starting-position-match) do not produce any matching candidates. Because of the noisy nature of the cases, those filters are arranged in such manner to give most opportunity of getting some best-matched case(s).

For “Best-price” and “Must-buy” scenarios, some heuristics are also developed for case similarity assessment. Details of description is given in [19]. For example, in the “Must buy” scenario, there is an addition rule before the concession-match filter and profile match filters which says:

If the opponent agent’s offer reaches a plateau and offer is within the agent’s budget, suggest a concession that can reach the opponent agent’s offer.

3.2.3 Reusing Previous Concessions

Once a case is selected as the best matched case to the current scenario, the negotiator takes the next concession that was used in previous negotiation and suggests a concession for the generation of a counter-offer. For example, in Fig. 8, the concession "B%-6" will be reused to recommend to the negotiation agent to follow for generating a counter-offer.

Some adaptation is done during the boundary conditions [17], [19], e.g. if the concession results in a counter-offer which is higher than the buyer's maximum budget, then the concession must be adjusted accordingly to arrive only at the counter-offer equal to or not greater than the buyer's maximum budget. During a negotiation episode in the negotiation process, the case base negotiator retrieves relevant cases and adapts the concession from the best-matched negotiation case to generate a counter-offer. The dynamics of the negotiation process will be tracked and displayed on the user screen in the prototype, as shown in Fig 9.

In the course of negotiation, when the offers from buyer and seller meet, the user makes the final decision whether the deal is done. Fig 9 shows the scenario where a potential deal is identified. A pop-up window then provides the user with the options to accept or reject the deal. If the final offer is accepted, the case maintenance component of the prototype updates the case base by adding relevant information to the current negotiation, including agent's profile,

4 Development of Web-Based Negotiation Agent

A web-based system has been developed as a negotiation agent for used car trading. The software is designed as a distributed three-tier server client architecture [16]: Negotiator client, Negotiator server, and Negotiation Case Repository, as shown in Fig. 9.

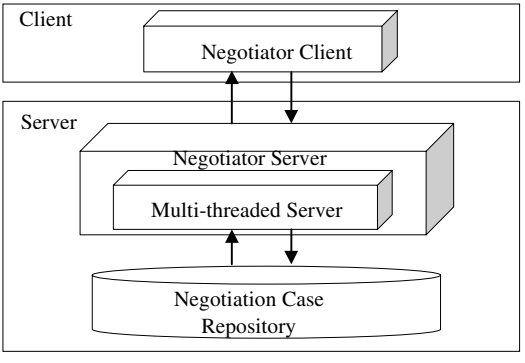


Fig. 9. The negotiation agent software client-server architecture.

Java 1.2 is the programming language used for implementation. The Negotiation Case Repository manages all the stored negotiation cases. The Case Repository is imple-

mented via a standard SQL capable relational database management system. All the cases are captured within several relational tables in a database.

The Negotiator server is the reasoning engine that performs processes of Case-based Negotiator including offer evaluation, case retrieval, case matching and reuse as well counter-offer generation. For example, it applies various similarity assessments to find the best matched case from a pool of relevant cases retrieved from the Case Base and adapts best-matched case to obtain episodic strategy to generate counter-offer. It is also the data processing engine for case browsing and descriptive statistical reports. Using specific communication protocol via socket connection, the negotiator server accepts input from the Negotiator client and generates output to the client. It also accesses the Negotiation Case Base directly using SQL queries via Java Database Connectivity (JDBC) API.

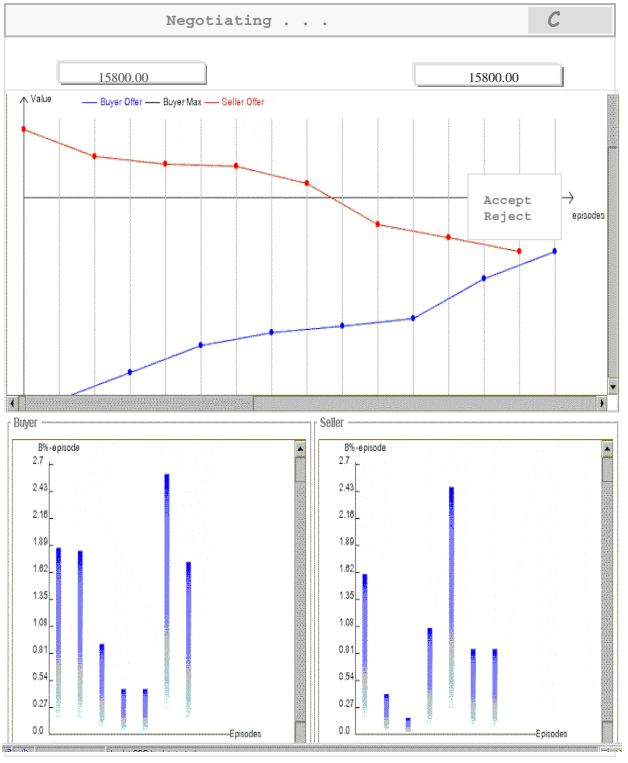


Fig. 10. Tracking negotiation process in the Case-Based Negotiator (CBN).

The negotiator client provides the graphical user interface and socket connection to server via internal communication protocol. The graphical user interface provides data collection, data representation and display facilities. The graphical user interface is implemented using the Swing API. When designing this server-client system, we have emphasized lessening processing load on the client-side. This is to compensate the extensive processing the client has to do to maintain the interactive user interface.

The dynamics of the negotiation process can be tracked and displayed on the user screen. The system shows the decisions made from both seller and buyer during the negotiation. Fig. 10 gives a screen of negotiation system, which captures offers and concessions for every episode. A potential deal is identified. A pop-up window then provides the user with two options: 'accept' or 'reject'. The user makes the final decision. At the end of negotiation, the system presents summary information of the negotiation, including number of episodes in the negotiation., buyer's final offer, seller's offer, etc.

5 Conclusions

To deal with the changing world information, the negotiation agent needs to revise and adapt its negotiation strategies from session to session during negotiation. This paper has described a research project for providing adaptive negotiation strategies for the negotiation agents. A web-based negotiation agent framework is proposed, which is characterized by its capability of learning from experiences using CBR techniques. Such an approach can help the negotiation process improve its performance by providing adaptive contextually appropriate strategies. The current negotiation agent is developed for single-issue (e.g., car-price) negotiation. An assumption is made that all cases provide successful negotiation experiences. Learning from failure is not considered yet.

The proposed negotiation agent uses CBR to learn from its experiences, which is similar to Sycara's intelligent computer-supported conflict resolution program in PERSUADER [15]. In Sycara's approach, the intelligent negotiation support program is designed in a form of mediation, which uses CBR and decision theoretic techniques to provide the enhanced conflict resolution and negotiation support in group decision problem solving. The support program itself evaluates and generates potential agreements and then proposes them to involved negotiation agents. The negotiation goals and context of both parts are well known to the negotiation support program. The negotiation agent framework is characterized by its negotiation protocol. The evaluation of offers and generation of counter-offers are performed by the negotiation agents with ability of acquiring and learning strategies from previous negotiation experiences. Each agent needs to make decisions in the situation where other agents' negotiation goals are not revealed.

We believe strongly that one effective way to capture the utility functions is to examine the previous behaviors of the agents. In our case, we looks at all successful negotiation experiences of the buyer and seller agents and assume those collective experiences indicate well the inherent utility functions of a particular type of agent under a particular type of circumstances. We then adapt and apply the result to current negotiation situation. In doing so, we avoid explicitly specifying the utility functions by assigning arbitrary values. Moreover, our case-based approach can allow us to accommodate easily the change/growth of the utility functions e.g., through time.

We are currently examining the effectiveness of this negotiation agent framework in the car training domain [2] to produce better and efficient negotiation. The proposed negotiation framework is being improved in several aspects. First, in the current

scope, only a single issue, namely car-price forms the core of our negotiation, more research effort will be made to extend the negotiation agent to multi-issue negotiation. Second, it is being considered to reuse failure negotiation experiences to avoid the same mistakes to take place in new negotiations. Third, more sophisticated approach for similarity measurement will largely improve effectiveness and efficiency of negotiation process.

References

1. Bertsekas, D. P., Dynamic Programming and Optimal Contral. Athena Scientific, Belmont, MA 1995.
2. Bui, V. Intelligent trading agents. CSIRO CMIS Internal report, 1999.
3. Cyert, R. M. and DeGroot, M. H., Bayesian Analysis and Uncertainty in Economic Theory, Rowman & Littlefield, New York, 1987.
4. Guttman R., Moukas A. and Maes P. Agent-mediated Electronic Commerce: A survey. In Knowledge Engineering review, June 1998.
5. Kasbah (http://asbah.media.mit.edu)
6. Kolodner, J, Case-Based Reasoning. Morgan Kaufmann Publishers (1993).
7. Kowalczyk, R. and Bui, V. "Towards Intelligent Trading Agents", the 1999 International Conference on Intelligent Systems and Active DSS, Turku, Finland
8. Maes, P., Guttman, R. H., and Moukas, A. G. Agents that buy and sell". Communications of The ACM. Vol. 42. No.3 (1999) pp. 81-91.
9. Maher, M. L., Balachandran, M.B. and Zhang, D. M. *Case-Based Reasoning in Design*, Lawrence Erlbaum: New Jersey. (1995)
10. Nwana H., Rosenschein J., Sandholm T., Sierra C., Maes P. and Guttmann R. Agent-Mediated Electronic commerce: Issues, Challenges and some Viewpoints. 2rd International Conference on Autonomous Agents. Minneapolis, May 1998.
11. Raiffa H. The art and Science of negotiation. Harvard Univ. Press (1982).
12. Rosenschein J. and Zlotkin G. Rules of Encounter: Designing Conventions for Automated Negotiation among Computers. MIT Press, 1994.
13. Siriwan, A and Sadananda R. An agent-mediated Negotiation Model in Electronic Commerce, The Australian Workshop on AI in Electronic Commerce, The Australian Joint Conference on Artificial Intelligence. 1999.
14. Sycara, K., Utility theory in conflict resolution. Annals of OR, Vol. 12 (1988) pp. 65-84.
15. Sycara, K Machine learning for intelligent support of conflict resolution, Decision Support systems, Vol. 10 (1993) pp. 121-136.
16. Tete-a-Tete (http://ecommerce.media.mit.edu/tete-a-tete/)
17. Wong, W., Zhang, D.M. and Kara-Ali, M. Towards an experience based negotiation agent. Accepted by the Fourth International workshop on Cooperative Informtaion Agents (CIA 2000), Boston, USA.
18. Zeng, D and Sycara, K. Bayesian Learning in Negotiation, Int. J. Human-Computer Studies 48, pp 125-141, 1998.
19. Zhang, D.M., Wong, W. and Kowalczyk, R. Reusing previous negotiation experiences in multi-agent negotiation, Proceedings of WAEC'99. pp 79-87. HongKong. 1999.

Intelligent Information Agents

Seng Wai Loke

School of Computer Science and Information Technology
RMIT University
Melbourne, Australia
swloke@cs.rmit.edu.au

1 Introduction

With the unprecedented abundance of digital information available on the Web and the Internet, there is an increasing need for intelligent software assistants to effectively manage such information and transform them into useful knowledge in the right form for the right person at the right time and situation. Indeed, intelligent information agents find application in almost every facet of life including business, education, commerce, industry, and entertainment. Developing such software is non-trivial and spans diverse areas such as artificial intelligence, distributed systems, knowledge management, programming languages and information retrieval.

PRIIA 2000 is a workshop on intelligent information agents held on 29 August 2000 in Melbourne, Australia, in conjunction with PRICAI 2000. PRIIA is a forum for intelligent information agent researchers in Pacific Rim countries to exchange and discuss their research results. We received an interesting collection of papers, the accepted papers of which can be roughly categorized into two streams: 1. The Role of Reasoning in Information Agents, and 2. Web-Based Information Agents in the Future.

In the first stream, Raymond et al. present a logical basis for agents to revise their beliefs about their users' changing information needs. Kriaa et al. and Zhang et al. present agent-based methods to reason about and analyse complex systems. Each provided an example, a system for emergency health care and a system for product life cycle cost prediction, respectively. Heinze et al. propose the perception module as a key component in designing and building agent systems.

Papers in the second stream describe four different technologies, all useful for information agents of the future: 3D visualization of Web structure (Hiraishi et al.), strong migration of logic programming based mobile agents to enable backtracking to continue across different hosts (Fukuta et al.), constructing agents by constructing knowledge bases (Gao et al.), and user profile management for the wireless environment using agents (Pils et al.). We have also an invited paper from Harold Boley which discusses mappings between RDF (W3C's recommended standard for structured information) and logic programming, and visualization of RDF descriptions as hypergraphs. We would like to thank all

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 199–200, 2001.

the authors who submitted papers to the workshop and participated in the interesting discussions at the workshop. We would also like to thank the program committee.

2 Program Committee

Executive Committee

Workshop Chairs:	Leon Sterling (Australia) Fumio Mizoguchi (Japan)
Program Chair:	Seng Wai Loke (Australia)

Program Committee

W. Wen (Japan)
H. Ohwada (Japan)
X. Gao (New Zealand)

Relationships between Logic Programming and RDF*

Harold Boley

DFKI GmbH
boley@dfki.de

Abstract. Mutual relationships between logic programming and RDF are examined. Basic RDF is formalized with ground binary Datalog Horn facts. Containers are modeled using ('active') polyadic constructors. For meta-statements a modal-logic treatment is suggested. To reduce large fact sets, an "inferential RDF", with special Horn rules, is introduced. A direct representation of non-binary relations is proposed. Reification is thus abandonned and RDF diagrams are generalized using hypergraphs. RDF types are considered as sort predicates. RDF Schema's class/property hierarchies are regarded as a second-order subsumes/subsumes2 syntax or as simple Horn rules. Its domain/range constraints are extended to (polymorphic) signatures. All concepts are explained via knowledge-representation examples as usable by information agents.

1 Introduction

The Resource Description Framework (RDF) [LS99,BG00] has been standardized by the W3C mainly to capture metadata about arbitrary resources addressed by URIs/URLs. It is thus intended for encoding a **resource index**, as e.g. needed for information localization by semantic search engines and for information interoperation by broker agents. However, since the boundary between metadata and data is not an absolute one, RDF can also be used for encoding a **knowledge base**, as e.g. needed for question answering. Moreover, one can use an RDF-like encoding as a combination of a knowledge base, e.g. answering some questions directly, and a resource index, e.g. finding further helpful documents [Bol99]. RDF may be regarded as an initial language for the *Semantic Web* [BLF99] [http://www.w3.org/2001/sw/].

RDF in the narrow sense [LS99] [http://www.w3.org/TR/REC-rdf-syntax/] consists of a directed-labeled-graph or triple-set model with full and abridged XML serialization syntaxes. It is broadened by RDF Schema

* I would like to thank Leon Sterling, Seng Wai Loke, and the programm committee of the 1st Pacific Rim International Workshop on Intelligent Information Agents for inviting me to this article. It is based on preparatory research done for the EU project CLOCKWORK. Special thanks for valuable RDF discussions and proof-reading go to Michael Sintek.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 201–218, 2001.

[BG00] [http://www.w3.org/TR/rdf-schema/], itself specified in RDF syntax, for (pre)categorizing RDF nodes and arc labels. RDF was thus influenced by work in knowledge representation (KR) such as semantic nets, frame systems, and logic languages.

It may therefore be worthwhile to again look at RDF from a KR perspective. Here, we focus on relationships between RDF/XML and logic programming (LP); the companion papers [http://www.dfki.uni-kl.de/~boley/xmlp-engl.ps] and [Bol01] deal with XML-LP comparisons. To cover the diagrammatic aspects of (extended) RDF, we will employ (hyper)graphs as discussed for KR in [Bol92]: While the current RDF standard – in the tradition of simple semantic nets – focusses on binary relations visualized as arcs, this paper moves on towards a more direct treatment of non-binary relations, demonstrating their direct visualization as hyperarcs (cf. appendix A).

More generally, we will show trade-offs between the simplicity of the representation language – e.g. RDF's binary Horn facts – and the complexity of representation constructions in such a language – e.g. RDF's need for reification to represent containers, meta-statements, Horn rules, and N-ary relations. For RDF Schema, we will discuss first-order reductions of the (useful) second-order syntactic sugar of class/property hierarchies and extend domain/range constraints to polymorphic signatures.

2 Basic RDF Becomes Simple Horn Facts

Let us start with a version of the **informal** XML sample sentence from section 2 in [Bol01]:

`<sentence>` Onoffbook sold 12417 copies of XML4You online `</sentence>`

As in that paper we can proceed to its **semiformal** XML `triple` element representation:

```
<triple>
  <subject> Onoffbook </subject>
  <predicate> sold online </predicate>
  <object> 12417 copies of XML4You </object>
</triple>
```

But now let us assume the bookstore Onoffbook and the book XML4You are considered as 'resources' represented by their URIs/URLs. Then, as a new third stage, we obtain the **formal** XML element consisting of a `graph` of `triple` subelements:[1]

[1] The third triple, briefly `("http://www.onoffbook.com/bookId/00498", quantity, 12417)`, stands for "the number of `"http://www.onoffbook.com/bookId/00498"` items sold is 12417". This would be clearer if combined with the first triple into a single statement, foreshadowing the need for non-binary predicates as discussed in section 6.

```
<graph>
  <triple>
    <subject resource="http://www.onoffbook.com"/>
    <predicate>online-sales</predicate>
    <object resource="http://www.onoffbook.com/bookId/00498"/>
  </triple>
  <triple>
    <subject resource="http://www.onoffbook.com/bookId/00498"/>
    <predicate>name</predicate>
    <object>XML4You</object>
  </triple>
  <triple>
    <subject resource="http://www.onoffbook.com/bookId/00498"/>
    <predicate>quantity</predicate>
    <object>12417</object>
  </triple>
</graph>
```

Resources (in RDF always in the subject position and sometimes in the object position) are represented by a URL attribute of empty elements. Literals (in RDF sometimes in the object position) become ordinary (PCDATA-)content elements.

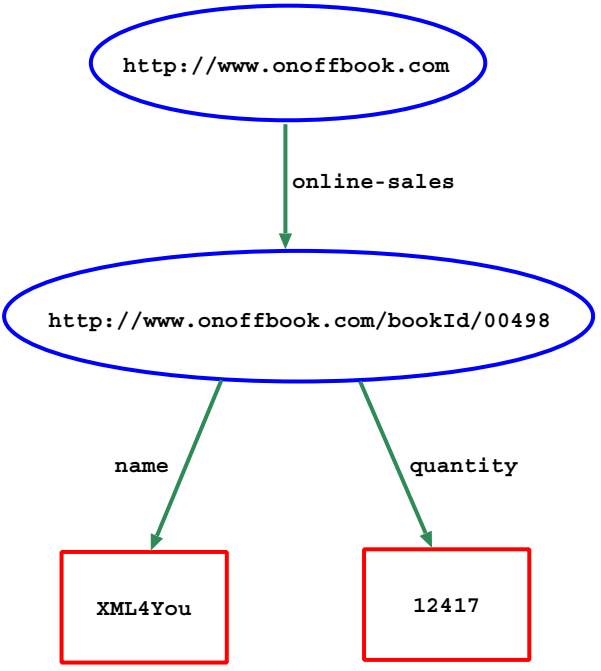
In general, a `graph` element is an XML represention of a directed labeled graph (DLG) via a set of triples, {..., (source-node, arc-label, target-node), ...}, marked up as

```
<graph>
  ...
  <triple> source-node' arc-label' target-node' </triple>
  ...
</graph>
```

where the primes indicate further source (subject) and target (object) node as well as label (predicate) markup.

Our example corresponds to the following DLG diagram, which already happens to be an RDF diagram:[2]

[2] Because of other diagram features such as the different shapes for resources and literals, the colors of the online version will not be essential here and later on.



Its above XML `triple-graph` serialization becomes full (unabridged) RDF/XML via a join of triples having the same subject and an attribute renaming in the subject position (`resource` into `about`) together with some further element restructuring/renaming (using XML's *namespace*:*localname* tags):

```
<rdf:RDF>
  <rdf:Description about="http://www.onoffbook.com">
    <s:online-sales
      rdf:resource="http://www.onoffbook.com/bookId/00498"/>
  </rdf:Description>
  <rdf:Description about="http://www.onoffbook.com/bookId/00498">
    <v:name>XML4You</v:name>
    <v:quantity>12417</v:quantity>
  </rdf:Description>
</rdf:RDF>
```

In LP, the earlier triple form can be compactly rewritten as binary relations over URI/URL and ordinary constants, which (when enriched by XML namespaces) also capture the RDF/XML version:

```
online-sales("http://www.onoffbook.com",
             "http://www.onoffbook.com/bookId/00498").
name("http://www.onoffbook.com/bookId/00498",xml4you).
quantity("http://www.onoffbook.com/bookId/00498",12417).
```

Since each such RDF re-representation leads to a finite set of ground Datalog Horn facts, this also illustrates how the RDF model, which is a *data model*, can be semantically construed as simple *Herbrand models*, in the sense of model theory [Llo87] (giving up the *unique name assumption* for multiple-URL individuals): A set of ground facts like the above actually is its own Herbrand model.

Of course, these three RDF/XML-derived Prolog facts could again be represented as XML elements, e.g. in the manner of section 4 in [Bol01] or as in RuleML [http://www.dfki.de/ruleml]. Since the subject of an RDF statement must be a resource, the first argument of each such fact must be a URI/URL; in RuleML we thus specified a "UR-centered" DTD for RDF-like languages [http://www.dfki.de/ruleml/inrdf.html].

3 Containers Lead to Generalized Structures

In RDF containers are treated by a kind of reification, with a new anonymous resource node standing for a container as a whole, and special arcs leading to its elements. In LP, we can avoid these 'new nodes' by using generalized Prolog structures applying a polyadic (N-ary, $N \geq 0$) and possibly 'active' constructor to the N container elements. The corresponding diagram versions will similarly avoid reification by using *hyperarcs*, connecting N nodes [Bol92] (cf. appendix A).

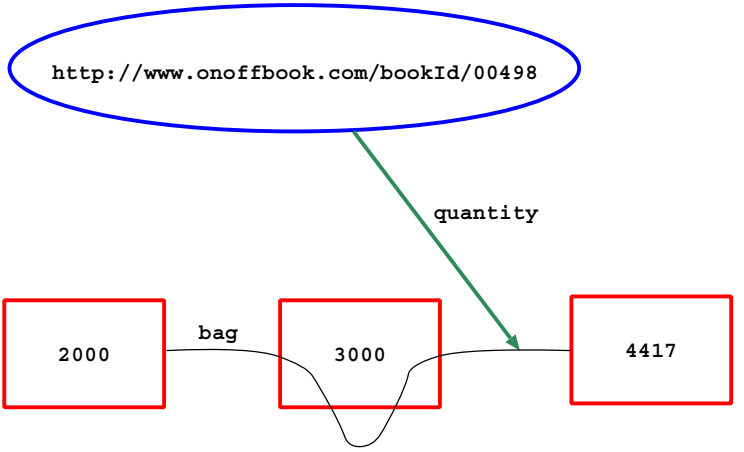
Sequence just becomes the list (or tuple) datatype of N ordered elements with repetitions, i.e. a `tup` constructor applied to N arguments. This may be diagrammed as a directed `tup`-labeled hyperarc similar to unasserted arcs in section 4.

Bag becomes a multiset datatype of N unordered elements with repetitions. It employs an N-ary constructor, `bag`, that disregards the argument order, e.g. by sorting ground arguments lexicographically (and using AC-unification for non-ground arguments). This may be diagrammed as an undirected `bag`-labeled hyperarc or as a complex `bag`-labeled node containing labeled nodes. While both diagram forms can cope with the duplicates permitted within bags we here prefer undirected hyperarcs in order to avoid labeled nodes.

In our example suppose the `quantity` is split into a sum of four unspecified (temporal or regional) subquantities, where two or more subquantities may be identical. This leads to a `bag` structure like `bag[2000,3000,3000,4417]` usable in facts such as

```
quantity("http://www.onoffbook.com/bookId/00498",
         bag[2000,3000,3000,4417]).
```

It is diagrammed as the following undirected (no arrow head) `bag`-labeled hyperarc with a duplicate `3000` node (cut twice by the hyperarc):

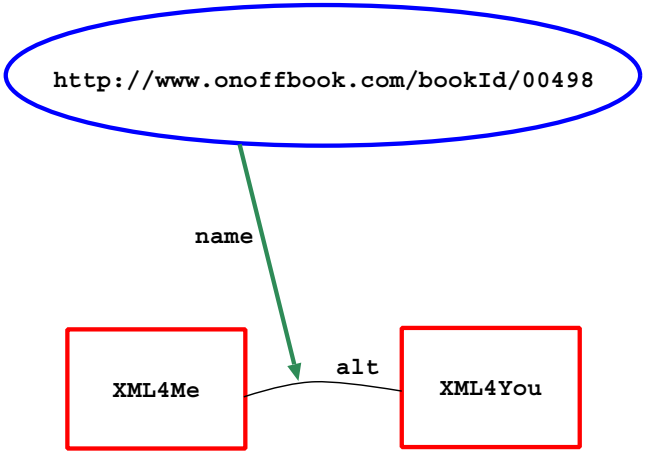


Alternative may be similarly formalized as a kind of disjunctive datatype and diagrammed as an undirected `alt`-labeled hyperarc or as a complex `alt`-labeled node containing unlabeled nodes. For uniformity we again prefer undirected hyperarcs.

In our example suppose the `name` has two alternatives, `XML4You` and `XML4Me`. This leads to the `alt` structure `alt[xml4me,xml4you]` usable in facts such as

```
name("http://www.onoffbook.com/bookId/00498",alt[xml4me,xml4you]).
```

This is diagrammed as the following `alt`-labeled undirected hyperarc (which happens to be an arc):



However, if we want `alt` to 'distribute out of' certain contexts, 'absorb' failures in its arguments, or exhibit similar behavior, it begins to look more like a control structure than like a data structure. In this view `alt` may exhibit Prolog's

notion of *"don't know" non-determinism* or committed-choice languages' notion of *"don't care" non-determinism*, where the latter, search-free non-determinism may be preferable in the open Web environment of RDF.

RDF's *distributive referents* use an `aboutEach` attribute for property 'distribution into' containers. They could be represented in LP by reinterpreting `aboutEach` as a universal quantifier over structures.

4 Meta-statements via Modal Operators

RDF meta-statements (statements about statements) are based on *reified statements* similar to our original `triple` formalization in section 2, with an additional property specifying that the described new resource has an `rdf:type` of `Statement`. Such a reified statement can then become the target of another statement using a further property, `a:attributedTo`. For example, we could specify that Onoffbook themselves make the `online-sales` claim as follows:

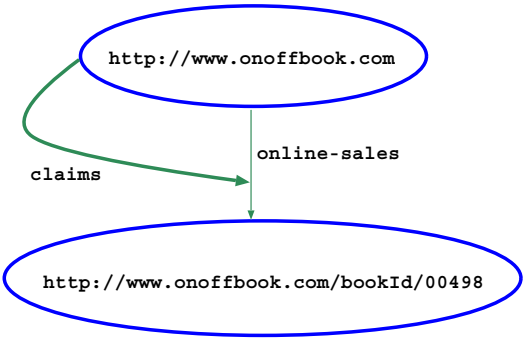
```
<rdf:RDF>
  <rdf:Description>
    <rdf:subject resource="http://www.onoffbook.com"/>
    <rdf:predicate resource=
                    "http://description.org/schema/online-sales"/>
    <rdf:object resource="http://www.onoffbook.com/bookId/00498"/>
    <rdf:type resource=
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
    <a:attributedTo resource="http://www.onoffbook.com"/>
  </rdf:Description>
</rdf:RDF>
```

However, a modal-logic treatment as indicated in section 5 of [Bol01] with a `belief` operator would more directly capture the intended semantics. Here we similarly employ a `claims` statement – corresponding to the inverse of `a:attributedTo` – about an `online-sales` statement:

```
claims("http://www.onoffbook.com",
       online-sales("http://www.onoffbook.com",
                    "http://www.onoffbook.com/bookId/00498")).
```

The DLG representation can be accommodated to such modal statements as follows: I. Allow nodes and node-connecting (level 1) arcs as the source and/or target of (level 2) arcs, allow all of these in (level 3) arcs, etc., to any level of modal nesting. II. Introduce two weights for arcs, where heavy arrows denote asserted arcs (expressing true statements) while light arrows denote unasserted arcs (needed to express other statements).

In the example, the heavy `claims` (level 2) arc has the `"http://www.onoffbook.com"` node as the source and the light `online-sales` (level 1) arc from there to the `"http://www.onoffbook.com/bookId/00498"` node as the target:



In this DLG extension it is easy to switch a given arc between light and heavy. Thus, after verifying Onoffbook's sales claim, we need just toggle the light arc to heavy for asserting both Onoffbook's claim and its truth (lightness and heaviness of arcs do not affect the accessing higher-level arcs). In LP, this requires an additional copy of the embedded `online-sales` as the fact from section 2. In RDF it even requires a completely differently represented non-reified statement. Reified and non-reified statement versions may be confusing since the epistemological status of 'unused' reified statements is unclear. Here we regard a top-level light arc as a spurious construction, which should lead to user notification calling for either top-level deletion or heavy-making, where the graph that pops up after top-level deletion will be re-checked recursively.

5 Inferential RDF through Horn Rules

Suppose we want to use the RDF/XML example from section 2 – enriched by price information (cf. section 6) etc. – as metadata to describe some other data such as bookshop portals for use by, say, comparison-shopping agents. Since metadata should change more slowly than data, the exact quantity (= 12417) of books (here: XML4You) sold is clearly inappropriate. Some magnitude interval (say $>$ 10000, $\leq$ 20000) would be preferable. To achieve this, in the LP formulation it would be easy to generalize the above `quantity` fact to a `magnitude` rule (the builtin "<", called here in prefix notation, in full Prolog could be replaced by a user-defined `less-than` relation, e.g. via successor structures):

```
magnitude("http://www.onoffbook.com/bookId/00498",Int) :-
                                        <(10000,Int), <(Int,20001).
```

As one example out of 10000, this rule can infer the result for the goal `magnitude(Url,12417)` via `<(10000,12417), <(12417,20001)`, by binding its `Url` variable to `"http://www.onoffbook.com/bookId/00498"`. More completely, the Herbrand model would now include this set of 10000 ground facts:

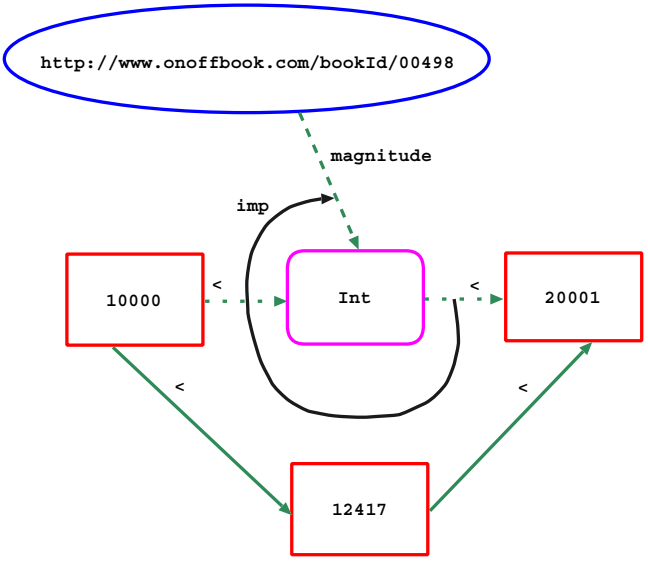
```
magnitude("http://www.onoffbook.com/bookId/00498",10001).
magnitude("http://www.onoffbook.com/bookId/00498",10002).
. . .
magnitude("http://www.onoffbook.com/bookId/00498",20000).
```

To obtain an unbounded interval one could omit the rule's second premise, leading to a Herbrand model with an infinite set of ground Horn facts.

Rules could again be marked up in XML, e.g. as shown in section 4 of [Bol01], for textually ordered Prolog rules, or as in RuleML (cf. end of section 2).

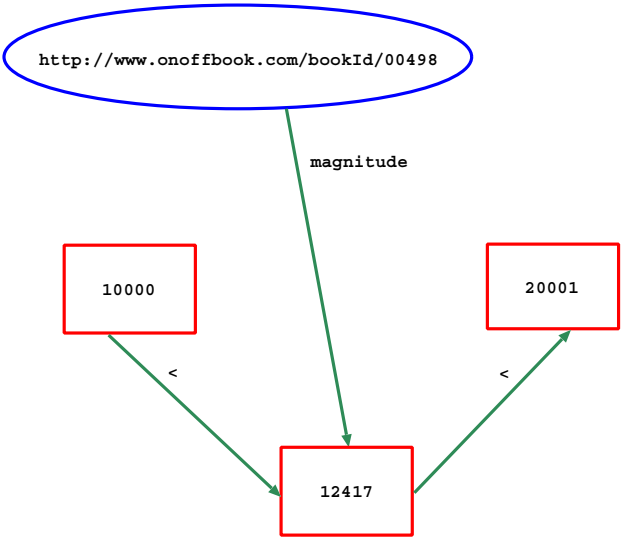
As an extension of (RDF) DLG diagrams, a rule can be depicted with dotted and dashed arcs for, respectively, premises and the conclusion, and with a generalized directed hyperarc labeled `implies` connecting a (Prolog-like) ordered sequence of premise arcs[3] with the conclusion arc. Logic variables like `Int` are depicted by rounded rectangles, whose names should be unique in a given namespace: since we do not allow node labels, the logical uniqueness of variables coincides with the graph-theoretical uniqueness of nodes.

Our example rule, augmented by the two premise facts `<(10000,12417).` and `<(12417,20001).`, thus corresponds to the following extended diagram:



For applying this rule to the premise facts the variable `Int` is unified with the constant `12417`. With the binding `Int = 12417` both "`<`" premises can then be verified. Therefore, the conclusion `magnitude("http://www.onoffbook.com/bookId/00498",12417)` is inferred. As a fact, it can again be depicted as a solid arc in a basic (RDF) diagram:

[3] The premises could thus also be represented by the RDF container Sequence; alternatively, in a more declarative representation, the premises would constitute an unordered multiset, represented by the RDF container Bag; cf. section 3. However, the properties of logical conjunction would require a container Set (N unordered elements without repetitions), still absent from RDF [LS99].



Such recursionless Datalog Horn rules – corresponding to relational views – would thus naturally enhance the expressive power of basic RDF, arriving at an inferential RDF. It is less obvious, however, where to stop in the expressiveness hierarchy towards recursive pure Prolog Horn rules or beyond (incl. versions of negation) for the purposes of RDF. Here, in section 3 we dealt with generalized Prolog constructors (for containers), in section 4 considered modal-logic operators (for meta-statements), and in the following section 6 will treat Prolog predicates (for N-ary relations).

6 Non-binary Relations, Logically

The RDF data model intrinsically only supports binary relations, and the recommended technique to deal with higher-arity relations is using "an intermediate resource with additional properties of this resource giving the remaining relations" [LS99]. This resource can be regarded as the reification of a higher-arity relationship. The two examples used in [http://www.w3.org/TR/REC-rdf-syntax/] both have a ternary flavor. Continuing our example in a similar manner, we could specify the book price in AUstralian Dollars as follows:

```
<rdf:RDF>
  <Description about="http://www.onoffbook.com/bookId/00498">
    <n:price>
      <rdf:value>39.5</rdf:value>
      <n:units rdf:resource=
                    "http://www.rba.gov.au/about/ab_monpol.html"/>
    </n:price>
  </Description>
</rdf:RDF>
```

In LP, we could directly reflect such a formalization, but it seems to be preferable to apply the unit as a constructor, `aud`, to its value, `39.5`, obtaining a Prolog-like structure. This, then, becomes the complex second argument of a binary `price` relation:

```
price("http://www.onoffbook.com/bookId/00498",aud[39.5]).
```

Actually, it would be clearer to distinguish a binary **relation** such as `online-sales` from a unary **attribute** such as `price`, which in functional-logic programming could be defined as a unary function returning a `price` value (as detailed in ONTOFILE [Bol99]):

```
price("http://www.onoffbook.com/bookId/00498") = aud[39.5]
```

Thus, these examples are not typical for N-ary relations. In a general non-binary relation the arguments cannot be grouped into a top-level pair in a natural way. Nor can they generally be split in another natural way, as e.g. – assuming a (Lisp-like) polyadic "`<`" – the ternary `<(10000,12417,20001)` into `<(10000,12417), <(12417,20001)` from section 5. For example, consider a ternary `ships` (sender-freight-receiver) relation with relationships such as the following (`john` could be a third URL):

```
ships("http://www.onoffbook.com",
      "http://www.onoffbook.com/bookId/00498",
      john).
```

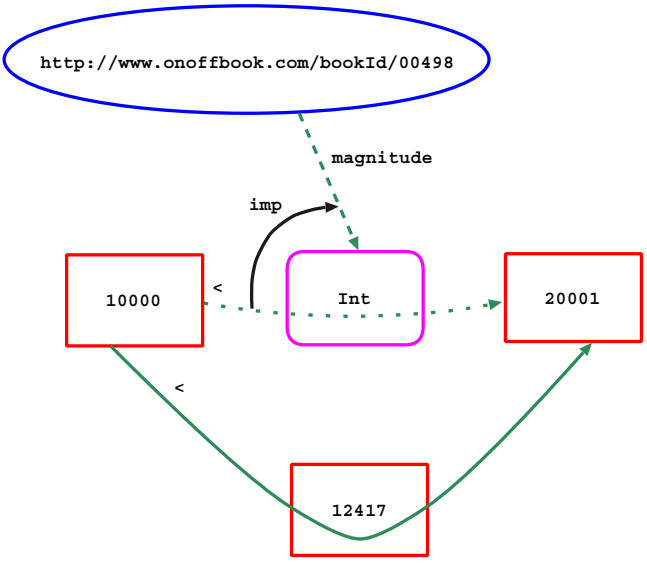
The reduction of such relationships to binary relationships would require unnatural reifications of the kind criticized in [Bol92] [http://www.dfki.uni-kl.de/~boley/drlhops.abs.html].

To avoid this, an N-ary extension of basic RDF can be used, where an R-labeled directed hyperarc connects the ordered sequence of N arguments of a relation R. The diagram in section 2 could, e.g., be extended by a `ships`-labeled hyperarc from the Onoffbook URL to John (cf. appendix A).

This can be combined with the rule extension of section 5. For example, a ternary "`<`" use permits joining the two premises of our previous rule into one, obtaining the rule

```
magnitude("http://www.onoffbook.com/bookId/00498",Int) :-
                                                <(10000,Int,20001).
```

Its diagram, augmented by the premise fact `<(10000,12417,20001).`, can be depicted thus:



7 RDF Types as Sort Predicates

In section 4 we already made use of `rdf:type` for the peculiar case of reified `Statements`. In general, using `rdf:type` properties, a resource can be specified to be an instance of one or more classes. In our RDF/XML example from section 2 we can specify the resource `"http://www.onoffbook.com"` to be an instance of class `Bookstore` and the resource `"http://www.onoffbook.com/bookId/00498"` to be an instance of class `Book`:

```
<rdf:RDF>
 <rdf:Description about="http://www.onoffbook.com">
   <rdf:type resource="http://description.org/schema/Bookstore"/>
   <s:online-sales rdf:resource="http://www.onoffbook.com/bookId/00498"/>
 </rdf:Description>
 <rdf:Description about="http://www.onoffbook.com/bookId/00498">
   <rdf:type resource="http://description.org/schema/Book"/>
   <v:name>XML4You</v:name>
   <v:quantity>12417</v:quantity>
 </rdf:Description>
</rdf:RDF>
```

Let us assume here that the classes `Bookstore` and `Book` are linked to their schema definition from a central place (rather than from each occurrence). Then, in LP, they could be used as special unary relations applied to resources to represent the above type declarations:

```
bookstore("http://www.onoffbook.com").
book("http://www.onoffbook.com/bookId/00498").
```

Such special unary predicates in sorted logics are called *sorts*, and are used for variable typing.[4] The above unary ground facts extensionally characterize ('leaf') sorts, like the A-box of description logics. The next section will intensionally characterize ('inner' and 'leaf') sorts, like their T-box.

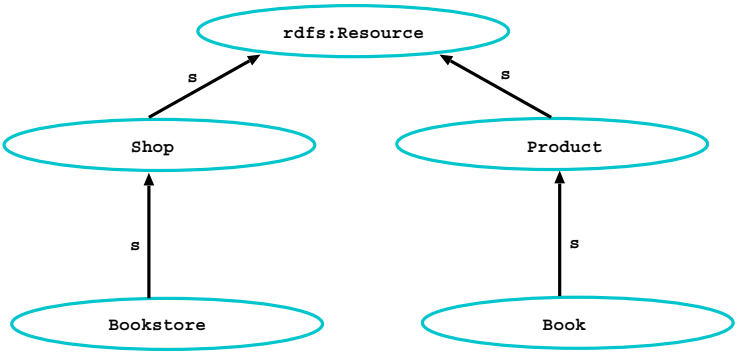
8 RDF Schema Core Properties in Second-Order Syntax

Let us now proceed to RDF Schema. It can be used to organize resources of `type Class` in a hierarchical fashion via a `subClassOf` element. For example, below a `Resource` root, we can specify that a `Bookstore` is a `Shop` and a `Book` is a `Product` as follows:

```
<rdf:RDF>
  <rdf:Description ID="Shop">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf
       rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  </rdf:Description>
  <rdf:Description ID="Product">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf
       rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
  </rdf:Description>
  <rdf:Description ID="Bookstore">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#Shop"/>
  </rdf:Description>
  <rdf:Description ID="Book">
    <rdf:type resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    <rdfs:subClassOf rdf:resource="#Product"/>
  </rdf:Description>
</rdf:RDF>
```

We use a simplified version of the corresponding RDF Schema diagram, assuming all classes (all nodes) point to `"http://www.w3.org/2000/01/rdf-schema#Class"` instead of repeating these `t(ype)` links, and only show the `s(ubClassOf)` links:

[4] RDF's permission of an instance I having multiple types $T_1, T_2, \ldots, T_k$ poses problems to object-oriented/frame systems (if I is a new instance of, say, class/frame T_1, classes/frames $T_2, \ldots, T_k$ must also be instantiatable to it) and to sort systems (an I-constrained variable would require a 'dynamic' type intersection $T_1 \sqcap T_2 \sqcap \ldots \sqcap T_k$). Since it crosses the class-instance boundary, such *multiclass membership* is harder than the *multiple subclassing*/inheritance considered in section 8. Multiclass membership can be simulated via multiple subclassing by 'statically' creating a new named subclass like $T = T_1 \sqcap T_2 \sqcap \ldots \sqcap T_k$, as in description logics [http://dl.kr.org/], of which I can then be a member.



In general, RDF Schema hierarchies need not form a tree but may constitute a directed acyclic graph (DAG). This partial order is usable for inheritance (DAGs: multiple inheritance) over classes in RDF applications.

In LP, it can be captured via a sorted logic and some second-order syntax, in the example essentially (omitting the namespaces `rdf`/`rdfs` and the repeated `type`s) using the `subsumes` relationships between super- and subsorts on the left:

```
subsumes(resource,shop).
subsumes(resource,product).
subsumes(shop,bookstore).
subsumes(product,book).
```

```
resource(X) :- shop(X).
resource(X) :- product(X).
shop(X) :- bookstore(X).
product(X) :- book(X).
```

These "second-order facts" abbreviate the first-order (Horn) rules on the right. However, the normal use of `subsumes` relationships is for sort-subsumption checking, not for backward deduction.

RDF Schema also uses a `subPropertyOf` element, for hierarchically organizing properties. For example, we can specify that `online-sales` and `offline-sales` are `sales`, thus:

```
<rdf:RDF>
  <rdf:Description ID="online-sales">
    <rdf:type resource=
           "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:subPropertyOf rdf:resource="#sales"/>
  </rdf:Description>
  <rdf:Description ID="offline-sales">
    <rdf:type resource=
           "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:subPropertyOf rdf:resource="#sales"/>
  </rdf:Description>
</rdf:RDF>
```

We omit here a `subPropertyOf` diagram, which would be quite similar to the above `subClassOf` diagram. This information can be used by RDF applications for inheritance over properties.

In LP, it may again be captured via second-order syntax, in the example essentially (without namespaces etc.) using the `subsumes2` relationships between binary super- and subrelations on the left (using a degenerate `subsumes0`, and with the previous `subsumes` standing for `subsumes1`, the subsumption between N-ary relations could be generally expressed via a `subsumes`N):

```
subsumes2(sales,online-sales).    sales(S,P) :- online-sales(S,P).
subsumes2(sales,offline-sales).   sales(S,P) :- offline-sales(S,P).
```

These "second-order facts" abbreviate the first-order (Horn) rules on the right. The `subsumes2` relationships, however, can be used in several ways, including constraint checking (cf. section 9) and backward deduction.

9 RDF Schema Core Constraints as Signatures

RDF Schema employs `domain` and `range` constraints to specify the classes on whose instances a property can be used. For example, we can specify that `sales` has a `Shop` `domain` and a `Product` `range` as follows:

```
<rdf:RDF>
  <rdf:Description ID="sales">
    <rdf:type resource=
           "http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
    <rdfs:domain rdf:resource="#Shop"/>
    <rdfs:range rdf:resource="#Product"/>
  </rdf:Description>
</rdf:RDF>
```

In LP, this can be expressed (ignoring namespaces etc.) by signatures, which are written here as fact-like "^"-declarations applying, e.g., binary relations to their domain and range "$"-sorts [Bol99]:

```
sales($shop,$product)^
```

While RDF Schema does not explicitly demand it, RDF applications should certainly have `domain` and `range` constraints inherited from super- to subproperties (cf. section 8). For example, in our LP notation, the above signature declaration, together with the previous `subsumes2` declaration, should imply two further signature declarations:

```
online-sales($shop,$product)^
offline-sales($shop,$product)^
```

Unlike for RDF Schema properties, in LP one could also use *polymorphic* relations, further constraining certain domain-range pairs such as `bookstore` with `book` or `drugstore` with `drug`:

```
sales($bookstore,$book)^
sales($drugstore,$drug)^
```

For a shop like "http://www.onoffbook.com", which is a bookstore, this would mean that its product sales must actually be book sales.

Sort polymorphism can be combined with property inheritance, in our example finally implying these signature declarations:

```
online-sales($bookstore,$book)^
offline-sales($bookstore,$book)^
online-sales($drugstore,$drug)^
offline-sales($drugstore,$drug)^
```

10 Conclusions

This paper showed how RDF's syntaxes and diagrams can be reconsidered as a special case of knowledge representation with logic programs and hypergraph diagrams. For several representation problems we suggested RDF extensions, e.g. to permit a more direct treatment of non-binary relations and Horn rules. We discussed Herbrand semantics for facts and rules but did not go into semantics for modal logics and other LP extensions. For another formalized treatment of RDF in a KR context see [Cha00] [http://www710.univ-lyon1.fr/~champin/rdf-tutorial/], giving a logical account of RDF inheritance and constraints, and discussing RDF implementation issues of containers, self-reference, etc.

RDF queries and inferences have been implemented with techniques from LP, e.g. building on F-logic in SiLRI [DBSA97] and TRIPLE [ABvE+01]. Efficient subsumption checking for ontologies in RDF syntax is provided by the FaCT implementation of description logics [Hea00]. RDF inferencing based on tractable graph algorithms is implemented in Euler [ftp://windsor.agfa.be/outgoing/RCEI/NET/euler/index.html]. Several further approaches are indexed at "Mozilla RDF / Enabling Inference" [http://www.mozilla.org/rdf/doc/inference.html] and "RDF query and inference" [http://rdfinference.org/]; many related topics are discussed in 'RDF-Logic' [http://lists.w3.org/Archives/Public/www-rdf-logic/].

XML Schema with its differentiated type system in Part 2 will complement RDF typing [http://www.w3.org/TR/xmlschema-2/]. For this and other reasons XML Schema should be analyzed from a logic perspective as well. Future work should also explore which version(s) of negation would make most sense in the open world of the Web, where (negative) conclusions should be drawn skeptically: one candidate is van Gelder's well-founded semantics (Przymusinski: 3-valued stable semantics) [http://lists.w3.org/Archives/Public/www-rdf-interest/1999Dec/0134.html].

The current RDF/XML serialization syntaxes would benefit from the following revision of the XML 1.0 standard: Instead of writing each closing XML bracket redundantly as a full end-tag that even repeats the namespace prefix, one could permit – approaching Prolog structures – the use of a 'neutral' closing bracket </>, as in XML-QL [http://www.w3.org/TR/NOTE-xml-ql/]. Recent proposals by Berners-Lee and Melnik attempt to unify the RDF and XML syn-

taxes [http://www-db.stanford.edu/~melnik/rdf/fusion.html]; meanwhile Berners-Lee is developing Notation 3 (N3) [http://www.w3.org/2000/10/swap/Primer.html]. For a kind of 'syntax-independent' RDF editing a variant of Protégé-2000 has been developed [NSD+01] [http://smi-web.stanford.edu/projects/protege/protege-rdf/protege-rdf.html].

A wide use of information agents will call for a standardized, XML-based markup language for semantic resource description on the Web. RDF already provides a simple kernel of such a representation language, but will need extensions to permit less complex representation constructions for descriptions of the kinds discussed in this paper. The question is whether these extensions can be incorporated into the current RDF standard or will need some principal revision. The issues parallel the old debate of whether and how to extend simple semantic nets towards predicate logic [Woo75], except that now the nodes (and arc labels) can be URLs and the serialization syntax is based on XML. Finally, the relationships between such an extended RDF and emerging description-logic (DLML [http://co4.inrialpes.fr/xml/dlml/]), ontology (XOL [KCT99], OIL [Hea00]), rule [http://www.dfki.de/ruleml], and agent (DAML [Hen00] [http://www.daml.org/]) languages need further study.

A Diagram Form of Directed Hyperarcs Exemplified

Directed-hyperarc arrows for N-ary relationships cut N-2 intermediate node occurrences. Thus, the arrow for the `ships` relationship from section 6 (N=3) cuts the node for XML4You's book URL, `"http://www.onoffbook.com/bookId/00498"`:

References

[ABvE+01] A. Abecker, A. Bernardi, L. van Elst, A. Lauer, H. Maus, S. Schwarz, and M. Sintek. FRODO: A Framework for Distributed Organizational Memories. Milestone M1: Requirements Analysis and System Architecture. Technical Report D-01-01, DFKI, March 2001.

[BG00] Dan Brickley and R.V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. Candidate Recommendation CR-rdf-schema-20000327, W3C, March 2000.

[BLF99] Tim Berners-Lee and Mark Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of xxx the World Wide Web by its Inventor.* Harper, San Francisco, 1999.

[Bol92] Harold Boley. Declarative Operations on Nets. In Fritz Lehmann, editor, *Semantic Networks in Artificial Intelligence,* volume 23, pages 601–637. Special Issue of Computers & Mathematics with Applications, Pergamon Press, 1992.

[Bol99] Harold Boley. ONTOFILE: Exterior and Interior Ontologies of File/HTTP URLs. In Hannu Jaakkola, Hannu Kangassalo, and Eiji Kawaguchi, editors, *Information Modelling and Knowledge Bases X.* IOS Press, Amsterdam, "Frontiers in Artificial Intelligence and Applications", Spring 1999.

[Bol01] Harold Boley. Cross-Fertilizing Logic Programming and XML for Knowledge Representation. In Rolf Grütter, editor, *Knowledge Media in Healthcare: Opportunities and Challenges.* Hersey - London - Melbourne - Singapore: Idea Group Publishing, 2001.

[Cha00] Pierre-Antoine Champin. RDF Tutorial. Technical Report, LISI (Laboratory of Information Systems Engineering), Université Claude Bernard, Lyon 1, March 2000.

[DBSA97] Stefan Decker, Dan Brickley, Janne Saarela, and Jürgen Angele. A Query and Inference Service for RDF. In QL'98 -*The Query Languages Workshop,* http://www.w3.org/TandS/QL/QL98/. World Wide Web Consortium, 1997.

[Hea00] Ian Horrocks and Dieter Fensel et al. The Ontology Inference Layer OIL. Technical Report, http://www.ontoknowledge.org/oil/TR/oil.long.html, June 2000.

[Hen00] James A. Hendler. Activities at DARPA. http://www.cs.umd.edu/~hendler/darpa.html, 2000.

[KCT99] Peter D. Karp, Vinay K. Chaudhri, and Jerome Thomere. XOL: An XML-Based Ontology Exchange Language. Technical Report, Artificial Intelligence Center, SRI International, August 1999.

[Llo87] John W. Lloyd.*Foundations of Logic Programming.* Springer-Verlag, Berlin, Heidelberg, New York, 1987.

[LS99] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Recommendation REC-rdf-syntax-19990222, W3C, February 1999.

[NSD+01] N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R. W. Fergerson, and M. A. Musen. Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[Woo75] William A. Woods. What's in a Link: Foundations for Semantic Networks. In Daniel C. Bobrow and A. M. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, New York, 1975.

An Approach to Building Mobile Intelligent Agents Based on Anytime Migration

Naoki Fukuta, Takayuki Ito, and Toramatsu Shintani

Department of Intelligence and Computer Science,
Nagoya Institute of Technology,
Gokiso-cho, Showa-ku, Nagoya 466-8555, JAPAN
{fukuta,itota,tora}@ics.nitech.ac.jp

Abstract. In this paper, we present an implementation of a Java-based framework for building mobile agents : MiLog (**M**obile **i**ntelligent agents using **Log**ic programming). MiLog provides logic programming and Java programming for developing intelligent agents since it helps users to effectively construct intelligent mechanisms. We address how to realize an anytime migration mechanism, a HTTP(Hyper Text Transfer Protocol)-based communication mechanism among agents, and web service/access mechanism. Anytime migration enables agents to reactively and autonomously migrate from one computer to another. We present practical applications that have been built on MiLog framework. These applications demonstrate that MiLog is an effective tool for developing practical applications based on mobile intelligent agents.

1 Introduction

In this paper, we present an implementation of a Java-based framework for building mobile agents: MiLog (**M**obile **i**ntelligent agents using **Log**ic programming). Intelligent agents have been studied very widely in the field of artificial intelligence and multi-agent systems. An agent can act autonomously and collaboratively in a network environment on behalf of its users. Also, they can autonomously help users to extract, filter, integrate, and monitor desired information on the huge information source (e.g. the Internet). On the other hand, a lot of mobile agents have been proposed and developed (e.g. Telescript[27], Aglets SDK[16], Concordia[28], etc.) . Mobile agents are programs that can migrate from one computer to another computer in a network (e.g. the Internet, wireless communication network) based on their own decisions. They can suspend their execution anytime, transport to another computer and resume execution on the destination computer. The main advantages of mobile agents[17] are, they can reduce network load, they overcome network latency, they execute asynchronously and autonomously, they adapt dynamically, etc. According to the above advantages, we can expect to further enhance the ability of intelligent agents by creating them as mobile agents.

AI techniques are necessary to realize intelligent behavior and effectively develop practical applications based on mobile agents. Therefore, we selected a Prolog language as a logic programming language to design agents in MiLog. Prolog enables users to develop sophisticated inference engines that could lead to effective cooperation and

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 219–228, 2001.

coordination mechanisms among agents. In Prolog, the backtracking mechanism is one of the most important features for realizing flexible and intelligent inference mechanisms.

In MiLog, we can use logic programming and Java programming. By using logic programming we can develop practical intelligent agent systems based on mobility. In addition, MiLog can strongly support Java programming. Since the MiLog environment is implemented as Java classes, we can build it into our own Java programs and use MiLog functions. Also, we can implement our own MiLog predicates (Prolog functions) by using Java, and then add them into the MiLog environment.

When we implement mobile agents, it is important to carefully develop a **migration** mechanism[4]. Migration mechanisms transfer agents from a certain environment to another. In general, existing migration mechanisms are classified into three methods based on **mobile code**, **weak migration**, and **strong migration**. This classification depends on which internal parts - program, data, and execution stacks - of agents can be transferred during migration. If only agents programs are transferred, these agents are called mobile codes or downloadable software. The migration mechanism that transfers agents programs and data are called the weak migration. Most of existing mobile agents frameworks employ a weak migration mechanism. The problem is that weak migration mechanisms cannot transfer agent's execution stacks. This means that we cannot use Prolog-based backtracking techniques if we use a weak migration mechanism. If agent's program, data, and execution stacks are transferred, the migration mechanism is called the strong migration. In order to continue backtracking process during migration, agents need to bring their execution stacks from one computer to another. Therefore, in MiLog, agents are transferred by using a strong migration mechanism. Also, agents should be able to reactively respond to changes in an environment. Therefore, we propose an **anytime migration** mechanism for MiLog agents. Anytime migration enables mobile agents to migrate reactively and execute their jobs continuously even after migration. Anytime migration is based on strong migration described above and **interruption**. Interruption techniques are used for reactive behavior of agents.

Another significant difference between MiLog and similar approaches is the MiLog can provide web-related functions (e.g. Web servicing function, information-extracting function from web, etc.).

The rest of this paper consists of the following 4 sections. In Section 2, we present the architecture of MiLog. In Section 3, we show implementation of MiLog by using Java. In particular, we present a HTTP-based communication mechanism among agents, and an anytime migration mechanism. In Section 4, we present the performance of MiLog and its practical applications. Then, we discuss about a tracking technique for transferred mobile agents and a security model. In Section 5, we present related work to compare our approach with other approaches. Finally, in Section 6, we describe some concluding remarks.

2 Architecture of MiLog

MiLog is implemented on Java. MiLog can be used on any operating systems that are supported by Java (e.g. Mac OS, Windows 95/98/2000/NT, Linux, Solaris, FreeBSD, etc.). The MiLog runtime environment manages resource of all agents. The environment

invokes one meta-agent and one HTTP server agent at starting time. Agents have their own execution stacks, codes, and data. In MiLog, we have implemented extended Prolog predicates (functions) by using Java. We can develop built-in procedures in MiLog by using Java. In addition, since the MiLog environment is implemented as Java classes, we can build it into our own Java programs and use MiLog functions from Java programs.

An agent consists of a meta-interpreter and a base-interpreter. The base-interpreter is a Prolog interpreter that executes programs for agents. Meta-interpreter manages interruption of execution of the base-interpreter and communication among agents. Each MiLog runtime environment has one meta-agent and one HTTP server agent. The meta-agents are prohibited to migrate, and have a special interpreter to manage migration, communication among agents, and HTTPaccess functions. The HTTP server agents have special functions to manage HTTP requests.

3 Implementation of MiLog

3.1 Communication Mechanism Based on HTTP

In MiLog, all communications among agents are based on HTTP. Agents can receive an HTTP request message as a query in Prolog. Fig. 1 shows the communication mechanism between two MiLog agents. Agents can communicate based on two types of communication mechanisms. One is the "query" method that sends a query message to another agent, and waits a reply from that agent. The other is the "request" method that sends a query message to another agent without waiting a reply. Fig. 1 presents an example of communication via the "request" method between two MiLog agents. The sender agent tries to request sending a massage to the meta-agent that locates on the same MiLog runtime environment. Then the meta-agent tries to send the message to another meta-agent in a remote environment in which the receiver agent exists. To send the message to a remote computer, the meta-agent uses a built-in predicate. A MiLog environment can receive an HTTP request via a request router that has been implemented in Java. Then, the received message is passed to an HTTP server agent. The HTTP server agent tries to send the message to the meta-agent in that environment. If this meta-agent is doing tasks, the HTTP server agent waits until finishing these tasks. The HTTP server agent can keep several messages in a message queue. The meta-agent sends the message to the receiver agent. The receiver agent evaluates this message as a query message. In this case, the receiver agent does not reply since this communication is based on the "request" method. In the case of a "query" message, the sender agent suspends and waits the reply (the result) of the query message. The receiver agent does reply the result of evaluation of the message received. The sender agent is resumed by his environment's meta-agent and received the reply message.

In addition, MiLog agents can access to WWW servers on the Internet. Also, they can be received HTTP requests as a WWW server. To access WWW servers, agents use MiLog built-in predicates. To respond HTTP requests, agents receive them and send answers via the request router and the HTTP server agent.

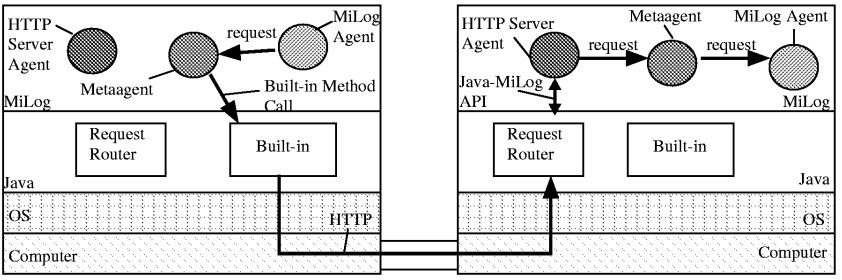


Fig. 1. Communication between Agents

3.2 Implementation of the Anytime Migration

The anytime migration consists of interruption and strong migration. An interruption mechanism is realized based on a two-layered architecture. Agents have two interpreters, a base-level interpreter and a meta-level interpreter. The base-level interpreter handles to execute agent's task. The mete-level interpreter manages interruption and migration. The layered architecture enables users to distinguish their agent programs into codes for agent's task itself and codes for handling interruption and migration. Users can easily write reactive behavior for agents by using meta-level interpreter. The followings are the advantages of the interruption mechanism. 1) The interruption mechanism enables agents to migrate anytime even if the migration was not expected previously. 2) Users can be relieved to write codes to interrupt. If users write codes for interruption, users need to carefully write how to suspend and resume tasks.

In order to implement strong migration, we need to capture agent's current execution stacks. Although Java supports capturing contents of static objects, it does not support to capture execution stacks. There exist several studies to capture execution stacks. There are two main approaches. One is to modify the Java virtual machine(VM). The other is to implement its own Java VMs. Related work is presented in Section 5. These extensions or modifications of the Java VM cannot keep the Java VM compatible. Therefore, in MiLog, we implemented a Prolog interpreter on the Java VM. In MiLog, internal execution stacks in a Prolog interpreter are constructed based on Java objects. This enables us to capture execution stacks by using serialization techniques for Java objects.

Fig. 2 shows an internal structure of execution stacks at the point of serialization. All Prolog related stacks (i.e. a control stack, a choice point stack, and a trail stack) and the program counter are implemented as java objects, and linked together by pointers. In order to effectively serialize all of the objects, we prepare a special object that has several pointers that refer these objects. All we need to do for serializing all of the objects is to apply the Java serialization method to this special object. By using the Java serialization method, we can get a binary data that keeps all object information and all pointer information.

The timing for capturing is one of the most important points to implement migration mechanisms. We decompose an internal processing of the Prolog interpreter into several units (one unit is called a stage). MiLog can capture the execution stack when a stage is

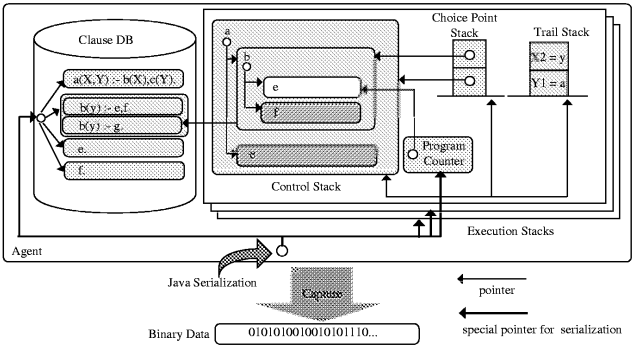


Fig. 2. Capturing execution stacks based on Java serialization technique

transferred to another stage. In MiLog, we utilize the box model[3] to define one stage. The box model provides a simple way to show the control flow of Prolog. This model has been adopted in a lot of Prolog systems, and used for debugging in general. In box model, one goal (procedure) is modeled as one box. Each box has four ports (CALL, EXIT, FAIL, and REDO) through which Prolog interpreter can enter (CALL/REDO) or leave (EXIT/FAIL) the goal. In MiLog, each entering and leaving step is defined as one stage. The followings are meaning for each port. CALL: When Prolog interpreter tries to prove the goal, it enters from the CALL port. EXIT: If Prolog interpreter entered from the CALL port, it can leave through EXIT port if the goal is proven. FAIL: If Prolog interpreter cannot prove the goal it leaves from FAIL port. REDO: When Prolog interpreter is backtracking, it enters into the goal from the REDO port. These boxes are linked from port to port.

Based on the above stages, we realized an interruption mechanism. In an agent, when the agent received an interruption signal, the meta-interpreter set the agent's *State* flag to *suspend*. The base interpreter checks the agent's *State* flag once a stage. If the *State* flag is *suspend*, the base interpreter captures the execution stack.

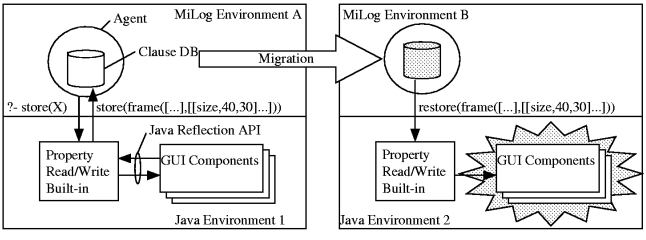


Fig. 3. Migration with graphical user interface

One of the most important issues for Java-based mobile agents is the way to migrate an agent that has graphical user interfaces(GUI). Since there exist non-serializable components in Java API (e.g. some components of Swing package), it has been difficult to

transfer non-serializable GUI. In MiLog, agents can transfer Java-based GUI even if it consists of non-serializable components. Fig. 3 shows the migration mechanism with GUI in MiLog. In MiLog, agents have only property information on GUI components. This property information (e.g. the size and location of a window, etc.) is read via the property read/write built-in methods. Before migration, agents read properties from GUI components, and store the properties as a clause set in MiLog. Then, after migration, agents restore GUI components based on this clause set via the property read/write built-in. The followings are advantages of this method. (a) An agent can behave like as he brings his GUI even if that GUI consists of non-serializable Java components. (b) By adjusting the property read/write built-in predicates, we can restore GUI components that are more suitable to the current Java environment. Assume that Java environment 1 has Swing package (the Java-based most recent GUI library), and Java environment 2 has only AWT package (the basic Java GUI library). Since the agent has only property information, this agent can restore the GUI component by using Swing package and AWT package in the Java environment 1 and the Java environment 2, respectively.

4 Discussion

4.1 Performance of MiLog

MiLog is implemented in 100% pure Java. Therefore, we can use MiLog on any computer that has Java VM. MiLog is provided as a Jar archive (Java's standard archive format), the size of the Jar file is 284KB. It is small enough to quickly download. The work for installation is only setting one Java class path. It is simple enough to quickly start. Performance of MiLog interpreter is 27.53 KLIPS on the following environment : AMD Athron/700MHz, 128MB Memory, Windows98, and IBM JDK1.1.8. The followings are concrete data for a simple agent that makes 100 round trips between one computer and another. In MiLog, the program length and the program size are 14 lines and 293 bytes, respectively. They are 82 lines and 1,775 bytes in the case of Aglets SDK 1.0.2. Aglets SDK has been one of the first Java based mobile agent systems[16]. The Aglets framework employs a weak migration mechanism that transfers only agent's program and data. In MiLog, the amount of data for one migration is 2005 bytes. In the case of Aglets SDK 1.0.2, it is 2,058 bytes. In MiLog, the required time to one migration is 166.2 milliseconds. In the case of Aglets SDK 1.0.2, it is 69.33 milliseconds. These results show that MiLog based on strong migration is thoroughly effective for developing mobile agent systems.

4.2 Applications

As practical applications, the following applications have been developed using MiLog. **GroupBuyAuction**[29]: GroupBuyAuction is an agent-based virtual commerce server that allows volume discount. In GroupBuyAuction, while buyer agents can make a coalition (a group), seller agents can offer group discount price for the coalition. These seller agents and buyer agents have been implemented on MiLog. Also, the place on which these agents negotiate has been implemented on MiLog. **Meeting scheduling system**[21][26]: We have developed a meeting scheduling system that has two types

of scheduling mechanisms. One is based on multiple negotiations among agents[21]. The other is based on a weighted distributed constraint satisfaction mechanism[26]. Since both of the mechanisms need agents that can continuously infer during migration, these agents are effectively implemented in MiLog. **Group Choice Design Support System(GCDSS)**[10][11]: GCDSS[11] is an agent-based group decision support system that can support group selecting a desired alternative from several candidate alternatives. In this system, agents negotiate with each other on behalf of their users. As negotiation mechanisms, we proposed a persuasion mechanism[11], and an argumentation mechanism[10]. Logic programming in MiLog helped us to build these mechanisms. **BiddingBot**[9][12]: BiddingBot is an e-commerce support system that can support to monitor, attend, and bid in multiple online auction sites. MiLog's WWW access and WWW server functions enable us to effectively implement this system.

4.3 GUI Development Tools: MiPage and iML

Existing development tools for mobile agents have not focused on building GUIs. MiLog supports efficient GUI development. In order to support to implement GUIs, we have developed the following two rapid development tools: MiPage and iML. In MiPage, users can use MiLog as a scripting language in HTML documents. This means that users can embed MiLog programs in HTML documents like Java scripts. The difference is that MiLog programs are executed on Web servers, whereas Java scripts are executed on Web browsers. Users can access to their agents via Web pages that are created by using MiPage. By using iML, users can write MiLog's GUIs in Java. These GUIs developed on iML are automatically created as parts of a MiLog agent. This means that users do not need to spend time to implement connection parts between Java and MiLog. As we present on Section 3.2, these Java-based GUIs can be migrated with agents.

4.4 Tracking Transferred Agents and Security Policy

In order to track transferred mobile agents, each meta-agent has a list that contains name of agents who visited the meta-agent's environment. Also, this list has information on where each agent left. For example, if the agent "Smith" went to the neighbor computer "comp A," this list has the information "Smith : comp A." This information is similar to "footprint" of agents. In order to track a certain agent, we can follow this footprint information. Whenever agents leave from a certain environment, the meta-agent in that environment updates the list. Users can track their own agents. If a user want to know where his agent is, he accesses to the meta-agent in his environment via a WWW browser. The meta-agent follows footprints and finds the user's agent by asking other meta-agents. Then, the meta-agent tries to establish connection to the user's agent. The user can now access his agent directly.

Security is the one of most important issues for mobile agents. There have been several researches on security of mobile agents[6][16]. Although our main focus is not on security, it's important to consider a security policy for MiLog. In MiLog, we assume that all MiLog environments should trust with each other, and agents also trust with each other. The MiLog environments are distributed on the Internet. This means that network bandwidths and latencies are varied, and sudden disconnection may occur.

On this assumption, the followings are two main issues that should be considered: (a) Protecting a machine from untrusted agents, and (b) Protecting an agent during migration.

In order to realize the first issue, when migrating agents, MiLog environments try to authenticate with each other. We employ an authentication mechanism on host-to-host communication including transmitting of agents. The authentication mechanism is based on secret key encryption techniques. We assume a group of MiLog environments in which a unique secret key is distributed and stored. Authentication information made by the secret key is put in header field of a request message for transmitting an agent. Transmitting of the agent is accepted by the destination only if the authentication information is valid for that group.

Although all hosts in that group are trusted, network is still untrusted. Someone could tamper or modify requests for migration on the network. Secure connections are important for this reason. This is the second issue. We use the Secure Socket Layer (SSL), which is a widely used public-key-based network security protocol that provides authentication, privacy protection, and data compression. In our implementation, two methods for connections, a simple socket-based method and an SSL-based method, are used. The advantage is that it makes better performance with enough security on the situation. For improvement of performance and stability, Sun's Java Secure Socket Extension (JSSE) APIs are used in our implementation.

5 Related Work

In this section, we show the difference between our approach and other approaches. We classify related mobile agent systems into the following 4 categories. **a)Systems based on their own interpreter:** These mobile agents systems (Telescript[27], TACOMA[13], Ara[20], MESSENGER[2], D'Agents[7],Flage[15], Mobidget[5], etc.) have their own interpreter to run agents. In general, they can use strong migration mechanism, since they can create their own interpreter based on their own design. The problem is that there is no compatibility with other software. **b) Systems based on weak migration and the original Java interpreter:** In general, in Java-based mobile agents systems(Aglets Software Development Kit[16], Concordia[28], Mole[1], Voyager[18], MobileSpace [8], Bee-gent[14], Gossip[25]), a mobile agent is developed as a Java class. The problem is the normal Java interpreter does not support capturing the execution stack. Therefore, they use weak migration. This means that agents cannot use backtracking and infer effectively during migration. **c)Systems based on strong migration and extended Java VM:** Systems (NOMADS[23] and MOBA[22]) in this category extended the Java VM in order to realize strong migration. They, however, lost Java's compatibility that is the most important feature of the Java VM. **d)Systems based on strong migration and their own interpreter written in Java:** MiLog, Plangent[19], and Jinni[24] are classified into this category. Plangent is a plan-based mobile agent system. The main difference between Plangent and MiLog is when agents can migrate. MiLog's anytime migration enables agents to move when they want. On the other hand, Plangent's agents cannot move during they are planning. They can move after finishing planning. Jinni is a Prolog-based mobile agent system. The main difference between Jinni and MiLog is an implementation method of migration. While our anytime migration enables agents

to infer by backtracking even after they migrate to another computer, Jinni's live code mobility does not handle backtracking during migration.

6 Conclusions

In this paper, we present an implementation of a Java based framework for building mobile agents : MiLog. The followings are significant advantages of MiLog. **1)Anytime Migration :** The anytime migration enables MiLog's agents to reactively and autonomously migrate from one computer to another. Agents can continue their inference during migration. In MiLog, we have developed a Prolog interpreter on the Java VM. This enables us to capture the execution stacks of agents, and realize an anytime migration mechanism. **2)Portability and Usability :** As we discussed in Section 4.1., MiLog is small enough to quickly download, and is simple enough to quickly start. Namely, MiLog has high portability. MiLog also has high usability. In a lecture (AI programming) of our department(ICS/NIT), most of undergraduate students could create simple MiLog-based applications. **3)User Interface Development support:** Existing systems ignore the importance of user interface, and leave it to the program language itself they used. In order to use mobile agents to build practical applications, it is important to support developing user interfaces. In MiLog, we provided MiPage and iML. These tools can assist users to create user interfaces.

References

1. J. Baumann, F. Hohl, K. Rothermel, and K. Strasser. Mole - concepts of a mobile agent system. *Journal of World Wide Web*, 1(3):123–137, 1998.
2. L.F. Bic, M. Fukuda, and M.B. Dillencourt. Distributed computing using autonomous objects. *IEEE COMPUTER*, 29(8):55–61, 1996.
3. W.F. Clocksin and C.S. Mellish. *Programming in Prolog*. Springer Verlag, 1981.
4. A.Fuggetta, G.P.Picco, and G.Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, may 1998.
5. S.Fujita, K.Koyama, T.Yamanouchi, S.Jagannathan, R. Kelsey, and J. Philbin. Mobile and distributed agents in mobidget. In *Proc. of the 1st International Symposium on Agent Systems and Applications 3rd International Symposium on Mobile Agents*, 1999.
6. R. S. Gray and D. Kotz and G. Cybenko and D. Rus. D'Agents, Security in a multiple-language, mobile-agent system. In G. Vigna, editor, *Mobile Agents and Security*, Lecture Notes in Computer Science, Springer-Verlag, pp.154–1871998.
7. R.S. Gray, D. Kotz, G. Cybenko, and D. Rus. Mobile agents: Motivations and state-of-the-art systems. In J. M. Bradshaw, editor, *Handbook of Agent Technology*. AAAI/MIT Press, 2000.
8. I. Satoh. Mobilespaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. In *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'2000)*, pages 161–168. IEEE Press, 2000.
9. H. Hattori, M. Yokoo, Y. Sakurai, and T. Shintani. Determining Bidding Strategies in Sequential Auctions: Quasi-linear Utility and Budget Constraints. In *Proc. of the 5th International Conference on AUTONOMOUS AGENTS(Agents-01)*, 2001(to appear).
10. H. Hattori, T. Ito, T. Ozono and T. Shintani. An Approach to Coalition Formation using Argumentation-based Negotiation in Multi-Agent Systems In *Proc. of the 14th International Conference on Industrial &Engineering Applications of Artificial Intelligence & Expert Systems(IEA/AIE-01)*, 2001 (to appear).

11. T. Ito and T. Shintani. Persuasion among agents : An approach to implementing a group decision support system based on multi-agent negotiation. In *Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 592–597, 1997.
12. T. Ito, N. Fukuta, T. Shintani, and K. Sycara. *BiddingBot*: A multiagent support system for cooperative bidding in multiple auctions. In *Proc. of the 4th International Conference on Multi-Agent Systems (ICMAS-2000)*, pages 399–400, 2000.
13. D. Johansen, R. v. Renesse, and F.B. Schneider. Operating system support for mobile agents. In *Proc. of the 5th IEEE Workshop on Hot Topics in Operating Systems*, 1995.
14. T. Kawamura, N. Yoshioka, T. Hasegawa, A. Ohsuga, and S. Honiden. Bee-gent : Bonding and encapsulation enhancement agent framework for development of distributed systems. In *Proc. of the 6th Asia-Pacific Software Engineering Conference*, 1999.
15. F. Kumeno, H. Sato, T. Kato, and S. Honiden. Flage: A programming language for adaptive software. In *Proc. of International Conference on Software Engineering'98*, 1998.
16. D.B. Lange and M.Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.
17. D.B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.
18. ObjectSpace. Voyager. www.objectspace.com/ products/ voyager/, 1995.
19. A.Ohsuga, Y.Nagai, Y.Irie, M.Hattori, and S. Honiden. Plangent: An approach to making mobile agents intelligent. *IEEE Internet Computing*, 1(4):pp.50–57, 1997.
20. H.Peine and T. Stolpmann. The architecture of the ara platform for mobile agents. In D.Milojicic, F.Douglis, and R.Wheeler, editors, *Mobility: Processes, Computers, and Agents*, pages 474–483. Addison-Wesley and the ACM Press, 1999.
21. T. Shintani, T. Ito, and K.Sycara. Multiple negotiations among agents for a distributed meeting scheduler. In *Proc. of the 4th International Conference on Multi-Agent Systems (ICMAS-2000)*, pages 435–436, 2000.
22. K. Shudo and Y. Muraoka. Noncooperative migration of execution context in java virtual machines. In *Proc. of 1st Annual Workshop on Java for High-Performance Computing*, pages 49–57, 1999.
23. N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, R. Jeffers, T.S. Mitrovich, B.R. Pouliot, and D.S. Smith. Nomads: toward a strong and safe mobile agent system. In *Proc. of the 4th International Conference on Autonomous Agents (Agents'2000)*, pages 163–164, 2000.
24. P. Tarau. Jinni: Intelligent mobile agent programming at the intersection of java and prolog. In *Proc. of The 4th International Conference on The Practical Application of Intelligent Agents and Multi-Agents (PAAM-99)*, 1999.
25. Tryllian. Gossip. www.tryllian.com, 1999.
26. T. Tsuruta and T. Shintani. Scheduling meetings using distributed valued constraint satisfaction algorithm. In *Proc. of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, pages 383–387, 2000.
27. J. E. White. Mobile agents. In Jeffrey M. Bradshaw, editor, *Software Agents*, chapter 19, pages 437–472. AAAI Press/The MIT Press, 1997.
28. D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet. Concordia: An infrastructure for collaborating mobile agents. In *Proc. of the 1st International Workshop on Mobile Agents (MA'97)*, 1997.
29. J. Yamamoto and K. Sycara. A Stable and Efficient Buyer Coalition Formation Scheme for E-Marketplaces. In *Proc. of the 5th International Conference on Autonomous Agents (Agents'2001)*, 2001 (to appear).

Knowledge-Based Information Agents

Xiaoying Gao[1] and Leon Sterling[2]

[1] School of Mathematical and Computing Sciences
Victoria University of Wellington
Wellington, New Zealand
xgao@mcs.vuw.ac.nz
[2] Department of Computer Science and Software Engineering
The University of Melbourne
Victoria, 3010, Australia
leon@cs.mu.oz.au

Abstract. This paper explains our approach to building knowledge-based information agents. Instead of building an agent from scratch, we advocate building an agent by adding knowledge to a framework. Knowledge is classified into three categories - general knowledge, domain specific knowledge and site specific knowledge - which enables knowledge reuse and sharing. The paper details the agent architecture, the components of each category of knowledge and the main functions of the framework.

1 Introduction

Information agents are intelligent pieces of software which can automatically search for information on the WWW [2] [10]. They usually deal with multiple Web sites in a single domain or multiple domains. One key step of building information agents is to extract information from multiple Web sites, that is, to transfer important information to structured data so that more accurate search can be carried out as querying a structured database.

The main challenge of building an information agent is how to make the agent scalable and adaptable. More and more online documents are becoming available and each has a different data format. The number of Web sites and their domains is huge and is growing very fast. Existing Web pages are being updated continuously, and their data formats may be modified at any time without any warning. While it might be easy to handcraft an information agent for one particular Web site in one specific domain for a particular time, how to update the Web site, how to adapt it and make it scalable to new Web sites and new domains, is a big challenge. There is an urgent need to develop methods and tools to ease agent generation and adaptation.

Recent research has used machine learning technology to build scalable agents [1] and to automatically learn information extraction patterns [6] [7] [9]. However, these systems work on relatively structured Web pages. The majority of Web pages with flexible data format, for example, data presented in free

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 229–238, 2001.

text and spread across sentences and paragraphs, are out of reach of current automatic systems.

Our research introduces a knowledge-based approach to support the generation and adaptation of information agents. We view an information agent as a knowledge-based system. The knowledge for guiding information extraction, such as information extraction patterns, is saved in the knowledge base of the agent. The information extraction process is coded as an inference engine. We assume the knowledge can be separated from the information extraction process. Instead of building an agent from scratch, an agent can be generated by adding knowledge bases to a reusable shell. An agent can be adapted to new domains and new Web sites by changing the knowledge bases. In slogan form,

Information Agent = Knowledge Bases + Agent Shell

We focus on building agents for information extraction from semi-structured data, that is data in an intermediate format between data in free text and structured data in databases. Typical examples are Web pages provided by online services such as classified advertisements, product categories, and telephone books. We believe that knowledge plays an important role for information extraction from semi-structured data, and information extraction from semi-structured data can be achieved based on a limited amount of knowledge with only simple natural language processing. Semi-structured data provides the right level of diversity and difficulty for testing our methods.

The rest of this paper is organized as follows. Section 2 discusses the knowledge that is useful for building agents and describes our classification of knowledge into three categories. Section 3 introduces the agent architecture. The two main parts of an agent, the knowledge bases and the information extraction engine are discussed in Sections 4 and 5 respectively. Section 6 gives some experimental results, while the final section concludes this paper.

2 Knowledge Classification

Focusing on information extraction from semi-structured data, we have examined thousands of Web pages. We summarize the knowledge that is useful for guiding the information extraction as follows. We classify knowledge into three categories: general knowledge, domain specific knowledge and site specific knowledge.

- General Knowledge. General knowledge is true for most online documents, if not for all of them, that is, the knowledge is both domain independent and site independent. Typical examples are the common usage of HTML tags, for example, what is a table, what is a paragraph, and what is a line.
- Domain Specific Knowledge. Domain specific knowledge is true in a particular domain. The knowledge is site independent, that is, the knowledge is consistent for most Web sites if not for all of them as long as the Web sites present data in the same domain. For example, in the real estate domain,

each property in an online advertisement has a suburb where it is located; the price for renting a property is usually denoted by a "$", followed by a number, and a unit such as "per week" or "per month". Domain specific knowledge is usually specified using terms in a specific domain and may not generalize to other domains.
- Site Specific Knowledge. Site specific knowledge is true for a particular site. To prevent the intersection with domain knowledge, we define the site specific knowledge being domain independent, that is, if the knowledge is true for this site and also true for this domain, then it is classified into domain knowledge. Site specific knowledge mainly consists of the site specific data formatting conventions, for example, in a particular Web site called NewsClassifieds, suburb names are printed in all capital letters. Site specific knowledge is tailored to a specific site and unlikely to be consistent with other sites.

This knowledge classification enables knowledge reuse and sharing, and also gives guidance for agent adaptation. General knowledge is completely reusable and can be shared for many information extraction tasks. Domain knowledge can be reused and shared for Web sites in the same domain. Site specific knowledge is limited to Web pages on the same site.[1] To adapt an agent to a new domain, new domain specific knowledge is needed. To adapt an agent to a new site, new site specific knowledge needs to be added.

3 Agent Architecture

The architecture of our knowledge-based information agent is shown in Figure 1. An agent contains three knowledge bases and a framework. The three knowledge bases are: General Knowledge Base (G), Domain Knowledge Base (D), and Site Specific Knowledge Base (S). The framework contains three main functions: Web Access Engine, Information Extraction Engine and Matcher.

We detail the three functions as follows:

- The Web Access Engine utilizes site specific knowledge to get access to the Internet and download Web pages. When the information source has a searchable interface, the Web Access Engine needs to use the original user query or the structured query (the IE results from the original query) to interact with the interface.
- The Information Extraction Engine uses general, domain specific and site specific knowledge to extract information from both the user query and Web pages, and to save them as structured data.

[1] We do not expect site specific knowledge can be shared by different domains on the same site, because the information to be extracted differs greatly from one domain to another. Actually, one site with different domains is regarded as different sites, that is, one site has only one specific domain. For example, the NewsClassifieds with real estate advertisements is one Web site and NewsClassifieds with car advertisements is another Web site.

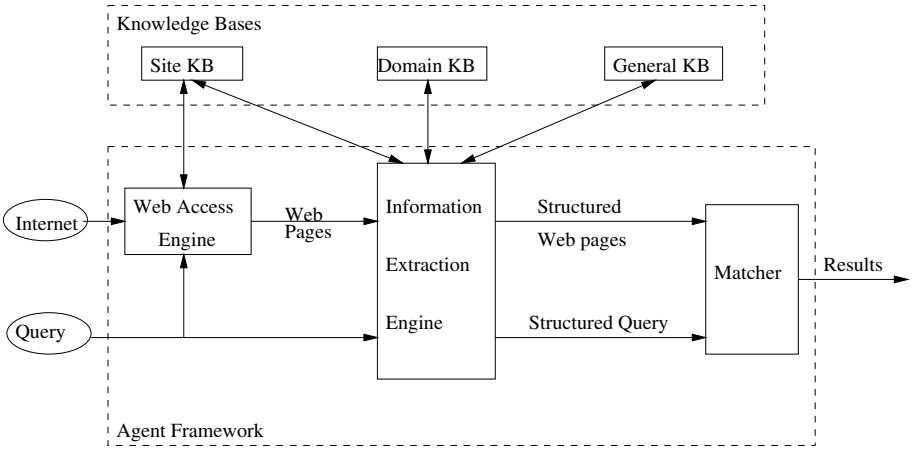


Fig. 1. The Architecture of Knowledge-Based Information Agents

– The Matcher matches the structured query with the structured data of Web pages and outputs the results.

An agent shell includes the framework and the general knowledge base. An information agent can be built by adding the domain knowledge base and the site specific knowledge base to the agent shell. An example of an agent shell is given in [8].

As we mentioned in the introduction, this paper focuses on how to use knowledge to extract information from Web page. The next two sections discuss the main components of three knowledge bases and one main function of the framework, the information extraction engine.

4 Three Knowledge Bases

Focusing on information extraction from semi-structured data, we summarize the main components of the three categories of knowledge as follows:

– General Knowledge (G)
  - The usage of HTML tags, particularly, the page structure levels (word, line, paragraph, page) the tags are linked to.
  - The identification methods of basic data types such as tag, text, character, etc.
  - Common sense knowledge such as related data are often presented together.
– Domain Specific Knowledge (D)
  - The concepts and the relationship of concepts. For example, in real estate domain, the concepts include “real estate ad”, “property”, “suburb”, “price”, “size”, “type”, etc. The concepts can be put in a hierarchy,

for example, "real estate ad" consists of a number of "property", each property consists of "suburb", "price", "size", and "type". The concept in the last level (the atomic concepts) are called knowledge units (KU) in this paper.
  - How to identify the knowledge units (atomic concepts) and how to extract the value of knowledge units. Its major components include the domain specific terminology (for example, in the real estate domain, a suburb database can be used to identify the *Suburb* of each property from online advertisements), and domain specific data formatting conventions (for example, the *Price* for renting a property is usually a "$", followed by a number, and a unit such as "per week" or "per month")
- Site Specific Knowledge (S)
  - Site specific knowledge of the interface of each Web site. Most semi-structured data is presented as the search results of local search engines. In order to interact with the local search engine, the system needs site specific knowledge of the interface of the local search engine. For example, if the interface is an HTML form, the system needs to know the access method "Get" or "Post", and the way to generate query strings.
  - The information extraction patterns for fields (a group of knowledge units).
  - Site specific information extraction patterns for concepts.
  - Site specific usage of HTML tags
  - Site specific concept hierarchy
  - Site specific information extraction pattern for individual knowledge units.

    The site specific knowledge base does not have to be complete. All items except the first one are optional. Site specific knowledge is used for describing some special sites which can not be described by domain specific knowledge.

The three categories of knowledge have different priorities when they are used for information extraction. The priorities are given as follows:

1. Site specific knowledge (S)
2. Domain knowledge (D)
3. General knowledge (G)

During the information extraction process, the site specific knowledge has the highest priority and the general knowledge has the lowest. When there are conflicts between the knowledge, the higher priority knowledge overrides the lower priority knowledge. When we get a particular site, we search for site specific knowledge first. If some site specific knowledge is found, this knowledge is used instead of the associated knowledge in either the general knowledge base or domain knowledge base. For example, if a site specific information extraction pattern is found for a special knowledge unit, then this pattern is used for extracting the knowledge unit, instead of using the more general pattern in the domain knowledge base.

5 Information Extraction Engine

The information extraction engine utilizes the three categories of knowledge, extracts information from Web pages and saves the information as structured data. Figure 2 shows how it works.

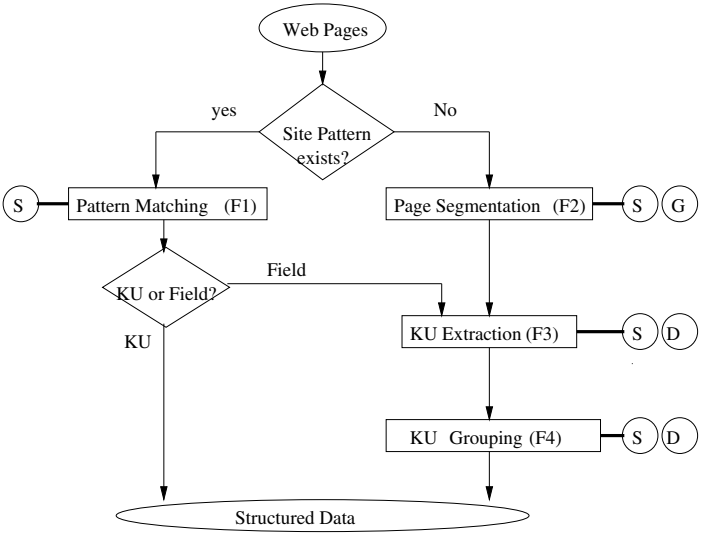


Fig. 2. The Information Extraction Engine

The input to the system is the source file of a Web page with its site name (the Web site where the page is downloaded from). The output is a number of concepts, each concept consists of a number of knowledge units. The system first checks whether there are site specific patterns 1) If yes, then the page is parsed through the pattern matching function. There are two kinds of output: a) if the output consists of concepts or knowledge units, they are directly saved to structured data. b) if the output consists of fields (each field contains one or more knowledge units), then the fields are used as the input to knowledge unit extraction function, and then through a knowledge unit grouping function, to be parsed to structured data and saved. 2) If no, the page is parsed through three functions: page segmentation, knowledge unit extraction and knowledge unit grouping.

The four functions and the categories of knowledge they use are detailed as follows:

– Pattern matching (F1): this function utilizes site specific knowledge, including site specific information extraction patterns for concepts, patterns for individual knowledge units and patterns for a group of knowledge units (fields), to parse a page into concepts, knowledge units and fields. The concepts and

knowledge units are then saved as structured data and the fields are passed to the next function.
- Page segmentation (F2): this function uses general knowledge of HTML tag usage and site specific knowledge of HTML usage, if available, to parse a page into "lines", which can be a line, a table row or an item of a list. It consists of four main steps: 1) representing a page as a character list, 2) rewriting the list using two tokens: tags and text, 3) classifying the tags into groups according to the page structure they represent including "word", "line", "paragraph", 4) segmenting a page into "lines".
- Knowledge unit extraction (F3): this function uses either domain knowledge of knowledge unit identification and extraction or site specific information extraction patterns for knowledge units, if available, to extract knowledge units from each "line".
- Knowledge unit grouping (F4): this function groups knowledge units into concepts and it uses two kinds of knowledge
  - either general knowledge of HTML usage or site specific knowledge, if available, of HTML usage AND
  - either domain knowledge of a concept hierarchy or site specific knowledge, if available, of a concept hierarchy.

6 Experimental Results

Our first information agent CASA (Classified Advertisement Search Agent) was built in 1997 [3] to search online real estate advertisements and help users to find rental property. It successfully searched for information automatically from multiple Web sites. It performed better than local search engines based on keyword matching.

An agent shell was developed based on the generalization of our first agent. The reusable agent framework, including the functions for Web accessing, information extraction and matching, forms the main part of the agent shell. The knowledge bases are completely separated from the framework and only the general knowledge base forms part of the agent shell. The other knowledge bases are kept separate from the shell. New information agents can be built by adding a new domain knowledge base and site specific knowledge base to the agent shell. The agent shell was successfully used to build a car classified advertisement search agent and a soccer score search agent [4].

Our experiments on building agents based on the framework show that:

- Our agent shell can be used to build information agents for multiple domains and multiple sites. Our agent can be easily adapted or extended by modifying or extending its knowledge bases, while most current information agents are tailored to one specific domain and are difficult to scale up.
- Our agents built using the agent shell accept user queries written in restricted natural languages, since the information extraction engine can extract specific requirements from the user query using the same method for extracting structured information from Web pages. The interface of our agents can be

as simple as that of keyword search engines with one single text input field. The interface is easy to generate and easy to use. The interface does not need to change for different domains. This differs from current local search engines, in which different user interfaces need to be designed for different domains.
- The information agents generated using our agent shell show better performance than local search engines based on keyword matching. The reason is that the information extraction engine transfers both the query and Web pages into structured data represented as a set of knowledge units. The matching is carried out between knowledge units which is more accurate than keyword matching.

In order to evaluate the agent's performance on information extraction from Web pages, we tested our agent on Web pages downloaded from over 100 Web sites. This paper will give some results based on our basic corpus. Our basic corpus was built by down-loading Web pages from 24 Web sites, 12 in the real estate advertisement domain and 12 in the car advertisement domain. Most of the Web sites are chosen from the top sites indexed by the search engine LookSmart at http://www.looksmart.com.

We use two parameters widely used in information extraction, precision and recall to evaluate our system. Precision is the percentage of correct responses out of all responses. Recall is the percentage of correct responses out of the total of correct answers. For each page, the information extraction answer keys are generated by manually correcting the output of our system. The performance of our system is evaluated by comparing the output with the answer keys.

In order to evaluate the performance of different steps of information extraction, we calculate precision and recall for the extraction of knowledge units, knowledge unit groups, and concepts.

- knowledge unit. Each knowledge unit is correct if its name and value are the same as that of the manually generated answers. The precision and recall of knowledge units indicate the ability of extracting individual knowledge units from Web pages.
- knowledge unit groups. We define a knowledge unit group as being correct when the correct knowledge units have been put in the right concept (group), ignoring false positive or false negative knowledge units. The precision and recall of knowledge unit groups indicate the ability of grouping knowledge units into concepts.
- Concept. A concept is considered correct when its all knowledge units at the lower levels are correct, that is, the concept is perfect, all of its knowledge units are extracted and all extracted knowledge units are correct. The precision and recall of concepts indicate the ability of extracting a "perfect concept".

The results are given in Table 1. The results show that our agent performs well on multiple Web sites, including Web sites with flexible data formats such as data presented as free text in paragraphs.

Table 1. Information Extraction Results

Domain	KU[a]		KU Groups		Concept	
	P[b]	R[c]	P	R	P	R
Reales[d]	87-100	76-100	75-100	75-100	66-100	60-100
Car[e]	92-100	95-100	94-100	97-100	56-100	56-100

[a] KU: Knowledge Unit
[b] P: Precision
[c] R: Recall
[d] Reales: Real estate advertisement
[e] Car: Car advertisement

7 Conclusion

This paper introduces a framework for building knowledge-based information agents. The knowledge-based approach that separates knowledge bases from other processes supports easy agent generation and adaptation. A new agent can be generated by adding new knowledge bases and an agent can be adapted by changing the knowledge bases. The classification of knowledge into three categories enables knowledge reuse and sharing. The general knowledge base is completely reusable and is built as part of the agent shell. The domain knowledge need to be changed for new domains and the site specific knowledge needs to be extended or learned for new Web sites.

This research focuses on semi-structured data and has been successful at extracting information from multiple Web sites in limited domains. With the rapid growth of the Internet, more and more services are become available online. Many of them present semi-structured data, for example, product catalogs, weather forecasts, phone books and stock market quotations. Our system is very useful for building information extraction systems for these online services. Users can generate their own information extraction system by creating knowledge bases and plugging them into our reusable shell. We believe this is much easier and faster than building a system from scratch.

We are currently working on how to learn part of the knowledge automatically. An algorithm is developed to learn site specific information extraction patterns from tabular Web pages [5]. Future work is needed to improve the learning techniques and to reduce manual work required for building and updating the knowledge bases.

References

1. Doorenbos, R. B., Etzioni, O.i , Weld, D. S.: A scalable comparison-shopping agent. In *Agent 97*, (1997)
2. Etzioni, O., Weld, D. S.: Intelligent agents on the internet: Fact, fiction, and forecast. *IEEE Expert*, 10 no. 4:44–49, (1995)

3. Gao, X., Sterling, L.: Classified advertisement search agent (CASA): A knowledge-based information agent for searching semi-structured text. In *The Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 621–622, London, UK, March 23-25 (1998)
4. Gao, X., Sterling, L.: A methodology for building information agents. In Yun Yang, Minshu Li, and Allan Ellis, editors, *Web Technologies and Applications, Asia Pacific Web Conference (APWeb'98)*, chapter 5, pages 43–52. International Academic Publishers, (1998)
5. Gao, X., Sterling, L.: AutoWrapper: Automatic wrapper generation for multiple online services. In Gilbert H. Young, editor, *World Wide Web: Technologies and Applications for the New Millennium*, chapter 8, pages 61–70. C.S.R.E.A. Press, (2000)
6. Hsu, C.-N.: Initial results on wrapping semistructured web pages with finite-state transducers and contextual rules. In *AAAI'98 Workshop on AI and Information Integration*. (1998)
7. Kushmerick, N.: Wrapper induction: Efficiency and expressiveness. *Journal of Artificial Intelligence*, 118:15–68, (2000)
8. Loke, S. W., Sterling, L., Sonenberg, L., Kim, H.: Aris: A shell for information agents that exploit web site structure. In *PAAM'98*, pages 201–219, London, (1998)
9. Muslea, I., Minton, S., Knoblock, C.: A hierarchical approach to wrapper induction. In *The 3rd conference on Autonomous Agents(Agent'99)*, (1999)
10. Sterling, L.: On finding needles in WWW haystacks. In *the Tenth Australian Joint Conference on Artificial Intelligence(Lecture Notes in Artificial Intelligence 1342)*, pages 25–36, Perth, Australia, 30 November-4 December (1997)

Designing Perception Modules to Shape Information for Agents

Clint Heinze[1,2], Adrian Pearce[1], Leon Sterling[1], and Simon Goss[2]

[1] Department of Software Engineering and Computer Science
University of Melbourne
Melbourne, Australia
clint,pearce,leon@cs.mu.oz.au
[2] Air Operations Division
Defence Science and Technology Organisation
Melbourne, Australia
Clinton.Heinze,Simon.Goss@dsto.defence.gov.au

Abstract. The difficulty associated with placing intelligent agents in environments is in part one of designing the means by which they sense their world. If the world is complex, or if the information available in the world is in an inappropriate form, then a mismatch between the information in the environment and the information required by the agent can exist. Practical means of dealing with this mismatch are available but designing the system requires careful modelling. By modelling perception as the module within the agent responsible for sensing and making sense of the environment several advantages are realised. The explicit representation of perception allows consideration of the issues, affords software engineering advantages with respect to the specification and design of systems and is compatible with many accepted definitions of agency.

1 The Environment–Agent Information Mismatch

Agents must sense the environments in which they situated in order to act. Beyond simply sensing the environment, agents must often store manipulate, model, and abstract the information coming from their sensors in ways that allow action, reasoning, or otherwise functions. Developing a model of the environment to support reasoning requires a mapping from the information available in the world to the form required internally by agent. If the environment is sufficiently rich or if there is a significant mismatch between the information available in the world and that required by the agent this mapping is not straightforward. In practice this information mismatch is solved in a number of ways. Conceptually it is common to think of an 'interface' or an API to an agent that allows it to interface with other systems. This interface provides the mechanism by which data enters the agent and typically there will be some manipulation or translation of data within this interface. This model of situating an agent in an environment is suitable if the information mismatch is small.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 239–248, 2001.

There are other ways of dealing with the mismatch however. It might be possible to design into the environment information structured in the form required by the agent. Meta-data within web pages is an example of an effort to provide constructs in the environment in a form appropriate for information agents reasoning. Just as the environment can be pushed conceptually closer to the agent so too can the agent be pushed closer to the environment. Sometimes this may be a compromise in agent design [13], or it may force the design of agent architectures that are designed to function directly with the information in the environment. An extreme case of this is Brooks' robots that utilise a subsumption architecture that provides the capacity to reason with virtually no representation of the environment at all [3]. The mismatch can removed if all of the system components utilise the same set of concepts. This drives research into "standard" or sharable ontologies. Finally a sophisticated mapping function can be added to convert from the form of the information available in the environment to the form required by the agent [7].

Within this landscape of agent–environment possibilities lies a design space that requires careful modelling. The task is to determine the concepts required by the agent and those that are to be made explicit in the environment. If these are different—as they almost inevitably are—then some way of resolving this mismatch is required.

Several design considerations are important in determining the nature of the solution. The following section provides a model of agency that addresses some of these issues. Later sections provide examples of agent systems that illustrate some of the issues.

2 Agents, Perception, and Agent Oriented Design

This paper addresses the problem of the information mismatch by explicitly modelling the function of sensing and representing the environment as a module within the agent—the *perception* module. This explicit modelling of perception as a module of an agent provides design and implementation time advantages for dealing with the agent–environment information mismatch. Not only is this view of perception in keeping with the anthropomorphic nature of the agent metaphor but it is compatible with most commonly accepted definitions of software agency. It is also in keeping with previous work where a module was added to provide the sub-cognitive aspects of the behaviour of an agent where the cognitive part was provided by a BDI agent [6].

A number of methodological approaches for the design of agent oriented systems have been suggested. These typically provide a set of base concepts that are used to provide the means by which specification and design of the agent system is undertaken. The Gaia approach [17] suggests that the use of interactions, roles, services, and acquaintances. Another approach [12] is to model agents as systems of beliefs, goals, and plans. Earlier work by Burmeister employs modifications of object oriented CRC methods and defines an agent model, a collaboration model, and an organisation model. In moving toward a unifying view of agent

oriented software engineering this paper poses the following broadly accepted[1] definition of agency.

An agent is *an autonomous entity that can perceive and effect the environment in which it is situated.*

Developing unified, or at least widely applicable agent oriented software engineering methodologies should address these general properties of agency before progressing to the more complex issues of team-work, roles, skills, goals, capabilities, and the like. The following sections present a preliminary view on the role of *perception* in the design of agent oriented systems. The explicit modelling of perception as a module of the agent allows a detailed focus on the interface between agent and environment that might otherwise can be problematic.

By considering the nature of the concepts that should be reasoned about, those that should be represented in the environment, and those that can be perceived an iterative design process makes explicit the important design choices. Typically the designer will iterate amongst these activities within the constraints of the design space to arrive at a solution.

Designing the flow, abstraction, and representation of information with an agent system is an important part of the overall system development process. Generally more complex environments will require more sophisticated sensory systems. Agents operating in the real world—robots—generally require sophisticated vision systems or location sensors. Agents that inhabit virtual worlds require simpler sensory systems. The behavioural complexity of the agent is also a factor. Agents required to exhibit very simple behaviors need little information from the environment and thus need little in the way of sensory apparatus even though their environment might be quite complex.

By explicitly representing perception it is possible to consider, in an abstract way, the concepts relevant to a system and to address questions that arise.

1. What concepts are usefully represented within the agent?
2. What concepts are usefully represented within the agent's environment?
3. What technology is available to sense the environment?
4. If there is a conceptual mismatch, what perception capabilities are available to perform the mapping.
5. What types of agent architectures and languages will be used?
6. What types of behaviour is the agent capable of?

The three sections below consider the design of the agents knowledge base, the design of the perception module and the design of the environment. It will be necessary to iterate these design activities to arrive at a solution.

2.1 Designing Agent Knowledge Bases

The *knowledge base* of an agent reflects in part its model of the environment in which it is situated. Different agents implement different representations of

[1] This definition is a synthesis of a number of quite suitable definitions provide by notable researchers. Definitions of agency by Wooldridge [11], Parunak [14], and Shoham [16] are examples.

the world. These can range from the complete absence of representation [3,4], through simple propositional logic, to first order logics of the type implemented by BDI agents [15].

The design process must determine the concepts about which an agent should reason. For the purposes of this paper primary consideration is given here to those concepts that in some way reflect the external world and are thus related to perception but the task of determining the concepts that reflect internal states of the agent are no less important.

Determining a useful and relevant set of concepts depends upon a number of factors: the nature of the domain, the type of agent, the information available from the environment, and the detail of the reasoning undertaken by the agent. This section presents three different views that can be taken to assist in the design of the knowledge base of an agent.

The selection of the concepts represented in the agent should reflect the general abstract level of its reasoning. It is generally undesirable to mix levels of abstraction in a module. Thus the concepts that represent the agents model of the world should loosely match other aspects of reasoning. By adopting different views it is possible to gain insights into the types of concepts that might be relevant.

A Domain Focussed View: this view of agent knowledge base design considers the general domain and the type of agent being constructed and poses the question — "What types of beliefs are relevant for the things that this agent is modelling?"

An Environment Focussed View: this view of agent knowledge base design considers the environment in which the agent will be situated and poses the question — "What are the things in the environment in which the agent will be situated that the agent can have beliefs about?" This is different from the domain focussed view in that there are things within the domain that may not be represented within the environment that the agent must function in. This view is only possible if the nature of the environment in which the agent is to be situated is known.

An Agent Focussed View: this view of agent knowledge base design considers the required behaviour of the agent being constructed and poses the question—"What types of beliefs are required to support the reasoning of the agent?" This approach has the advantage that only the knowledge required for the specific scenarios is generated and no time is wasted on superfluous design. This approach has parallels to use case analysis in object oriented programming [2] and forms the basis for a suggested methodological approach to the design of these types of agents [8]. A use case approach that is accompanied by extensions to the UML has been suggested for agent systems [9].

If the agent is required to simulate human reasoning, even if constrained to a specific domain, it is likely that taking the domain focussed view and considering the beliefs that might be possible in this domain becomes unwieldy. If

the environment into which the agent is to be placed provides the basis for the knowledge base design the outcome is likely to result in a loss of abstraction.

2.2 Designing Agent Perception

The perception module is responsible for mapping the information from the environment into the form required by the agents reasoning processes. In simple environments or where there is little mismatch this module will consist of little more than an interface specification. The design of this module must consider the types of knowledge required by the agent—the output of the knowledge base design process and provide them in the required form by converting from the data available. The exact functional requirements will vary across domains. The requirements of this process can be summarised as perceiving, recognising, and relating the features of the environment that are important to the abstract reasoning of the agent. This may include features in the environment, the activity of agents, including plan recognition, intention recognition, and team based plan recognition. Tidhar and Sonenberg provide a set of requirements on team based intention recognition that may in part be managed by this module and Heinze et. al. [10] provide a domain based view from a military simulation perspective of the types of features that might be recognised.

The agent knowledge base design is likely to generate requirements whereas the perception module design is likely to generate constraints. To develop an implemented system requires that technology be available to meet the design requirements. Examples from robotics of sophisticated perceptual systems are common. In simulation Heinze, Goss, and Pearce [7] adapted a pattern matching algorithm to provide a perception for an agent in a flight simulator. Similar work in plan recognition, object recognition, and intention recognition in agent systems will increase the feasibility of developing perception modules capable of bridging larger conceptual gaps. Generically there are a number of information processing functions that the perception module may subsume. Firstly the environment must be sensed and data extracted. It may be necessary to transform the data somehow. For example an axes transformation or a language translation. When multiple sensors provide data about a single object it might be necessary to undertake some sort of sensor fusion. Perhaps the robot sees the location of the wall and feels the location of the wall. It makes sense to fuse all sensory data about the wall into appropriate forms. Conceptually the agent "perceives" the wall. Finally, the classification or categorisation of things that are perceived in the environment are forms of data abstraction and another type of processing that may be required.

A number of support functions that might also fall within the scope of perception. For example: observation, the scanning of a section of the environment for a particular feature; recognition, the discovery of some familiar pattern in the environment; prediction, the ability to 'know' what is about to happen based upon the past.

2.3 Designing the Environment

In certain cases it is possible to design the environment in which and agent is situated. If this is the case it may be possible to remove the need for the perception module, or at least simplify it, by designing in properties of the environment that are *agent-friendly* in that they ease the task of perceiving that world. In many cases the environment cannot be designed to aid the reasoning of agents. For humans and robots the world can be bent and constrained but not radically altered. Most robots will only operate in fairly limited environments that can to some extent be designed but fundamentally those environments must be sensed.

In the case of the web it is possible to influence, even if only in small ways, the nature of the environment in which the agent exists. The XML standard is in part an attempt to create an environment that is designed to suit the requirements of information agents. Within small intra-nets it is possible to dictate in even more specific ways the form of web pages thus providing the necessary informational content.

In the case of virtual worlds inside simulators or games there is potential to exert even larger control over the design of the environment. It may be possible to add representations to the environment that assist agent perception. Other agent environments are exploring notions of labelled environments for this very purpose [1].

If the environment offers some level of design freedom then it is important to consider the nature of the useful representations that should be built into the world. The challenge of modelling perception does not get any simpler because it is moved into the environment model. The design of environments that support agents by providing direct access to useful conceptual structures raise becomes tightly coupled with the design of the agents. Information agents that rely on tags on web pages will perform better in sites where good meta-data is provided but will likely be out-performed by agents that do not rely on tags in sites that have misleading or no meta-data. There is a sense of surrendering some of the autonomy with which agents are endowed to the environment and in an implementation sense this is true. Agents are not autonomous in a pure sense but are 'situated in' environments and thus substantially linked to them. A choice to move aspects of the agent's perception to the environment still results in an agent that is autonomous within the meaning of 'situated'.

3 Three Examples

Consider the following examples that illustrate some points on the landscape of possible agent/environment design challenges.

3.1 RoboCup Robots

By utilising high level languages the designers of RoboCup Robots can take advantage of conceptual constructs within the languages that afford easier design

and implementation. This is particularly true when social aspects such as team play are important and the need to reason about the actions of others.

If a robot wishes to simply move toward the ball then there may be no requirement to explicitly represent the concept "ball" within the agent. Simply adjusting gains an a simple *sense-act* control system may suffice. But if a robot wishes to reason about the ball then it must first sense the ball and then characterise it in a useful manner[2]. The same is true of opponents. Not only must they be sensed but they must be represented for the purposes of reasoning. This representation may be simple and involve characterisation of their position, and the direction of motion—concepts that are reasonably apparent in the environment. The representation may also be more complex and may involve characterisation of aggregated or abstracted aspects of the environment. For example an agent might *see* a defender. *Defender* is an abstract concept that is not manifested in any particular physical difference between robots but is an inferred concept based on observed behaviour. This illustrates the mismatch. The agent might require a concept like "defender" to manipulate for the purposes of reasoning. This concept is not explicitly available in the world and must be somehow inferred, mapped, calculated, extracted, or abstracted.

Alternatively it is possible to alter the environment to more directly support the reasoning of the agents. Colored patches painted onto robots can make the task of categorising team members and opponents simpler. Coloring robots is in part an attempt to represent in the environment a concept—team—that the robot requires and thus reducing the information mismatch.

3.2 Information Agents

The Web is a significantly simpler environment than the natural world. Information agents that traverse the web need simpler sensory apparatus. Typically an HTML/XML parser is the sensory apparatus of the information agent[3] and much simpler than the machine vision of robots. Information agents that provide very simple search capabilities may require very little modelling of the web, the domain of interest or of the web pages that are found. They simple return the URL of a page that contains a matched substring. More sophisticated agents [5] will utilise some form of intelligent reasoning to more accurately assess the suitability of discovered pages. These types of information agents may require the capacity to reason about the nature of a web page found, or to model websites to provide an indication of the type of site or page that has been found. These more intelligent information agents may reason about concepts that are specific to the search domain. For example the 'Sportsfinder' [18] requires some

[2] Not all robot systems represent concepts for the purposes of reasoning. Brooks' robots inhabit complex dynamic worlds that must be sensed. Their subsumption architecture provides the ability to act with little or no representation of the environment.

[3] Some web based information systems function by rendering the page and parsing the resultant bit mapped image.

knowledge of the general domain of sport to assist its reasoning. They may also require some general knowledge about the likely structure of internet sites, and the formatting of pages so that they can characterise information based upon its location on a page or in a site. Supporting this type of reasoning requires that information that is discovered on the web can be represented in a useful form. It may be relevant for an information agent to classify a web site as being an academic site, a business site, or a personal site. A human browsing the web can make this judgement almost instantly but providing the same perceptive insight for an agent is more difficult. The XML and meta-tags are attempts to provide a mechanism for characterising web pages in a way that is useful for search engines and intelligent information agents. In this sense it is much like the painting of robots with red or blue circles to aid perception.

3.3 Computer Generated Opponents

Inside military simulators are models of the physical systems and models of the human behaviour. Intelligent agents are used to model human behaviour with more conventional technology for the aircraft, missiles, radars, and other sensors. These models typically provide some form of structured data that feed the intelligent agent a representation of the information that would be available to the pilot of a real aircraft from their sensors.

Systems such as SWARMM [13] utilise the BDI agent model to implements a high level means-ends reasoning that provides an appropriate level of abstraction for modelling human and human-like decision making. This level of abstraction is one of the strengths of BDI agents. It allows the analyst and designer to consider models based on beliefs, plans, goals, and intentions. This representation has utility in easing the knowledge acquisition task by reducing the semantic distance between representations in the heads of domain experts and the finished software. BDI agents manipulate abstract symbolic representations of knowledge in the form of *beliefs* about their world. The abstract nature of the beliefs and the type of reasoning produces an almost inevitable mismatch when these types of agents are placed into complex environments.

In human-in-the-loop (HIL) simulators and computer games the user is presented with a rendered three dimensional display of the world. Agents inhabiting the same simulators get a much different view[4]. When designing these simulations there is a greater degree of design control available over the environment in which the agent is situated than in the case of the virtual world of the Web or the real world of robots. With the structure of the environment and the agent both under design control the possibility for tailoring the environment for the reasoning of the agents exists.

[4] It might be possible to model perception in these systems by rendering the visual scene that would be available to the agent and then providing the agent with machine vision to decode the signal into its constituent parts. Common sense and engineering pragmatism dictate that this is not common practice and in its place there is some model of perception that takes data streams from the other components of the simulation and maps these into the agent.

Skillful experienced pilots, whether in a simulator or flying in the real world, intuitively recognise situations as they happen. Providing this capability to agents requires modelling of perception. A skilled pilot will recognise that the geometry of two "blips" on the radar screen indicates a certain type of behaviour—perhaps a formation adopted just prior to an attack. This recognition is subcognitive and requires no real thought. When designing an agent system that can exhibit this type of behaviour there are three options. The agent might be supplied with 'raw' data from the model of the radar which it must correlate and reason about. The agent might be supplied with a perception module that performs this task independently of its reasoning [6]. In keeping with the notion of 'tagging' web pages for information agents or colored circles for robots it is possible to *label* the virtual world to assist the task of perception.

4 Summary and Conclusions

Agent systems, by their very nature, presuppose a level of situatedness, autonomy, modularity, and, often, intelligence. Furthermore, agents are often embedded in existing environments. These features make likely the existence of a mismatch between the conceptual representations of the agents and the information residing in the rest of the system. The decisions about which concepts are useful to represent is a software engineering issue with implications for research into ontology and knowledge engineering.

The information mismatch can be overcome by moving the agent's ontology closer to that of the environment, as in Brooks' robots. By explicitly representing concepts in the environment that are suitable for the agent, as in tagged web pages. Or by providing a sophisticated perceptual system to perform the mapping [7].

A model of agent systems that includes a perception module allows for a model of these issues that spans the design space and allows for better management of software engineering, knowledge engineering and ontology design issues.

References

1. G. Au, S. Goss, C. Heinze, and A. Pearce. Rescuemodel: A multi-agent simulation of bushfire disaster management. In *Proceedings of the Roborescue Workshop, Boston*, 2000.
2. G. Booch, I. Jacobsen, and J. Rumbaugh. *Object-Oriented Analysis and Design with Applications*. Addison Wesley, third edition, 1999.
3. R. A. Brooks. Intelligence without reason. AI Memo 1293, MIT, Cambridge, Massachusetts, August 1991.
4. R. A. Brooks and L. A. Stein. Building brains for bodies. AI Memo 1439, MIT, Cambridge, Massachusetts, August 1993.

5. Xiaoying Gao and Leon Sterling. Semi-structured data extraction from heterogeneous sources. In David G. Schwartz, Monica Divitini, and Terje Brasethvik, editors, *Proceedings of 2cd International Workshop on Innovative Internet Information Systems (IIIS'99), in conjunction with the European Conference on Information Systems (ECIS'99)*, Internet-based organisational memory and knowledge management, pages 83–102, Copenhagen, Denmark, 1999. The idea group Publisher.
6. C. Heinze, S. Goss, I. Lloyd, and A. Pearce. Collaborating cognitive and subcognitive processes for the simulation of human decision making. In *Proceedings of the Fourth International Simulation Technology and Training Conference, (SimTecT '97)*, Melbourne, Australia, 1997.
7. C. Heinze, S. Goss, I. Lloyd, and A. Pearce. Plan recognition in military simulation: Incorporating machine learning with intelligent agents, 1999.
8. C. Heinze, M. Papasimeon, and S. Goss. Specifying agent behaviour with use cases. In *Pacific Rim Workshop on Multi-Agents, PRIMA '00*, 2000.
9. C. Heinze, M. Papasimeon, and S. Goss. Specifying agent behaviour with use cases. In *Proceedings of Pacific Rim Workshop on Multi-Agents, PRIMA2000*, 2000.
10. Clinton Heinze, Simon Goss, and Adrian Pearce. Plan recognition in military simulation: Incorporating machine learning with intelligent agents. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Workshop on Team Behaviour and Plan Recognition*, pages 53–63, 1999.
11. N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development, 1998.
12. D. Kinny and M. Georgeff. Modelling and design of multi-agent systems, 1997.
13. D. McIlroy, B. Smith, C. Heinze, and M. Turner. Air defence operational analysis using the swarmm model. In *Proceedings of the Asia Pacific Operations Research Symposium (APORS'97)*, Melbourne, Australia, 1997.
14. H. Parunak and R. Savit. Agent-based modeling vs. equation-based modeling: A case study and users' guide, 1998.
15. A. S. Rao and M. P. Georgeff. Modeling rational agents within a bdi-architecture. In *Second International Conference on Principles of Knowledge Representation and Reasoning*, San Mateo, CA, 1991.
16. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
17. M. Wooldridge, N. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design, 2000.
18. A. Wyatt, L. Sterling, and H. Lu. Sportsfinder: An information agent to extract sports results from the world wide web. In *Proceedings of the Forth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'99)*, pages 255–265, Lancashire, UK, 1999. The Practical Application Company Ltd.

Design of a Visualization Agent for WWW Information

Hironori Hiraishi, Hiroshi Sawai, and Fumio Mizoguchi

Information Media Center,
Science University of Tokyo
Noda, Chiba, 278-8510, Japan
email{hiraishi, sawai, mizo}@imc.sut.ac.jp

Abstract. This paper describes an information visualization agent that realize retrieval to effective WWW information access. In this agent, the link structure of WWW is displayed in a 3-D hyperbolic tree in which the height of a node within the tree indicates a user's "interestingness". Here, interestingness is calculated by a fitting function between a page and user-supplied keywords, and this measure can be used to filter irrelevant pages, reducing the size of the link structure. Such agent functions are incorporated within our browser, allowing us to discover desired pages from a large web site incrementally. Relatively large web sites were selected to show the performance of the agent with improved accuracy and efficiency in WWW information access.

1 Introduction

It is natural to regard WWW as a repository for knowledge discovery on the Internet. At present, such information can be accessed by a browser itself or by using a keyword search function. However, it is difficult to discover desired web pages using these functions due to the following reasons.

- Lack of global view of WWW information
  There are no functions for viewing the overall structure of WWW information. Thus, a user must access individual pages one by one.
- Lack of query navigation function
  Although keyword search or database-like queries realize efficient information access, it is hard for users to provide reasonable queries in knowledge discovery processes.

Some solutions to the first problem have recently been provided. The hyperbolic tree, developed by Lamping[Lamping 95], is known among the human-interface community as a reasonable tool for providing a global view of WWW information. It presents a GUI for hierarchical information structure including WWW and allows displaying the whole information structure within a hyperbolic plane. It also supports focus of attention by mouse operation; thus a user can see the structure at any angle.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 249–258, 2001.

As for the second problem, an automatic or semi-automatic structuring method for WWW information has been reported in the database community. Ashish, *et.al.*[Ashish 97] and Atzeni *et.al.*[Atzeni 97] developed "wrappers" to structure multiple web sites and to construct a WWW information database. Such approaches are useful for effective information retrieval by integrating various Internet sources and realizing SQL-like queries.

Although the above successful results were obtained by distinct communities, we hope these can be integrated to give birth to supporting tools for knowledge discovery from WWW information.

A hyperbolic tree approach is a comprehensive visual tool, but it is difficult to access a large amount of information using mouse operation only. In contrast, a database approach realizes efficient information access. This paper conjectures that a combinatorial use of information visualization and retrieval functions can navigate users to their desirable pages with improved accuracy and efficiency in WWW information access.

We have organized this paper as follows: Section 2 describes features of our WWW visualizing agent. Section 3 explains implementation of our agent. Section 4 provides experiment results, Section 5 compared with related works and the final section contains our conclusions.

2 WWW Visualizing Agent

WWW visualizing agent (WVA) integrates structure visualization and retrieval of WWW information, providing interactive and incremental access to Internet sources. This agent has the following features:

1. Visualizing the whole structure of hypertexts
   In order to look over a hypertext structure, the structure is displayed as a hyperbolic tree where nodes of upper-level texts are put around the center and their size is relatively large. In contrast, nodes of lower-level texts are distributed at the corners of the display. A user can see a focal node and nodes around it easily, and can also take in the larger hierarchical structure at a glance.
2. Smooth change of focus
   Focus change is used to access nodes of lower-level texts. From the viewpoint of human cognition, this change is done smoothly, so that there is little gap between text transitions. When the focal node is specified by a mouse click, several displays (D_i) are created and appear as a sequence $D \rightarrow D_1 \rightarrow D_2 \rightarrow ... \rightarrow D'$ to show the continuous process of focus change. Mouse dragging allows us to arbitrarily change viewpoints, and this provides a direct manipulation interface to access nodes easily.
3. Attribute-value query on the semi-structured texts
   This extracts information like attributes and keywords from a *flat* text and realizes an attribute-value query. By parsing the text, such information can be expressed as follows:

$$tag_1.tag_2.\cdots.tag_n.s \in t$$

where t is a text, s is a string and tag_i is a nested tag. The expression is then matched with a user-entered query $<tag'_1, ..., tag'_n, s'>$. The matching is approximated based on a similarity operator $\sim$, and is substituted in the following conditions:

$$tag_1 \sim tag'_1 \wedge ... \wedge tag_n \sim tag'_n \wedge s \sim s'$$

This means that we introduce a similarity measure between two strings or tags and process a query by computing the fitness of texts to be retrieved based on this measure. Retrieved texts are ranked in this way.

4. Visualizing attribute values for each node
   Attribute values for each node have useful information to navigate users to their desired texts. These values are also displayed within a hyperbolic tree. Attributes such as the title of the text and the fitness to a query tell users which nodes should be accessed. By changing the focus based on these attribute values, users may easily access their desired texts.
5. Filtering texts based on attribute values
   While every node is displayed in the previous feature, this feature eliminates meaningless nodes by using a threshold to determine nodes to be displayed. Consider node (t) such that every lower-level node (t') of t does not have a reasonable attribute value (which is determined by the threshold). In this case, we remove t from the hyperbolic tree, and reduce the tree size. This allows users to focus on interesting nodes.

The above features are interconnected. In particular, features (3), (4) and (5) are repeatedly executed. Through such a process, a user progressively explores the desired texts. This is a distinctive feature of our agent.

Our agent accelerates this interactive aspect by cooperating with a browser because feature (2) above is inter-related with the browser. After a node is mouse clicked, focus is changed, and the corresponding text is displayed in the browser. In the same way, when a user browses a text, this text becomes the focus within a hyperbolic tree display. This type of interaction helps users to understand which portion of hypertext structure is focused on.

3 Implementation of WWW Visualizing Agent

Fig. 1 shows the WVA agent architecture. WVA takes the hypertext source sent from a WWW server and accesses the server log for input information. This information is processed by tag and string parsers, and is used in modules such as the structure visualization. We also developed a browser that interacts with the visualization module. All modules except for the string parser are implemented in Java language, enhancing the portability of WVA.

The WVA output is shown in Fig. 2. A window is decomposed into a display for the structure of hypertext and areas for inputting queries and commands. Each text is displayed as a node whose height indicates node information such as user-access count and fitness to a query. The height is a good indicator for efficiently accessing interesting text.

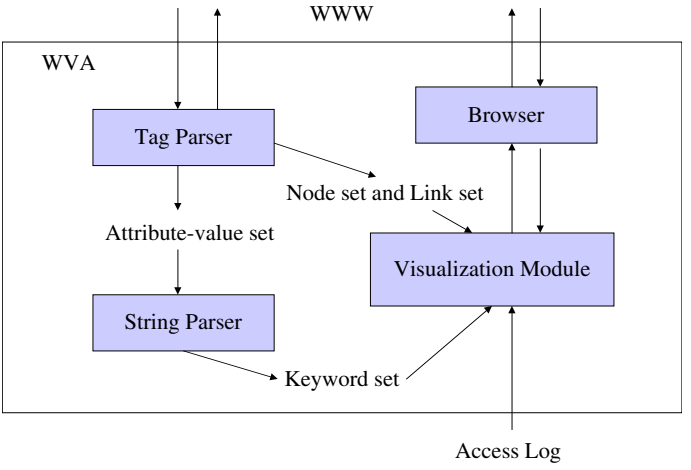


Fig. 1. Architecture of WWW Visualizing agent

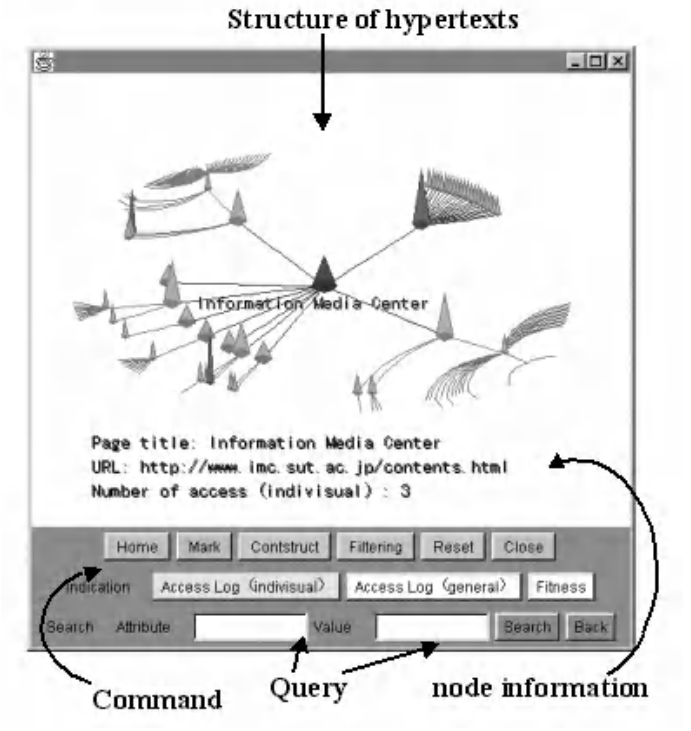


Fig. 2. Output of WWW Visualizing agent

A hyperbolic tree can be changed arbitrarily by mouse operation. Focus is changed by clicking the mouse on a node. Mouse dragging can be accepted at any position, making it easy for users to change the viewpoint of the tree.

3.1 Tag Parser

Tag parser parses the hyper-text specified URL and extracts an attribute-value set constructed by tag and string. A tag indicates an attribute name, and a string is regarded as an attribute value. WVA does not parse all tags provided in HTML; it only extracts tags to make a document structure. We selected only the following tags as attributes: title tag (<title>), headline tag (<h1>,...,<h6>) and item tag (<ul>, <ol>, <dl>,..). In XML, we can regard a tag as an attribute, so it is easy to extract the pair of attribute-value. In order to deal with HTML like XML, we extend title tags and item tags to specify an option representing semantics of a tag:

<h1 sem=purpose> str </h1>

Though the browser ignores the sem option, the tag parser directly interprets the str as the purpose. In general, HTML cannot be structured automatically, but such an option helps to structure an HTML document.

The link relation among home pages is derived from the anchor tag. The tag parser parses hypertext and goes to the next depth of pages by extracting the anchor tag contained in hypertext. Hyperlink chain generally becomes very long. We set a depth limit for the tag parser so it will not search below the depth limit. We are also making a setting to read only one site.

The tag parser searches hypertexts in a width-first manner. To avoid loops, it keeps all nodes already found. If we adopt a depth-first search, we cannot generate a well-balanced tree.

3.2 String Parser

The string parser analyzes the string of attribute-value sets from the tag parser and divides the string into clauses contained in the dictionary. The clause cannot be divided more, so the string is replaced by the set of clauses. We derive only nouns. We just compare two strings to judge whether two attributes are the same or not.

3.3 Visualization Module

The visualization module extends a 2-D hyperbolic tree to a 3-D tree because node information is represented as the height of the node. A typical hyperbolic tree algorithm lays out nodes as a circle, but our algorithm lays them out as an ellipse. Thus, front nodes cannot hide rear nodes.

Hyperbolic trees put a start node on the center, and child nodes are arranged radially. This tree is constructed on the Euclidean plane. After all nodes of the

tree are reflected in the hyperbolic plane, they are reflected in the ellipse plane. The line linking each node draws a curve.

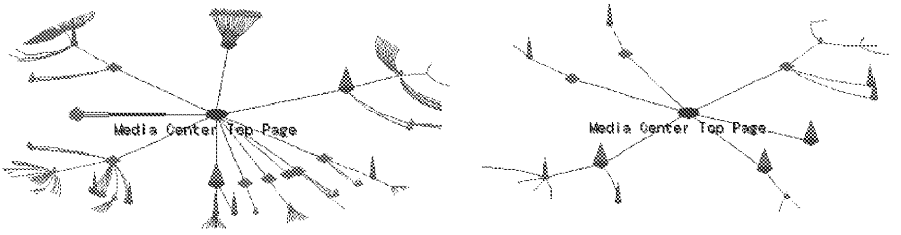
The process to display the tree structure is as follows:

1. The node (x, y) in Euclidean geometry is changed to hyperbolic coordinates $(x, y, \sqrt{x^2 + y^2 + 1})$.
2. The hyperbolic coordinates of the node and the point (0, 0, -1) are linked, and the node is reflected onto the ellipse plane. The coordinates become $(ax/(z + 1), y/(z + 1))$.
3. In order to link each node, the hyperbolic coordinates of the node are reflected onto the plane (z=1). The set of points on the line is calculated and reflected onto the ellipse plane.
4. A cone of each node is represented as the reciprocal of the distance from the center. The height is decided by the attribute value of a specified parameter.

The focus is moved by mouse dragging. The drag events are reported one by one, so the transformation of the hyper text structure is smooth.

3.4 Reducing the Tree

Filtering uninteresting nodes is the most important feature of WVA. Given a query, the fitness to the query for each node is computed and is displayed as the height of the node. The filtering function then removes nodes that have lower fitness, and restructures a reduced hyperbolic tree. This is very useful for large web sites because users can focus on interesting texts only.



(a) Before filtering (b) After filtering

Fig. 3. Filtering process

Fig. 3 shows a filtering process in WVA The left figure is a hyperbolic tree of our university research division web site (Science University of Tokyo, Information Media Center, http://www.imc.sut.ac.jp/) consisting of 226 texts. Since the height of each node indicates the fitness of the associated text to a user-supplied query, higher nodes are interesting for the user. There are nine nodes among a large number of nodes in the figure.

The right figure shows a reduced hyperbolic tree. Even uninteresting nodes that are reachable to interesting nodes still exits in the hyperbolic tree. The reduced tree constructs a web site for the user and allows the user to see the manageable-sized web site. Although existing search engines list up pages that are interesting for users, it is impossible to see the relationships between the searched pages. In contrast, a hyperbolic tree representation allows users to capture the web page structure and makes it easy to find interesting portal sites. Moreover, some queries can be put incrementally, and more interesting pages can be explored.

4 Experiments

We conducted two experiments in order to show the effectiveness of hyperbolic tree visualization. One experiment sought to show how the accuracy and efficiency change with the use of visualization. The other sought to show the effectiveness in long-term use.

A web site we used is our university research division site mentioned in Section 3.4. We selected as subjects ten bachelors belonging to our university who did not access the site before experiments. We divided the subjects into two groups; one uses a web browser with hyperbolic tree visualization and the other use the browser only. We gave each subject to 10 problems in which a hint is put as an abstract of a text to be found, and recorded passage time to access all texts and operation history. A problem was given, for example, "Search a text where you can see the picture of XXX professor in YYY workshop presentation". Since sufficient keywords were given, the subjects could find target texts correctly. Note that the possible operations the subjects used were mouse click for hyperlink selection, page back and forward, home position and bookmark registration.

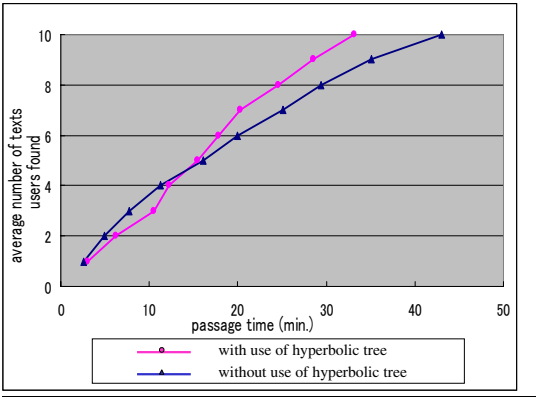


Fig. 4. Browsing performance with and without use of hyperbolic tree visualization

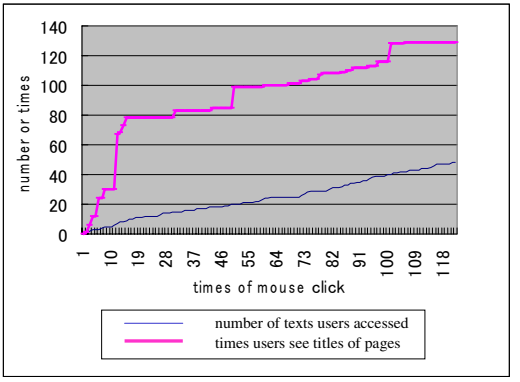


Fig. 5. Title reference in browsing

Fig. 4 shows the number of texts the subjects found. The graph indicates that use of hyperbolic tree visualization does not contribute efficient information access at an early stage, but achieves efficiency when 15 minutes passed. This means that global view function of hyperbolic tree visualization can assist users to find their desirable texts efficiently when a target web site becomes larger.

Checking subjects' histories about mouse operations is needed to clarify what is going on in browsing texts. WVA allows user to specify a node by mouse move and to see the associated text title. Fig. 5 shows how much times subjects see text titles in browsing about 50 texts. The total times was over 100 titles to see all the texts. At the beginning, the times increased dramatically, but were slightly changed after ten texts were found. This indicates that users saw the overall structure of the web site at the beginning and then found texts to be accessed. As a conclusion, the structure visualization based on the hyperbolic tree provides an efficient and exhaustive browsing function.

Our WVA was used in the project to stimulate and provided information for the underpopulated area of Nagao in Kagawa prefecture. The project was conducted for two years, but our system is still being used.

In this project, we focus on visualization and developed Visual Internet Positioning (VIP) that is implemented only by the visualization module in Fig. 1. VIP is shown in Fig. 6 and is implemented in Java Applet to run on a general web browser like Netscape. VIP starts by specifying the URL containing the VIP applet. VIP is on the left frame, and a selected home page on the VIP is displayed on the right frame. VIP is very simple to operate. Users click on a node of the hyperbolic tree or input a URL into the text field, then VIP displays the hyperbolic tree as the selected node in the center.

People can learn to use VIP very quickly because of VIP's simple operation and easy understanding due to the visualization. In this project, we tried to evaluate other systems. Since they required complicated operations and expression, they did not spread very quickly, thus VIP thus contributed significantly

Fig. 6. VIP(Visual Internet Positioning)

to this project. However, we also found that simple operation and visualization can cause the user to lose interest.

5 Comparison with Other Work

Ashish *et.al.*[Ashish 97] and Atzeni *et.al.*[Atzeni 97] have attempted to structure WWW information to support SQL-like queries. Their approach is to construct a database from multiple web sites using "wrapper" programs that deal with semi-structured information. Our framework does not compete with theirs but may exploit it in generating attribute-value pairs from WWW information. The difference is that a query in WVA is relatively simple because WVA just focuses on retrieving desired texts. Thus, we do not necessarily deal with a general relational model. This increases the efficiency of information retrieval by adopting special data structures.

Visualization methods for WWW information are recently proposed within a discovery science community[Hirokawa 98][Sawai 98][Shibayama 98]. However, integration with information retrieval is not realized. In our approach, a web browser was specially designed and implemented to communicate with visualization and retrieval, showing the performance of this integrated approach.

Munzner designed an information space in which multiple hyperbolic trees are configured three dimensionally [Munzner 98]. However, there are a number of nodes on the information space map, and it is hard to focus on the appropriate portion of a web site. An alternative visualization scheme was developed as a "cone" tree by Robertson[Robertson 91]. In a cone tree, nodes are distributed in a three-dimensional space, and thus front nodes may hide rear nodes. Moreover, focus change is not as easy as in a hyperbolic tree representation.

6 Conclusions

In this paper, we proposed a WWW Visualizing agent (WVA) that is the integration of structure visualization and retrieval of WWW information. The WVA consists of hyperbolic tree visualization and attribute-value pair query manipulation, and provides a filtering function to reduce the size of the WWW information structure. Experiments were conducted to show the advantage of the visualization. The obtained statistics demonstrate the effectiveness of WVA in accessing WWW information. Since WVA can be interconnected with our browser, the proposed framework will help users browse a large number of texts on the Internet.

References

[Ashish 97] N. Ashish and C. Knoblock, Wrapper generation for semistructured Internet sources, *Proc. of the Workshop on Management of Semistructured Data*, Tucson, Arizona, May 1997.

[Atzeni 97] P. Atzeni, G. Mecca and P. Merialdo, Semistructured and structured data on the Web: going back and forth, *Proc. of the Workshop on Management of Semistructured Data*, Tucson, Arizona, May 1997.

[Hirokawa 98] Hirokawa, S. and Taguchi, T.: KN on ZK – Knowledge Network on Network Note Pad ZK, *Proc. of the First International Conference on Discovery Science*, pp.411-412, 1998.

[Lamping 95] J. Lamping, R. Rao, P. Pirolli, A Focus+Context Technique Based on Hyperbolic Geometory for Visualizing Large Hierarchies, *Proc. of ACM CHI '95*, 1995.

[Matsumoto 99] Y.Matsumoto, The Japanese Morphological Analysis System ChaSen, *NAIST Technical Report, NAIST-IS-TR99008*, April, 1999.

[Munzner 98] T. Munzner, Exploring Large Graphs in 3D Hyperbolic Space, *IEEE Computer Graphics and Applications*, Vol. 18, No. 4, pp. 18-23, 1998.

[Robertson 91] G. G. Robertson, J. D. Mackinlay, S. K. Card, Cone Trees: Animated 3D Visualizations of Hierarchical Information, *Proc. of ACM CHI '91*, 1991.

[Salton 91] G. Salton, Developments in automatic text retrieval, *Science*, Vol. 253, pp. 974-980, 1991.

[Sawai 98] Sawai,H., Ohwada,H. and Mizoguchi,F.: Incorporating a navigation tool into a browser for mining WWW information, *Proc. of the First International Conference on Discovery Science*, pp.453-454, 1998.

[Shibayama 98] Shibayama,E., Yabe,J., Takahashi,S. and Matsuda, M.: Visualizing Semantic Clusters in the Internet Information Space, *Proc. of the First International Conference on Discovery Science*, pp.409-410, 1998.

Revisable Analysis and Design by Actors Interaction: Emergency Case Study

Hassen Kriaa and Guy Gouardères

Laboratoire LIA, IUT de Bayonne
3 Avenue Jean Darrigrand, 64100 Bayonne, France
{kriaa, Gouarde}@larrun.univ-pau.fr

Abstract. This paper describes a method of multi-agent analysis and design for reactive, real-time information systems, relating to complex and risks applications. The fundamental principle consists of using a series of models in "cascade" to go from an abstract representation of the problems to a formal one of the directly programmable agent (in Java for example). The first basic idea is not to have fixed goals or tasks, but rather for them to be gradually released from the analysis of the interactions between the actors (human or artifacts). The second idea aims at integrating the space-time constraints according to an individual and collective point of view. The last one proposes neither to process on a hierarchical basis nor to laminate the final architecture of the interactions between agents but to define the acquaintance rules and their evolution according to the context. This paper details the various stages of a Revisable Analysis and Design by Actors Interaction methodology (RADAI) and compares the issues with other current work.

1 Introduction

This paper describes the analysis and design of complex, hazardous process systems centered on the processes, based on a distributed architecture according to different point of view of various actors. A preliminary first approach of this problem can be found in Wooldridge [1] which presents an agent analysis and design methodology for distributed and evolutionary information systems. This method is strictly based on predefined agent's roles. Then it analyses the interactions between these various roles to take in account the collective aspect of the system.

Another point of view is developed by Kinny [2], with an agent methodology which use modeling techniques of individual agents based on beliefs desire-intention paradigm. Mixing the two way, it seems useful to define an hybrid approach (individual and collective) to design flexible but reliable systems software based on "intelligent" agents like those mentioned in [3] and [4]. According to the work of Jennings [3], it will be attractive to propose agent-based improvement of human-computer interaction in complex systems making them more safe, reliable and easy to use in critical or emergency situation. Brief extracts from the patient care process on the emergency chain at the hospital illustrate the method.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 259–268, 2001.

2 Why an Agent Oriented Method?

For several years, a great number of object analysis and design methods have been developed. We will refer reader to Cauvet [5] and Rumbaugh [6] work's for a fast survey of basic schemes and features. Most of these methods present as common drawbacks, the limits of the actors behavior representation, the knowledge re-use obtained by these actors (knowledge re-engineering) and a poor real time specification of space-time constraints.

An intuitive thinking about agent methods can be sum up as initially a reasoning step with phases, stages, etc., and then as dynamics which can reduce some of these inconvenient. To represent reasoning steps as an evolving process, we use some techniques and models providing coming out of dynamic associations (a real part of reasoning or "chunks "). These are evolutionary concepts and thus revisable to be encased by formalization steps, allowing the check of the coherence. Finally, we can implement them in agent form with functional programming tools such: a real-time object languages, or by means of logical objects (CLIPS) or a dynamic entities (SCHEME) or an agents framework such as Madkit [7].

3 States of the Art

Nowadays, agent design is a friendly paradigm for researchers and begins to be introduced in business software (as web). The role of an agent-oriented methodology is to assist and to manage agent application during all through its life cycle. In that way, literature relates on three approaches:

- Extension of object oriented methods and techniquesBurmeister [8], Kinny [1], [2]. This approach takes its advantage from the similarity between agents and objects and the experience of the objects technologies. The major disadvantage is not only the lack of the social dimensions of the agent but also the cognitive one (mental state).
- Extensions of the knowledge engineering methods, as in MAS-CommonKADS [9] which re-use the defined ontology library and tools, are released from knowledge engineering methods. The disadvantage of this extension is a centralized design and they don't tackle the social and distributed aspect of agent.

The third approach gathers together all others works as the use of a formal design (DESIRE framework) [10], hybrid methods based on dialogue analysis and interaction applied to medical domain or cooperative agent design.

4 General Framework

First, we admit that dialogue analysis between actors is a dynamic way to design human behaviour embedded in the global process. As a consequence, actor's activity in the dialogue can be defined as a generic unit of representation, with a distributed control and an assigned role to be a vector of interaction between the constraints of the system. The adaptation of the system to the user and vice versa is done by revision in

real time of scheduling and activity. Instead of analysing separately individual and collective dynamics system, we have adapted a convergent interactive analysis from Barber [11], to better articulate them, by including the distributed aspect of multi-agents concepts.

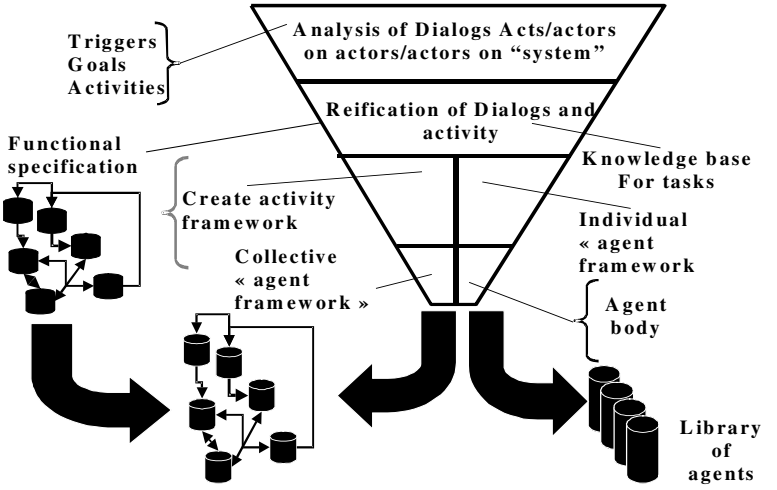


Fig. 1. Synopsis of RADAI

The starting point is based on the analysis of the dialogues for goals identification released from activities. The tasks identification and their gathering allow a dynamic design and a structural and functional composition of the system components. During this stage, the architecture and the component are revised by rules (conflict resolution) not as in a Guided Use Case [5] approach but with a Case Base Reasoning [12] solver.

4.1 Process

The first step in the method consists of dialogue analysis. In that way, the basic set of goal requirements needed by the system is previously defined by a heuristic and non-formal mapping of dialogue between identified actors. Then with a semi-formal checking we extract the exact role of missing triggers between the actors and the system. For each trigger we construct the integrated class of activity launched. Then we associate them adequate goals. An example is given bellow (see case study).

The second step brings up the reification of both dialogue and related activity. This stage can be decomposed in two sub-steps. The first sub-step allows us to construct scripts for each goal using the GOMS method [13] for each class of integrated activity. This sub-step begins the construction of the agent body. The second sub step is done in parallel way and detects the basic task connected to each activity. This sub step uses an ETAG grammar [14]. Next, we specify for each method, function or step

the target of needed objects, states, events, tasks and interaction diagram to start the construction of the collective model.

The third step is the design step. It consists of the construction of different entities needed by the system by aggregation and classification. At the next step, we use the STROBE model [15] to check the communication protocol between the different entities by an "one to one" mode to assert the collective dimension into each agent. Then we assigned for each agent the resource extracted from the different ontology of the system. These resources are differently instantiated according to the context. Finally the entire set of agents is implemented with Java. An example of such agents is given in the case study section. During the second and the third step some conflicts are detected so knowledge revision and control is done for each step to arise error.

4.2 Concepts

The different concepts used within this approach are "process", "trigger", "activity", "goal", "script", "task". They bring up together the individual and the collective model of agent and architecture. Here are some definitions of the basic concepts:

- Trigger: not only events that launch a group of activity to accomplish a goal, but also any change, which take a perceptible effect on the context.
- Activity: is a generic unit of representation. It can be considered as interaction vector between all the constraints of the system.
- Goal: a target for features which describe the objective to reach.
- Task: a set of specific goal fixed by the user who tries to achieve them.
- Agent: autonomous software by the meaning of the STROBE model.

4.3 Models

A "cascade" of constituent models steps the different stages into the method:
Activity model: to identify and integrate activities into the system. An activity is to be done by the actor in a conscious or unconscious way to accomplish goal. An activity is supported by a method specified in semi formal way using "GOMS".
Task model: this model allows us to describe the task composition assigned to one or more actor under a time constraint imposed by the speech act between actors. This task basis will be formally represented by an ETAG grammar.
Agent model (individual): it represents the basic features of an agent. Moreover, it includes goals, services, reasoning mechanism, communication modules, etc. This model provides a description for all agents in the system. An agent will be defined by all the tasks and services associated for each one. Each service will then be defined by means of its data and its pre- and post-conditions which represent the constraints.
Organizational model (collective): this model represents agent's organization. It describes the architecture which is made up of agents and the relations among them plus their environment. This model itself is composed of a coordination model and a communication model. The coordination model describes the various interactions among agents. Whereas the communication model details the human-computer interactions and the related constraints necessary for the development of the user interface. To

describe the various scenarios among the agents, the exchanged messages will be represented in order to build relationship (acquaintances). Those scenarios will allow the modeling of the event and data flows, the analysis of each interaction and the determination of the synchronization type (synchronous, asynchronous).

5 Case Study

To illustrate the complete process of the method, we will study the emergency health care (Case of Cerebral Vascular Accident). Within this case, the studied process has to cope with different actors: the doctor, the nurse, the reception and orientation nurse, laptops and the patient. All these actors cooperate together to accomplish the main goal: a good assumption of the patient care. The space work agency allows a hearing shared information due to the presence of the whole actors in the same time-space environment.

Model	Trigger	Constraints	Activity
M01	Previous or non previous patient admission GON Prefatory note	Urgency Time0	Reception Orientation File establishment
M02	Patient file emergency doctor	Primary-care room Time2 material available	Primary exam to do Result interpretation Nurse care patient
M03	Patient Nurse Material available appointment	reeducation room	Reeducation and care

Fig. 2. Trigger, Constraints, and Activity Sample

Actors can exchange dialogue and help each other. They can hear and see each other, take decision to call other actor, evaluate unavailability and make decision instead of other actor.

specification.Step 1: speech act analysis

Here we will construct manually the hole of trigger and associated activity, then we will release goals. For that we consider D = {D1, D4} the hole trigger with:

D1: "patient arrival alone or not "
D4: "patient and doctor meeting"
For these, trigger here has some associated activity:
For D1: A1: "reception of a patient"
A2: "orientation of a patient"
For D4: A4: "suggest some exam"
A5: "establish a diagnostic"
A6: "suggest a treatment"

Some released goals to accomplish are:

For D1 (A1, A2), the goal is "admission of a patient"
For D4 (A4, A5, A6), the goal is "patient take care"

Step 2: analysis

GOMS specification

This step is done to specify task body. GOMS is a tool for constructing the individual agent model and his body. To do that we consider:

Rules:
R2: rule to accomplish "admission of a patient"
If (new patient) then (accomplish "admission new patient")
If " old patient" then accomplish "admission old patient"
Return with goal accomplished
According to these rule we found some script:
S1: method for goal "admission new patient"
step1: social allowances (M1)
step 2: visual allowances (M2)
step 3: method for goal: "create file" (M3)
step 4: method for goal: "move to care unit" (M4)
step 5: validation (M5)
step 6: return goal accomplished (M6)
S2: method for goal "admission old patient"
step 1: social allowances (M1)
step 2: visual allowances (M2)
step 3: display old value (M7)
step 4: modify certain value (M8)
step 5: method for goal: "move to care unit" (M4)
step 6: validation (M5)
step 7: return goal accomplished (M6)

ETAG formalization

This formal approach describes how task interaction is independently triggered by events within the agent body. Then to model a collective framework, it associates an **Entry** to the event that generates acquaintances within the organizational model.

Type [object = identification]
Value set: traumatology | medical
End object
Type traumatology isa object
Value set: fracture | entorse | luxation
End traumatology
Type [object = box]
Value set: occupied | non occupied
End object
Type [object = file]
Value set: medical | social | follow-up
End object
EVENT]::= [event.NEW ([Patient])] | [event.update ([file])] | ....

type[EVENT > event.NEW ([Patient: *NP])]
precondition: [state.IS-AT ([Box: *B])] ;
clears: [state.IS-AT ([Box: *B])] ;
postcondition: [state.IS-AT ([Box: *B])] ;
end [EVENT]
type[EVENT > event.CLICK ([Box: *B])]
precondition: [state.IS-AT ([Box: *B])] ;
end [EVENT]

Entry 1:
[*task* > identification patient] , [*event* > patient], [*object* > patient = *P]
t1 [event > new patient],[OBJECT > patient = p]
"identification new patient"

Entry 2:
[*task* > transfer] , [*event* > patient and nurse], [*object* > patient = *P]
t2 [event > patient and nurse and appointment and available bad and update],[OBJECT > patient = p]
"moving toward patient from emergency to reception service "

Entry 3:
[*task* > box attribution] , [*event* > patient] , [*object* > patient = *P], [*object* > Box = *B]
t4 [event > patient], [*object* > patient = p], [*object* > Box = "non occupied "]
"attribute a box to a patient"

Entry 5:
[*task* > orientation] , [*event* > patient], [*event* > cause], [*object* > patient = *P], [*object* > Box = *B], [*object* > cause = *C]
t5 [event > patient], [*object* > patient = p], [*object* > Box = "non occupied"], [*object* > cause = c]
"move patient to box "

Related Agent Specifications and Implementation with CIAgent [20].

```
Public class AgentBox extends CIAgent{
        // agent name
         String name ;
        //acquaintance
        private Vector listeners = (AgentSaisieOrientation, AgentInformation,
        MedicalAgentSaisieOrientation)
        // task list
        private task-state-box () { specify the box state}
        private task-attribute-box () {attribute a patient to a box}
        private task-modify-state-box () {update box}
        };
Public class AgentSaisieOrientation extends CIAgent{
        // agent name
         String name ;
        // acquaintance
        private Vector listeners = (AgentRecherche, AgentBox, Medical-
        AgentSaisieOrientation)
        // task list
        private task-create-social-file () { create social file}
        private task-create-first-medical-file () { create medical file }
        private task-update-social-file () {update social file }
        private task-add-to () {add information to patient file}
```

```
        private task-get-first-medical-causes () {get entry causes}
        private task-move-to () {move to care unit}
        };
Public class AgentPathologie extends CIAgent{
        // agent name
         String name ;
        // acquaintance
        private Vector listeners = (AgentBox, MedicalAgentSaisieOrientation,
        AgentInformation, AgentDiagnostiqueTraitement)
        // task list
        private task-get-history () { get malady history}
        private task-get-clinical-exam () { get clinical information }
        private task-get-radio-result () { get radiological information}
        private task-get-analysis-result () { get exam analysis}
        private task-get-extend-exams () { get complementary exams}
        private task-create-medical-file () { create medical file}
        };
Public class MedicalAgentSaisieOrientation extends CIAgent{
        // agent name
         String name ;

        //acquaintance
        private Vector listeners = (AgentPathologie, AgentInformation,
        AgentSaisieOrientation)
        // task list
        private task-generate-dignostic () { generate a diagnostic}
        private task-generate- treatment () { generate a treatment}
        private task-generate-observation () { create observation}
        private task-generate- prescription () { create prescription}
        private task-learn () {to learn}
        };
```

6 Conclusion and Perspectives

The main purpose of this paper is the definition of an agent based stage methodology (RADAI) which is based on the upstream analysis of dialogue among the system of actors.

Using the RADAI method we assert as a significant result, the combination of the convergent ascending interactive analysis based on multiple points of view with the specification and the downward implementation of agents. This generates flexibility based on revisability and autonomy, instead of issues from the Jennings's work [1] that presents a strict and rigid methodology based on predefined agents through out their specific roles.

However, some of the advantages acquired by this flexibility of RADAI steps are restricted by the development (due to the sponsor requirements) of the issued agent architecture using the ABE platform [16] with Java. First, because the adaptation of agents by rules is supported by poor mechanisms of inference but mostly because the communication process between agents is acted with Java Beans. Thus, we may consider some improvements such as the use of a training mechanism to implement

agents, according to the issues on knowledge revision [17] by genetic algorithms, or those with mutant agents.

On the other hand, we noted that the used methods (GOMS and ETAG) to specify tasks and activities are too simple, too heavy and limited. Then we plan to adopt a more powerful and richer formalization method, such as the DESIRE framework of Brazier's work [10]. Moreover, agent must have a superior agility then the actual framework. For that, evolutionary or/and anticipative model developed by Ekdalh and Astor [18] explores different possible solution.

References

1. M. Wooldridge, N. Jennings et D. Kinny. a Methodology for Agent-Oriented Analysis and Design. Third International Conference on AUTONOMOUS AGENTS (Agents '99), Seattle, Washington, May 1-5 1998.
2. Kinny D., Georgeff M., and Rao A. (1996). A methodology and modelling technique for systems of BDI agents. In W. van de Velve and J. W. Perram, editor, Agent Breaking Away: proceeding of the seventh european workshop on modelling autonomous agents in a multi-agent world, (LNAI volume 1038), pages 56-71. Springer-Verlag: Berlin, Germany.
3. Jennings, N.R. et Wooldridge, M. (1995). Applying agent technologie. Applied artificial Intelligence, 9(6): 357-370.
4. Frasson C., Mengelle T., Aïmeur E, Gouardères G., «An Actor-based Architecture for Intelligent Tutoring Systems», Third International Conference ITS'96, Montreal, Lecture Notes in Computer Science, Heidelberg Springer Verlag, 1996.
5. Cauvet C., Rolland C. (1992) Object-Oriented Conceptual Modelling. CISMOD92.
6. Rumbaugh J. (1995). What is a method? Journal of Object Oriented Programming, 10-16.
7. Ferber, J., Gutknecht, O., (1999) Operational semantics of a Role-based Agent Architecture. In: Jennings, N.R., Lesperance, Y., (eds.) Intelligent Agents VI. Agent Theories, Architectures, and Languages, LNCS 1757, Springer Verlag (1999).
8. B. Burmeister. (1996). Models and methodology for agent-oriented analysis and design. In K Fischer, editor, Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems. DFKI Document D-96-06.
9. Iglesias C, Mercedes C., Gonzalez J.,Velasco J.. (1996). Analysis and design of multiagent systems using MAS-CommonKADS. In AAAI'97 Workshop on Agent Theories, Architectures and Languages, Providence, RI,
10. Brazier, F.M.T, Dunin Keplicz, B.M, Jenning, N.R. and Treur, J. (1997). DESIRE: Modelling multi-agent systems in a compositional formal framework. In M. Huhns, M. Singh, (Eds), International Journal of Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems.
11. Barber, K. S., Graser, T. J., Jernigan, S. R. and McGiverin, B. (1998). Features of the Systems Engineering Process Activities (SEPA) Methodology. AAAI's SIGMAN Workshop on Artificial Intelligence and Manufacturing, Albuquerque, NM.
12. Kolodner, J. L.: Case-based reasoning. San Mateo, CA: Morgan Kaufmann (1993
13. Card, S.K., Moran, T.P. and Newell, A. (1983). The Psychology of Human–Computer Interaction. Lawrence Erlbaum Ass., Hillsdale, New Jersey
14. De Haan, G. (1996). ETAG-based Design: User Interface Design as Mental Model Specification. In: Palanque, P. and Benyon, D. (eds.) Critical Issues in User Interface Systems Engineering. Springer Verlag, London, 81-92.

15. Cerri, S.A. (1999). Shifting the focus from control to communication: the STReams OBjects Environments model of communicatiing agents. Padget J.A. (ed.), Collaboration between Human and Artificial Societies, Coordination and Agent-Based Distributed Computing, Lecture Notes in Artificial Intelligence 1624, Springer Verlag: 71-101.
16. Bigus J.P and Bigus J. (1998). Constructing Intelligent agent with java. A programmer's guide to smarter applications. Wiley computer publishing, Toronto.
17. Gouardères G., Canut M.F., Sanchis E., (1998). From Mutant to Learning Agents. 14th European Meeting on Cybernetics and Systems Research - EMCSR'98, Vienna, Austria.
18. Ekdahl B., Astor E., Davidsson P., «Toward Anticipatory Agents», In M. Woolridge and N.R. Jennings, editors, Intelligent Agents - Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence 890, pp. 191-202, Springer Verlag, 1995.

A Logic-Based Approach for Adaptive Information Filtering Agents

Raymond Lau[1], Arthur H.M. ter Hofstede[1], and Peter D. Bruza[2]

[1] Cooperative Information Systems Research Centre
Queensland University of Technology, Brisbane, Qld 4001, Australia
{raymond, arthur}@icis.qut.edu.au
[2] Distributed Systems Technology Centre, The University of Queensland
Brisbane, Qld 4072, Australia
bruza@dstc.edu.au

Abstract. *Adaptive information filtering agents* have been developed to alleviate the problem of information overload on the Internet. However, the explanatory power and the learning autonomy of these agents can be improved. Applying a logic-based framework for representation, learning, and matching in adaptive information filtering agents is promising since users' changing information needs can automatically be deduced by the agents. In addition, the inferred changes can be explained and justified based on formal deduction. This paper examines how the AGM belief revision logic can be applied to the learning processes of these agents.

1 Introduction

Information filtering (*IF*) and information retrieval (*IR*) are "two sides of the same coin" [2]. However, IF is more concerned with the removal of irrelevant information from a stream of incoming information. With the explosive growth of the Internet and the World Wide Web (*Web*), it is becoming increasingly difficult for users to retrieve relevant information. This is the so-called problem of *information overload* on the Internet. Augmenting existing Internet search engines with personalised information filtering tools is one possible method to alleviate this problem. Adaptive information filtering agents [3,10] can *autonomously* filter the incoming stream of information on behalf of their users. Moreover, they are able to learn and revise their beliefs about the users' changing information needs. The **AGM** belief revision logic [1] provides a rich and rigorous foundation for modelling such revision processes. As the semantic relationships among information items can be taken into account during the belief revision process, it is possible for an agent to deduce a user's changing information needs. Therefore, less amount of users' direct relevance feedback [11] is required to train the filtering agents, and hence a higher level of *learning autonomy* can be achieved.

2 System Architecture

Figure 1 depicts the system architecture of an agent-based information filtering system. A user communicates with the system via the interface agent. The

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 269–278, 2001.

retrieval agents (RA) retrieve Web pages from external sources. The data management agent is responsible for house-keeping and characterisation of Web documents. The focus of this paper is on the **adaptive information filtering agent**. Particularly, the learning component of the filtering agent. Learning in the filtering agent consists of two steps. Firstly, based on a user's relevance feedback and the statistical data stored in the system's output queue, the learning component induces beliefs [6] about a user's information needs. Secondly, these beliefs are revised into the filtering agent's memory in a *consistent* and *minimal* way. The AGM belief revision framework [1] and the corresponding computational algorithms[14,15] provides a sound and rigorous mechanism to carry out such an operation. With a logic-based learning approach, a user's changing information needs can be automatically deduced by the agent. The matching component carries out the filtering function based on logical deduction. Web documents deemed relevant by the filtering agent are transferred to the output queue via the Data Management Agent.

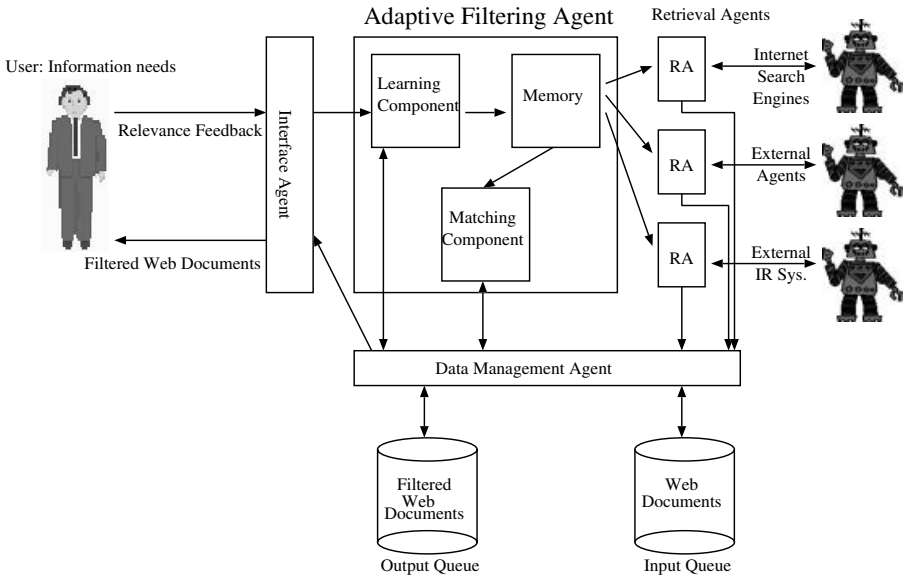


Fig. 1. Agent-based Information Filtering System

3 The AGM Belief Revision Paradigm

The AGM framework [1] formalises *consistent* and *minimal* belief changes by sets of postulates and belief functions e.g. *expansion* (K_α^+), *contraction* (K_α^-), and *revision* (K_α^*). One of the constructions of these functions is by *epistemic entrenchment* ($\leqslant$) [7]. Beliefs with the lowest degree of epistemic entrenchment are

given up when inconsistency arises because of applying changes. Nevertheless, for computer-based implementation, *finite partial entrenchment ranking* **B** which ranks a finite subset of beliefs with the minimum possible degree of entrenchment $\leqslant_{\mathbf{B}}$, and *maxi-adjustment* [14,15] which repeatedly transmutes **B** using an absolute measure of minimal change under maximal information inertia, have been proposed. Williams [14] has formally defined finite partial entrenchment ranking as a function **B** that maps a finite subset of sentences in $\mathcal{L}$ (e.g. classical propositional language) into the interval $[0, \mathcal{O}]$, where $\mathcal{O}$ is a sufficiently large ordinal such that the following conditions hold for all $\alpha \in dom(\mathbf{B})$: (PER1) $\{\beta \in \mathrm{dom}(\mathbf{B}) : \mathbf{B}(\alpha) < \mathbf{B}(\beta)\} \not\vdash \alpha$; (PER2) If $\vdash \neg\alpha$ then $\mathbf{B}(\alpha) = 0$; (PER3) $\mathbf{B}(\alpha) = \mathcal{O}$ if and only if $\vdash \alpha$.

The set of all partial entrenchment rankings is denoted $\mathcal{B}$. $\mathbf{B}(\alpha)$ is referred as the *degree of acceptance* of α. The explicit beliefs of $\mathbf{B} \in \mathcal{B}$ is $\{\alpha \in \mathrm{dom}(\mathbf{B}) : \mathbf{B}(\alpha) > 0\}$, and is denoted $exp(\mathbf{B})$. Similarly, the implicit beliefs represented by $\mathbf{B} \in \mathcal{B}$ is $Cn(exp(\mathbf{B}))$, and is denoted $content(\mathbf{B})$. Cn is the classical consequence operation. The degree of acceptance of implicit belief α is defined as [14]: $degree(\mathbf{B}, \alpha)$ = largest j such that $\{\beta \in exp(\mathbf{B}) : \mathbf{B}(\beta) \geq j\} \vdash \alpha$ if $\alpha \in content(\mathbf{B})$; otherwise $degree(\mathbf{B}, \alpha) = 0$. Let $\mathbf{B} \in \mathcal{B}$ be finite. The range of **B** is enumerated in ascending order as $j_0, j_1, j_2, \ldots, j_{\mathcal{O}}$. Let α be a contingent sentence, $j_m = degree(\mathbf{B}, \alpha)$ and $0 \leq i < \mathcal{O}$. Then the (α, i) maxi-adjustment of **B** is $\mathbf{B}^{\star}(\alpha, i)$ defined by [14]:

$$\mathbf{B}^{\star}(\alpha, i) = \begin{cases} (\mathbf{B}^{-}(\alpha, i)) & \text{if } i \leq j_m \\ (\mathbf{B}^{-}(\neg\alpha, 0))^{+}(\alpha, i) & \text{otherwise} \end{cases}$$

where for all $\beta \in dom(\mathbf{B})$, $\mathbf{B}^{-}(\alpha, i)$ is defined as follows:

1. For β with $\mathbf{B}(\beta) > j_m$, $\mathbf{B}^{-}(\alpha, i)(\beta) = \mathbf{B}(\beta)$.

2. For β with $i < \mathbf{B}(\beta) \leq j_m$, suppose $\mathbf{B}^{-}(\alpha, i)(\beta)$ for β is defined with $\mathbf{B}(\beta) \geq j_{m-k}$ for $k = -1, 0, 1, 2, \ldots, n-1$, then for β with $\mathbf{B}(\beta) = j_{m-n}$,

$$\mathbf{B}^{-}(\alpha, i)(\beta) = \begin{cases} i & \text{if } \alpha \vdash \beta \text{ or} \\ & \alpha \not\vdash \beta \text{ and } \beta \in \Gamma \\ & \text{where } \Gamma \text{ is a minimal subset of} \\ & \{\gamma : \mathbf{B}(\gamma) = j_{m-n}\} \text{ such that} \\ & \{\gamma : \mathbf{B}^{-}(\alpha, i)(\gamma) > j_{m-n}\} \cup \Gamma \vdash \alpha \\ \mathbf{B}(\beta) & \text{otherwise} \end{cases}$$

3. For β with $\mathbf{B}(\beta) \leq i$, $\mathbf{B}^{-}(\alpha, i)(\beta) = \mathbf{B}(\beta)$.

For all $\beta \in dom(\mathbf{B}) \cup \{\alpha\}$, $\mathbf{B}^{+}(\alpha, i)$ is defined as follows:

$$\mathbf{B}^{+}(\alpha, i)(\beta) = \begin{cases} \mathbf{B}(\beta) & \text{if } \mathbf{B}(\beta) > i \\ i & \text{if } \alpha \equiv \beta \text{ or} \\ & \mathbf{B}(\beta) \leq i < degree(\mathbf{B}, \alpha \to \beta) \\ degree(\mathbf{B}, \alpha \to \beta) & \text{otherwise} \end{cases}$$

4 Knowledge Representation

A Web page is pre-processed based on traditional IR techniques [12]. For instance, a Web document is first converted to plain text. The text is indexed resulting in a set of tokens. Stop words (e.g. "the", "a", etc.) are removed, the remaining tokens are stemmed. The stemmed tokens are then weighted according to the tf-idf measure [12], and those above a fixed threshold are deemed to be sufficiently descriptive to represent the text. At the symbolic level, each selected token k is mapped to the ground term of the *positive keyword* predicate pkw e.g. $pkw(k)$. Each atomic formula $pkw(k)$ is in fact a *proposition* since its interpretation is either true or false. For example, if $\{businesses, commerce, trading, \ldots\}$ are the top n extracted tokens (e.g. keywords) from a document, the corresponding symbolic representation becomes: $\{pkw(business)$, $pkw(commerce)$, $pkw(trad)$, $\ldots\}$.

Kindo's keyword classifier [9] is modified to induce a user's information need. In a relevance feedback environment, a user's information needs can be induced from a set of relevant documents D^+ and a set of non-relevant documents D^- judged by the user [4,11]. The basic idea is that a keyword (i.e. token) appearing more frequently in D^+ is probably an indicator of the user's information preference. On the other hand, a keyword appears frequently in D^- may be an indicator of what the user does not want to retrieve. Therefore, the notions of *positive*, *neutral*, and *negative* keywords are proposed [9]. Intuitively, positive keywords represent the information items that a user requires, whereas negative keywords represent the information items that the user does not want. Neutral keywords are not useful for determining a user's information need. Accordingly, a positive keyword k is represented as a formula $pkw(k)$, whereas a negative keyword is represented as $\neg pkw(k)$. Eq.(1) is used to classify the keywords from D^+ and D^-, and to induce the *preference value* $pre(k)$ for each keyword k. Whenever a Web page is judged by the user, the preference values of the set of keywords representing that page can be computed. The corresponding beliefs will be updated in the agent's memory via the belief revision mechanism.

$$\begin{aligned} pre(k) = \epsilon \times \tanh\left(\frac{df(k)}{\xi}\right) \times \\ \left(p(k_{rel}) \tanh\frac{p(k_{rel})}{p_{rel}} - (1 - p(k_{rel})) \tanh\frac{(1-p(k_{rel}))}{(1-p_{rel})}\right) \end{aligned} \tag{1}$$

where ϵ is used to restrict the range of $pre(k)$ in the interval $-1 < pre(k) < 1$. The examples illustrated in this paper assume that $\epsilon = 0.9$. $df(k)$ is the sum of the number of relevant documents $df(k_{rel})$ and the number of non-relevant documents $df(k_{nrel})$ that contain the keyword k, and tanh is the hyperbolic tangent. The rarity parameter ξ is used to control rare or new keywords and is expressed as $\text{int}(logN{+}1)$, where N is the total number of Web documents judged by a user, and int is an integer function that truncates the decimal values. $p(k_{rel})$ is the estimated probability that a document containing keyword k is relevant and is expressed as the fraction $\frac{df(k_{rel})}{df(k_{rel})+df(k_{nrel})}$. p_{rel} is the estimated probability that a document is relevant. In our system, it is assumed that the probability

that a Web document presented by the filtering agent and judged as relevant by a user is $p_{rel} = 0.5$. A positive value of $pre(k)$ implies that the associated keyword is positive, whereas a negative value of $pre(k)$ indicates a negative keyword. If $pre(k)$ is below a threshold value λ, the associated keyword is considered neutral. It is assumed that $\lambda = 0.5$ for the examples demonstrated in this paper. Basically a positive keyword k is mapped to $pkw(k)$, and a negative keyword k is mapped to $\neg pkw(k)$. There is no need to create the symbolic representations for neutral keywords. For $pkw(k)$ or $\neg pkw(k)$, the degree of acceptance $\mathbf{B}(\alpha_k)$ of the corresponding formula α_k is defined as:

$$\mathbf{B}(\alpha_k) = \begin{cases} |pre(k)| \text{ if } |pre(k)| \geq \lambda \\ 0 \qquad\quad \text{otherwise} \end{cases} \tag{2}$$

Table 1 shows the results of applying Eq.(1) and Eq.(2) to induce the beliefs and the associated degree of acceptance based on D^+ and D^-. It is assumed that there are a set of five documents having been judged as relevant (i.e. $|D^+| = 5$) and another set of five documents having been judged as non-relevant by a user (i.e. $|D^-| = 5$). Each document is characterised by a set of keywords e.g. $d_1 = \{business, commerce, system\}, d_2 = \{art, sculpture\}$.

Table 1. Inducing a user's information preferences

Keywords	D^+	D^-	$pre(k)$	Formula:α_k	$\mathbf{B}(\alpha_k)$
business	5	0	0.856	$pkw(business)$	0.856
commerce	4	0	0.836	$pkw(commerce)$	0.836
system	2	2	0	-	-
art	0	5	-0.856	$\neg pkw(art)$	0.856
sculpture	0	3	-0.785	$\neg pkw(sculpture)$	0.785
insurance	1	0	0.401	-	-

A user's state of knowledge includes their understanding about a particular application domain. For example, in the domain of databases, classification knowledge such as (O-O databases $\rightarrow$ databases) can be used by a searcher. Linguistic knowledge such as (VDU $\equiv$ monitor) may also be used in a search. In our current framework, background knowledge is assumed to be elicited from a particular user or to be transferred from a thesaurus of a particular domain. A manual procedure is involved to encode these knowledge in the form of first-order formulae. The following is an example of background knowledge used for the discussion in this paper. In the context of IR, these rules can be interpreted as: an information item about "business" implies that it is about "commerce" and vice versa; "sculpture" is a kind of Arts.

$$pkw(business) \equiv pkw(commerce), 1.000$$
$$pkw(sculpture) \rightarrow pkw(art), 1.000$$

5 Learning and Adaptation

Whenever a user provides relevance feedback for a presented Web document, the belief revision process is invoked to learn the user's information preferences. Conceptually, the filtering agent's learning and adaptation mechanism is characterised by the belief *revision* and *contraction* processes. For example, if $\Gamma = \{a_1, a_2, \ldots, a_n\}$ is a set of beliefs representing a Web document $d \in D^+$, the belief revision process $(((K^*_{a_1})^*_{a_2})\ldots)^*_{a_n}$ is invoked for each $a_i \in \Gamma$, where K is the belief set stored in the filtering agent's memory. On the other hand, the belief contraction process $(((K^-_{a_1})^-_{a_2})\ldots)^-_{a_n}$ is applied for each $a_i \in \Gamma$ if $d \in D^-$. At the computational level, belief revision is actually taken as the adjustment of entrenchment ranking $\mathbf{B}$ in the *theory base* $exp(\mathbf{B})$. Particularly, *maxi-adjustment*[14,13] is employed by our system to modify the ranking in an absolute minimal way under maximal information inertia. As the input to the maxi-adjustment algorithm includes an ordinal i representing the new degree of acceptance of a belief α, the procedure described in Section 4 is used to obtain the new degree for each $\alpha \in \Gamma$. Moreover, for our implementation of the maxi-adjustment algorithm, the maximal ordinal $\mathcal{O}$ is chosen as 1. Therefore, any formula assigned the entrenchment value 1 will not be removed from the theory base $exp(\mathbf{B})$.

Example 1:

The first example shows how adding one belief to the agent's memory will automatically raise the entrenchment rank of another related belief. It is assumed that the belief $\mathbf{B}(\neg pkw(sculpture)) = 0.785$ and the belief $\mathbf{B}(pkw(business)) = 0.856$ have been learnt by the filtering agent. If several Web documents characterised by the keyword *art* are also judged as non-relevant by the user later on, the preference value of the keyword *art* can be induced according to Eq.(1). Assuming that $pre(art) = -0.856$, the corresponding entrenchment rank can be computed as $\mathbf{B}(\neg pkw(art)) = 0.856$ according to Eq.(2). By applying $\mathbf{B}^\star(\neg pkw(art), 0.856)$ to the theory base $exp(\mathbf{B})$, the before and after images of the agent's *explicit beliefs* (i.e. $exp(\mathbf{B})$) can be tabulated in Table 2. Based on the maxi-adjustment algorithm, $\mathbf{B}^+(\alpha, i)(\beta) = i$ if $\mathbf{B}(\beta) \leq i < degree(\mathbf{B}, \alpha \rightarrow \beta)$.

$$\begin{aligned} &\because \mathbf{B}(\neg pkw(sculpture)) \leq 0.856 < \\ &\quad degree(\mathbf{B}, \neg pkw(art) \rightarrow \neg pkw(sculpture)) \\ &\therefore \mathbf{B}^+(\neg pkw(art), 0.856)(\neg pkw(sculpture)) = 0.856 \end{aligned}$$

The implicit belief $\neg pkw(art) \rightarrow \neg pkw(sculpture)$ in $content(\mathbf{B})$ is derived from the explicit belief $pkw(sculpture) \rightarrow pkw(art)$ in the theory base $exp(\mathbf{B})$, and its degree of acceptance is 1 according to the definition of *degree*. As the belief $\neg pkw(art)$ implies the belief $\neg pkw(sculpture)$ and the agent believes in $\neg pkw(art)$, the belief $\neg pkw(sculpture)$ should be at least as entrenched as the belief $\neg pkw(art)$ according to (PER1). In other words, whenever the agent believes that the user is not interested in *art* (i.e. $\neg pkw(art)$), it must be prepared to accept that the user is also not interested in *sculpture* at least to the degree of the former. The proposed learning and adaptation framework is more effective than other learning approaches that can not take into account the semantic

relationships among information items. This example demonstrates the automatic revision of the agent's beliefs about related keywords given the relevance feedback for a particular keyword. Therefore, less users' relevance feedback is required during reinforcement learning. Consequently, *learning autonomy* of the filtering agent can be enhanced.

Table 2. Raising related beliefs

Formula:α	$\mathbf{B}(\alpha)$ Before	$\mathbf{B}(\alpha)$ After
$pkw(business) \equiv pkw(commerce)$	1.000	1.000
$pkw(sculpture) \rightarrow pkw(art)$	1.000	1.000
$pkw(business)$	0.856	0.856
$\neg pkw(sculpture)$	0.785	**0.856**
$\neg pkw(art)$	0	**0.856**

Example 2:

The second example illustrates the belief contraction process. In particular, how the contraction of one belief will automatically remove another related belief from the agent's memory if there is a semantic relationship between the underlying keywords. Assuming that more Web documents characterised by the keyword *sculpture* are judged as relevant by the user at a later stage, the belief $\mathbf{B}(pkw(sculpture)) = 0.785$ could be induced. As $\mathbf{B}^{\star}(\alpha, i) = (\mathbf{B}^{-}(\neg\alpha, 0))^{+}(\alpha, i)$ if $i > j_m$, where $i = 0.785$ and $j_m = 0$ in this example, $\mathbf{B}^{\star}(\ pkw(sculpture), 0.785)$ leads to the contraction of the belief $\neg pkw(sculpture)$ from the theory base $exp(\mathbf{B})$. Moreover, $\mathbf{B}^{-}(\ \neg\ pkw(sculpture), 0)\ (\neg\ pkw(art)) = 0$ is computed because $\mathbf{B}(\ \neg\ pkw(\ art)) = 0.856 = j_{m-n}$ and the set $\{\gamma : \mathbf{B}^{-}(\ \neg\ pkw(sculpture), 0)\ (\gamma) > 0.856\ \} \cup \{\ \neg\ pkw(art)\ \} \vdash \neg\ pkw(sculpture)$ is obtained. The before and after images of the filtering agent's explicit beliefs are tabulated in Table 3.

Table 3. Contracting related beliefs

Formula:α	$\mathbf{B}(\alpha)$ Before	$\mathbf{B}(\alpha)$ After
$pkw(business) \equiv pkw(commerce)$	1.000	1.000
$pkw(sculpture) \rightarrow pkw(art)$	1.000	1.000
$pkw(business)$	0.856	0.856
$pkw(sculpture)$	0	**0.785**
$\neg pkw(sculpture)$	0.856	**0**
$\neg pkw(art)$	0.856	**0**

The explanation of the above learning processing is that given a user's new information preference about *sculpture*, the previous requirement of *not sculpture* should be removed from the agent's memory since it is impossible for the user to require Web documents about *sculpture*, and not require Web documents about *sculpture* at the same time. Moreover, as *sculpture* implies *art*, a user who rejects Web documents about *sculpture* may also reject documents about *art*.

The belief revision mechanism exactly carries out this kind of reasoning to infer the possible changes of a user's information needs. As can be seen, the filtering agent's behavior can be explained based on logical deduction and the explicit semantic relationships among information items. The process is more transparent than examining the weight changes in a keyword vector. In summary, the agent's memory consists of the following beliefs before the learning process:

$$pkw(business) \equiv pkw(commerce), 1.000$$
$$pkw(sculpture) \rightarrow pkw(art), 1.000$$
$$pkw(business), 0.856$$
$$\neg pkw(sculpture), 0.856$$
$$\neg pkw(art), 0.856$$

However, after the belief revision process, the filtering agent's memory becomes:

$$pkw(business) \equiv pkw(commerce), 1.000$$
$$pkw(sculpture) \rightarrow pkw(art), 1.000$$
$$pkw(business), 0.856$$
$$pkw(sculpture), 0.785$$

6 Filtering Web Documents

In our current framework, the matching function of the filtering agent is modelled as logical deduction. Moreover, a Web document is taken as the conjunction of a set of formulae [5,8]. The following example illustrates the agent's deduction process with reference to previous examples. The resulting belief sets $K = Cn(exp(\mathbf{B}))$ from examples 1 and 2 in the previous section are used to determine the relevance of the following three Web documents:

$$\phi = \{pkw(business) \wedge pkw(art)\}$$
$$\varphi = \{pkw(sculpture) \wedge pkw(art)\}$$
$$\psi = \{pkw(business) \wedge pkw(commerce)\}$$

The filtering agent's conclusions about the relevance of the Web documents are summarised as follows:

Time: (t1)

$$K = Cn(\{pkw(business) \equiv pkw(commerce),$$
$$pkw(sculpture) \rightarrow pkw(art),$$
$$pkw(business), \neg pkw(sculpture), \neg pkw(art)\})$$
$$\therefore \quad K \not\vdash \phi, \quad K \not\vdash \varphi, \quad K \vdash \psi$$

Time: (t2)

$$K = Cn(\{pkw(business) \equiv pkw(commerce),$$
$$pkw(sculpture) \rightarrow pkw(art),$$
$$pkw(business), pkw(sculpture)\})$$
$$\therefore \quad K \vdash \phi, \quad K \vdash \varphi, \quad K \vdash \psi$$

At time ($t1$), the filtering agent believes that the user does not want Web documents about *art*. Therefore, documents ϕ and φ are rejected. Moreover, since *business* is equivalent to *commerce* for this particular information searcher, and she is interested in *business*, a Web document ψ about *business* and *commerce* should also be relevant to her. At time ($t2$), after the belief revision process, the agent learns that the user is interested in *sculpture*. As *sculpture* is a kind of *art*, the agent deduces that she may want Web documents about *art* as well. Therefore, documents ϕ and φ are considered as relevant at this time. So, a logic-based framework for representation and reasoning facilitates the explanation of an agent's decisions for information retrieval since the decisions can be explained by logical deduction.

7 Conclusions and Future Work

The expressive power of logic enables domain knowledge properly captured and reasoned about in information agents. Accordingly, a symbolic framework for modelling the agents' learning and adaptation processes is desirable. The AGM belief revision paradigm offers a sound and robust formalism to develop the learning components of adaptive information filtering agents. Since semantic relationships among information items are taken into account during the agent's learning and adaptation process, changes about a user's information requirements can automatically be deduced. Therefore, the proposed symbolic learning and adaptation framework as presented in this account seems to promote higher learning autonomy than other reinforcement learning approaches that cannot, or do not, take semantic relationships into consideration. Moreover, the explicit representation of these semantic relationships also facilitate the explanation of the agents' learning and filtering behavior. However, quantitative evaluation of the effectiveness and the efficiency of these filtering agents are necessary. Future work involves simplifying the maxi-adjustment algorithm to gain higher computational efficiency.

Acknowledgments. The work reported in this paper has been funded in part by the Cooperative Research Centres Program through the Department of the Prime Minister and Cabinet of Australia.

References

1. C.E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.
2. N. Belkin and W. Croft. Information Filtering and Information Retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.
3. D. Billsus and M.J. Pazzani. A personal news agent that talks, learns and explains. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 268–275, Seattle, WA, 1999. ACM Press.

4. C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Routing, pages 292–300, 1994.
5. Y. Chiaramella and J. P. Chevallet. About retrieval models and logic. *The Computer Journal*, 35(3):233–242, June 1992.
6. P. Gärdenfors. *Knowledge in flux: modeling the dynamics of epistemic states*. The MIT Press, Cambridge, Massachusetts, 1988.
7. P. Gärdenfors and D. Makinson. Revisions of knowledge systems using epistemic entrenchment. In Moshe Y. Vardi, editor, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–95. Morgan Kaufmann Inc., 1988.
8. A. Hunter. Using default logic in information retrieval. In Christine Froidevaux and Jürg Kohlas, editors, *Proceedings of the ECSQARU European Conference on Symbolic and Quantitive Approaches to Reasoning and Uncertainty*, volume 946 of *Lecture Notes in Artificial Intelligence*, pages 235–242, Berlin, Germany, July 1995. Springer Verlag.
9. T. Kindo, H. Yoshida, T. Morimoto, and T. Watanabe. Adaptive personal information filtering system that organizes personal profiles automatically. In Martha E. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 716–721. Morgan Kaufmann publishers Inc., August 23–29, 1997.
10. A. Moukas and P. Maes. Amalthaea: An evolving information filtering and discovery system for the WWW. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):59–88, 1998.
11. G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of American Society for Information Science*, 41(4):288–297, 1990.
12. G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
13. M.-A. Williams. Applications of belief revision. In B. Freitag, H. Decker, M. Kifer, and A. Voronkov, editors, *Transactions and Change in Logic Databases*, volume 1472 of *Lecture Notes in Computer Science*, pages 287–316, Schloss Dagstuhl, Germany, September 23–27, 1996. Springer.
14. M.-A. Williams. Towards a practical approach to belief revision: Reason-based change. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *KR'96: Principles of Knowledge Representation and Reasoning*, pages 412–420. Morgan Kaufmann Publishers Inc., 1996.
15. M.-A. Williams. Anytime belief revision. In Martha E. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 74–79. Morgan Kaufmann Publishers Inc., August 23–29, 1997.

The User Agent: An Approach for Service and Profile Management in Wireless Access Systems

Carsten Pils[1] and Jens Hartmann[2]

[1] Informatik 4 (Communication Systems)
RWTH Aachen, 52056 Aachen, Germany
{pils@i4.informatik.rwth-aachen.de}
[2] Ericsson Eurolab Deutschland GmbH
Ericsson Allee 1, 52134 Herzogenrath
jens.hartmann@ericsson.com

Abstract. With their special ability to operate disconnected and autonomously mobile agents are well suited for wireless access networks: Mobile user can dispatch mobile agents to the fixed network where they operate autonomously in the user's behalf. While the agent operates in the fixed network the user's mobile device disconnects. Reconnection is only required when the agent needs user feedback or returns results. Thus, mobile agents have the ability to save wireless network resources. While agents operate disconnected they require a certain knowledge of the user's preferences for service trading. But user preferences might change frequently and thus agents require a central information storage which provides up to date user preference descriptions. In this paper we propose the User Agent which manages user profile entries and acts as the user's central service trader. Agents can request services at the User Agent which correspond to the user's preferences.

Keywords: Mobile agents, service trading, agent management, disconnected operations, personal mobility.

1 Introduction

Although 3rd generation communication systems like UMTS will offer mobile users network access at considerable quality of service, radio resources are still scarce and suffer from interferences. Thus, applications programs specially designed for mobile users should be optimised for saving network resources and must tolerate frequent disconnections. With their special ability to operate disconnected and autonomously mobile agents are well suited for wireless access networks [CHK94]: An agent based application running on a mobile device can dispatch a mobile agent to the fixed network. While the agent autonomously operates in the fixed network the device disconnects. Reconnection is only required when the agent needs user feedback or returns results.

Thus, mobile agents have the ability to save networks resources and tolerate frequent disconnections. In recent years, many research projects have investigated mobile agent usage for wireless access networks [GKN+96], [Col],

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 279–288, 2001.

[KRL+99], [Mit98], [JAE], [BCS00]. The objective of these projects is the development of mobile agent management approaches to support terminal and personal mobility.

Most management facilities offer agents a certain degree of freedom with respect to quality of service. Agents can request different quality of services degrees by submitting parameters or use different services. To discover services there is a need for a service trader. In [PES98] Park et al propose a Service Center which allows agents to retrieve service references from a distributed database. If required the Service Center returns a list of appropriate services. Thus, agents can choose a service based on user preferences out of a multitude of alternatives to complete their tasks.

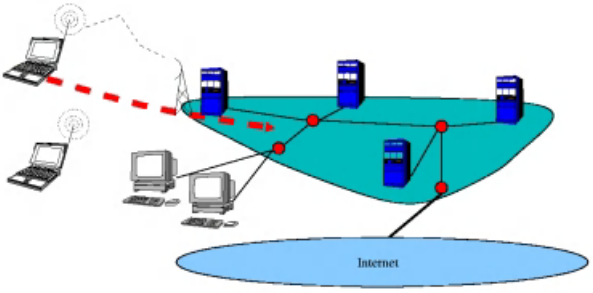


Fig. 1. Wireless Access Network

As user preferences and registered services might frequently change, fixed service lists are not useful for agents. To solve this problem agents can carry service descriptions and trade services right before service usage. But the latter solution still has some disadvantages: First, the agent size is increased by service descriptions. Second, processing of agents is delayed due to the service discovery and selection process. Third, once the agent is launched the user cannot modify the service descriptions.

To solve this problem, we propose the User Agent which supports mobile agent service trading and management. Basically, the User Agent acts as an active extension of the user profile. It is envisaged to implement the User Agent in JAE (Java Agent Environment) [JAE].

The remainder of this paper is organised as follows: Next we briefly describe the JAE agent technology in section 2 and the User Agent in section 3. Finally, we address performance considerations in in section 4 and security issues in section 5.

2 The Java Agent Environment

Each JAE agent system provides the core technology for mobility, security and services. An integral part of an agent system is the Service Center. The Service Center takes care of the administration of local services and provides a trading mechanism for remote services. Remote service trading requires a central repository for all available services. To this end, the current release of JAE uses the LDAP directory service, as it has the advantage of providing a distributed, standardised, and publicly accessible database.

2.1 The JAE Service Center

The JAE technology distinguishes between mobile and fixed service agents. While mobile agents can migrate to different agent systems by means of the *Agent Transport Facility*, fixed service agents stay at the agent system where they have been launched. Since service agents cannot migrate they are considered trustworthy and are allowed to access system resources. Mobile agents must always interact with a service agent when accessing system resources. Apart from providing a controlled access to system resources, service agents can also provide application services. The mobile agents must somehow locate the available service agents. Therefore, agents issue a service description of the desired service to the Service Center (see figure 2). The Service Center either mediates the request to a suitable local service agent or provides information about another agent system that offers the requested service.

Non local services are looked up in the LDAP directory tree. Among others the LDAP directory has a service and a user profile subtree. Below the service subtree service descriptions as well as service reference information are stored [PES98].

2.2 The User Profile

User profiles are stored below the user profile subtree. They can be accessed via service agents and contain for instance the user's private as well as business email address, his phone and fax number, and the system he is currently logged in to. The user's current agent system (i.e. the agent system he is currently logged in to) is used to provide personal mobility.

Personal mobility allows a user to access communication services through different devices, e.g., at home, office, mobile phone, notebook, PDA. This means that the user has several different access points. Agents which need user feedback information or return results must know the user's current agent system. By applying the user profile, agents can retrieve the current agent system of a user: When the user logs in at an agent system, the latter writes its address to his user profile. Now if an agent wants to get in contact with the user the agent can look up the user profile and retrieve the current agent system's address.

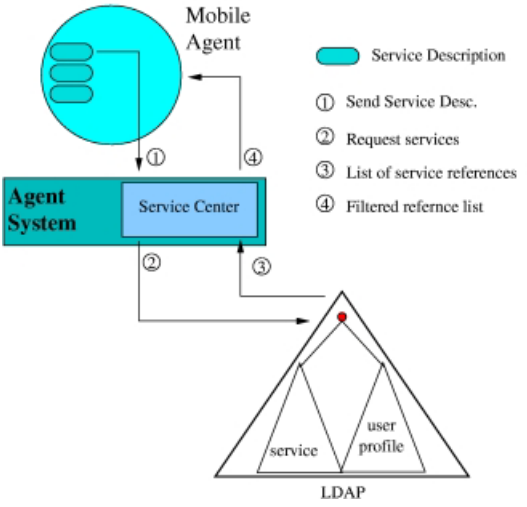


Fig. 2. Service Trading

3 The User Agent

The User Agent is an active extension of the user profile. Basically, its envisaged application is to act as the user's central service trader. Thus, mobile agents can request services from the User Agent which fit the user's preferences.

The User Agent comprises of a static and a dynamic database. The static database contains the user profile, his preferences, and service descriptions. Using the stored service descriptions and the user's preferences the User Agent looks up services with the help of the Services Center. Those services which match both service descriptions and user preferences are stored in the dynamic database. Once the dynamic database is generated, mobile agents can request a service from the User Agent. Therefore, it Agent just retrieves a suitable service from its dynamic database and sends the reference to the mobile agent.

3.1 Service Trading

As mentioned before the User Agent retrieves service reference from the Service Center using service descriptions and preferences stored in the static database.

Service Descriptions. JAE service descriptions contain all information about a service agent. This information is stored in attribute-value pairs. There are four attribute-value pairs specifying a service:

1. *ServiceType*: Description of the service category.
2. *Action*: Functions offered by this service.

3. *Parameters*: Parameters accepted or needed by this service.
4. *Return Values*: Type values returned by the service functions.

Since there must be common base values for these attributes, JAE introduces the so-called *Met-Keywords* that agent programmers use to specify services. Service agents use these key words to describe the service they offer, while mobile agents need them to specify the service they want to use. For example, a service description object of a bank could look as follows:

– *ServiceType* = BANK.MY_BANK
– List of ACTION descriptions:
  · *Action*= CHECK_BALANCE
  · *Parameters*= INTEGER.ACCOUNT_NUMBER
  · *Result Value*=OBJECT.STATEMENT
  · *Action* = GET_STOCK_INDEX
  · *Parameters* = STRING.STOCK_MARKET_ID
  · *Result Value* = INTEGER.INDEX

A service agent of "MY_BANK" could register its reference with this service description object at the Service Center. An agent which wants to use a banking service to get the index of a particular stock market could issue the following service description to the Service Center:

– *ServiceType* = BANK
– List of ACTION descriptions:
  · *Action* = GET_STOCK_INDEX
  · *Parameters* = STRING.STOCK_MARKET_ID
  · *Result Value* = INTEGER.INDEX

Among others the Service Center will return a reference of the BANK.MY_BANK service. Note that the Service Center filters service descriptions with respect to the *ServiceType*. Thus, if an agent issues the same service description to the Service Center, but *ServiceType* set to BANK.MY_BANK service, the Service Center would only return the BANK.MY_BANK service [PES98].

Preference Definitions. Along with the service description, a service agent can store quality of service information in the service directory. This information is stored in attribute value pairs. While the attribute is just a string the value is an object allowing a complex quality of service description. For example, the BANK.MY_BANK service could describe the costs of a single transaction as follows:
attribute=COST.TRANSACTION, value={ float=0,1; STRING="USD" } — a single transaction amounts to 0,1 US dollar.

It is the User Agent's task to order services with respect to user preferences. To this end the User Agent calculates a preference number for each service.

Numerous attribute-value pairs can be associated with one single service. The user will value each quality of service parameter differently. To ease individual calculation of preference numbers (with respect to individual services) we propose the use of preference objects. Each preference object is associated with a service and has two methods: *STRING getKey()* and *INTEGER getPreference(service)*. *getKey()* returns the name of the associated service and *getPreference()* returns the preference number. To calculate a service preference number the *getPreference()* method requires its quality of service descriptions (retrieved from the Service Center) as input parameter. Quality of service information is stored in the service reference object.

Preference objects are created by the user and submitted to the User Agent. Each service entry in the static database is identified by a service key. The entry comprises of service key, service description, and preference object.

3.2 Manager

Basically, the Manager controls the User Agent. Among others it allows modification of the static database and generating of the dynamic one.

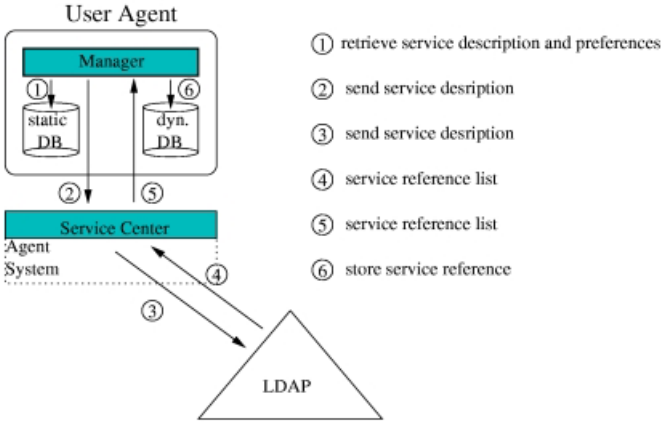


Fig. 3. Generating dynamic entries

Generating dynamic entries. Generating of dynamic database entries is depicted in figure 3. The Manager fetches an entry from the static database and issues the service description to the Service Center. The latter responds with a list of service references. Next, the Manager applies the preference object to the reference list and sorts it in order to the calculated preference numbers.

Finally, the Manager stores the best service reference in the dynamic database. Again the entry is identified by the service key.

An agent which requests a service reference from the User Agent must only provide the service key. Having the key it is straightforward to retrieve a service reference from the dynamic database. Note that it is required that agent applications and the User Agent must agree about a service key.

A dynamic entry is always updated when the user modifies the preference object. As services might change their quality of service parameters, it is required that dynamic entries are updated at regular intervals. To this end the user specifies an update interval for each entry. Furthermore, a service entry is updated if an agent reports that the recommended service is not available.

User profile management. Yet, the User Agent showed to be the user's central service trader. But being an active extension of the user profile its ability goes far beyond mere service trading. We motivate additional features of the User Agent with the help of the following example:

A user runs an agent based application in his office. During his free time he might be committed to a honorary position and runs a special agent application. Thus, the user might want to run three current agent systems: one in his office, one at home related to the honorary position, and one "private" agent system. For each of these agent systems the user is expected to have a different user profile. Thus, the user is registered to the agent world three times. But registering three user profiles has several disadvantages: First, each user registration consumes system resources. Second, the user must access several user profiles if he wants to modify a common entry. Third, removing or adding a user profile is quite complicate.

To ease managing multiple user profiles we propose that the User Agent allows maintaining more than one user profile. It distinguishes different profiles by adding a suffix to the user's login name. With respect to the given example a user could have the following profiles: *user:business*, *user:honorary_position*, or *user:private*. Our approach has several advantages:

- System resources are not wasted since the user's different profiles share information.
- The user must access only one User Agent to modify his user profile.
- Removing and adding a user profile is simple since it must not be registered by the network provider.
- Since the User Agent can access the user profiles, it can forward information from one profile to another. For example, a user does not want that agents related to his business profile send messages to his private profile. But if an agent has some important business information then the agent should send the information to the user's current agent system. As each agent must authenticate to the User Agent, the latter knows the agent's owner identification. Based on agent and agent owner identification the User Agent can decide in accordance to user defined rules which information should be forwarded to the agent. A simple approach for user defined rules are access control lists. But access control lists do not allow description of complex forwarding rules like: "If agent X returns results to my business profile indicating *emergency*

give all business related agents access rights to my private user profile entries: current agent system, email address, and fax". There is a need for a rule based information exchange management.

With its ability to maintain multiple profiles the User Agent eases profile management and allows flexible personal mobility management.

Equipped with service trading and user profile management functionality the User Agent possesses powerful tools to shield the user from agent management tasks.

4 Performance Considerations

Like every central approach, the User Agent has its weaknesses. If the User Agent fails all applications depending on it will fail too. Furthermore, due to its importance network traffic is increased because all mobile agents of a user are expected to interact with the user's User Agent. It seems that the weaknesses surpass the strengths of our approach, but there is way to mend these problems.

4.1 On Mending the Flaws of a Centralised Approach

The User Agents stores those static and dynamic database entries in the LDAP directory (user profile subtree) which are of interest to agent applications. Thus, agent's can retrieve information from both the LDAP directory or the User Agent. In that way the User Agent utilises the replication mechanism of the directory service to distribute information. If the User Agent fails agent applications can still retrieve required information from the LDAP directory. Furthermore, most agent's are expected to interact with the directory and thus traffic is kept local (use replica server).

Still there are agents which must interact with the User Agent (e.g. the Waiting Support Service, user profile modifications). Most of these agents are management services and located close to the user's access point. Implementing the User Agent as a mobile agent which follows the user's movement keeps User Agent interactions local. Although the User Agent might have a huge code and data size, its migration costs are low: Since it is expected that every user has a User Agent its code is well distributed and almost available at each agent system in the fixed network. Thus, it is expected that the agent's code is not transfered to the destination agent system. Furthermore, most entries of the static and dynamic database are stored in the user profile subtree — therefore, static and dynamic database entries are not transfered during agent migration. We conclude that User Agent migration can be performed at considerable costs.

4.2 On Analysing the Central Trader

Apart from network traffic caused by User Agent interactions we must also consider the traffic caused by dynamic database updates.

Basically, the User Agent should trade those service which are frequently used by mobile agents. If the average number of service entry updates is larger than the number of requests, updating the service entry at regular intervals is useless.

There is a need for a intelligent service entry update strategy as some services like the Waiting Support Service are only requested in certain situations. Obviously, it makes no sense to trade Waiting Support Services at regular intervals while the user is logged-in. At that instant the user disconnects the User Agent should discover a Waiting Support Services.

5 Security Considerations

The User Agent approach requires that the agent contains sensitive user information and therefore the approach requires a strong security concept. Particularly, when the agent roams through the network it is required that the visited agent systems are trustworthy.

Since a user must have a contract with a mobile network provider we assume that the network provider is responsible for taking care that the User Agent visits only trusted agent systems, i.e. those ones which are possessed by the provider or one of his contractual partners. In that sense the User Agent code and migration mechanism (if it implements one) is in the possession of the Network Provider.

To protect the User Agent against other agents visited agent system must take care that agents exchange data only via well defined interfaces (but this is just a problem of good software development). It is up to the User Agent that it sends sensitive data just to those authenticated agents which have appropriate access rights.

6 Conclusions

In this paper we proposed the User Agent which is an active extension of the user profile. Among others the User Agent acts as the user's central service trader. Therefore, it maintains a dynamic database which holds service references. The dynamic database is constantly updated according to service descriptions and user preferences. Our approach allows frequent modifications of services descriptions and user preferences. These modifications affect all agents including those which are already dispatched and still running.

It is envisaged to implement the agent in the Java Agent Environment (JAE). JAE provides access to a LDAP directory service. To distribute data, the agent's dynamic database is stored in the directory. Mobile agents either retrieve service references from the agent itself or from the directory.

Apart from service trading the User Agent allows maintaining more than one user profile, e.g. private and business profile. The User Agent eases profile management and controls information exchange between different profiles.

Although, we give reasons that the User Agent scales and performs well further performance analysis is required. To this end we are implementing a simulation model.

References

[BCS00] P. Bellavista, A. Corradi, and C. Stefanelli. A Mobile Agent Infrastructure for Terminal, User, and Resource Mobility. In *IEEE/IFIP Network Operations and Management Symposium*, Honolulu Hawaii, April 2000.

[CHK94] David Chess, Colin Harrison, and Aaron Kershenbaum. Mobile Agents: Are they a good idea? Technical Report RC 19887, IBM, October 1994.

[Col] Dartmouth College. D'Agents. http://agent.cs.dartmouth.edu/.

[GKN+96] Robert S. Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile agents for mobile computing. Technical Report PCS-TR96-285, Dartmouth College, Computer Science, Hanover, USA, May 1996.

[JAE] JAE: Java Agent Environment. http://www-i4.informatik.rwth-aachen.de/jae/.

[KRL+99] Ernoe Kovacs, Klaus Roehrle, Hong-Yong Lach, Bjoern Schiemann, and Carsten Pils. Agent-based mobile access to information services. In *4th ACTS Mobile Communications Summit*, pages 97–102, June 1999.

[Mit98] Mitsubishi Electric ITA, Horizon System Laboratory. *Mobile Agent Computing — A White Paper*, January 1998.

[PES98] Anthony S. Park, Michael Emmerich, and Daniel Swertz. Service trading for mobile agents with ldap as service directory. In *IEEE 7th Intl. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE'98*. IEEE, June 1998.

System Analysis of Agent-Based LCC Information Gathering

Tiemei Irene Zhang[1] and Elizabeth Kendall[2]

[1]Computer Systems Engineering, Royal Melbourne Institute of Technology, City Campus, GPO Box 2476V, Melbourne, Vic. 3001 Australia
t.zhang@rmit.edu.au
[2]Sun Microsystems Chair of Network Computing, The School of Network Computing, Monash University, Peninsula Campus, McMahons Rd., Frankston, Vic. 3199 Australia
kendall@infotech.monash.edu.au

Abstract. To respond to the challenge of global economic competition, manufacturers are searching for ways to gain a sustainable market advantage for their products. They have to bring high-quality products into being in response to established needs. Simultaneously, as cost is a key factor among many physical and social factors to determine the success of a product, they attempt to reduce costs during every phase of the product's life cycle. One of obstacles to using LCC (Life Cycle Costing) models is data gathering from a highly distributed heterogeneous environment with a huge number of information sources. This paper presents a system analysis approach for agent-based system development for a product life cycle cost.

1 Introduction

Cost is a key factor to determining the success of a product [14]. Manufacturers need to reduce costs during every phase of a product's life cycle [28]. The true cost to manufacture a product is much more than the sum of purchase prices of materials. This product will carry with it a host of costs associated with its acquisition, use, and ultimate disposal; these include costs of storage, transportation, facilities, energy, labor, training, and record-keeping, among others. Therefore, life cycle costs of a product can be the total costs associated with the product life cycle, including the phases of research and development (R&D), production and construction, and operation and support (O&S) [5].

One of significant barriers to using LCC models [4] [17] is data gathering from organizations to meet requirements of a life cycle costing model. This results from a highly distributed heterogeneous environment with a huge number of information sources. When data-processing systems are distributed in various formats, manufacturers have to search them separately and manually integrate information from flat files, relational databases, and remote supplier parts catalogs. Due to the explosion in the amount of information, it is more complex for collectors to understand customer needs, develop a product to meet these needs, and bring that product to market quickly and at fair value. Existing methods, such as Concurrent

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 289–298, 2001.

Engineering (CE) and Computer Integrated Manufacturing (CIM), have been applied to achieve these goals. However, there is a growing consensus that traditional organizational structures and rigid corporate rules are a significant barrier to CIM [35] and CE [21]. These methods require organizations to facilitate cooperation and eliminate cross-functional barriers between engineering, marketing, manufacturing, accounting and information service departments. Doing so requires a flexible organizational structure. In recent years, a new wave of changes to the business environment has emerged [20]. The exponential growth of the Internet throughout the last decade has led manufacturing companies to move into a globalized business environment. They can interact with business partners and customers around the world over the Internet. As information from the Internet is diverse, this barrier becomes outstanding.

In this paper we describe an analysis approach for developing an agent-based system to predict product life cycle costs using data gathered from organizations. This approach is outlined in sections 2 and 4. Section 3 provides a layered conceptual model derived from the architecture of InfoSleuth for information gathering [30]. We finally identify general roles of agents used to estimate product life cycle costs.

2 A System Analysis Approach

Since agent technology is getting more popular today, agent-oriented software engineering process needs developing for a multi-agent system. Object-oriented (OO) approach has been widely used in software development and use case analysis has proved to be useful and successful for requirement specification of OO systems. However, we can not directly apply this approach to agent-oriented software engineering because an agent is autonomous, social, reactive and proactive [24], [34], while an object does not possess these characteristics. Current research in role models shows promising results for agent analysis and design [23]. We have combined these two approaches into one to specify and develop an agent-based information gathering system for estimating product life cycle cost. Fig. 1 shows a flow diagram for this approach.

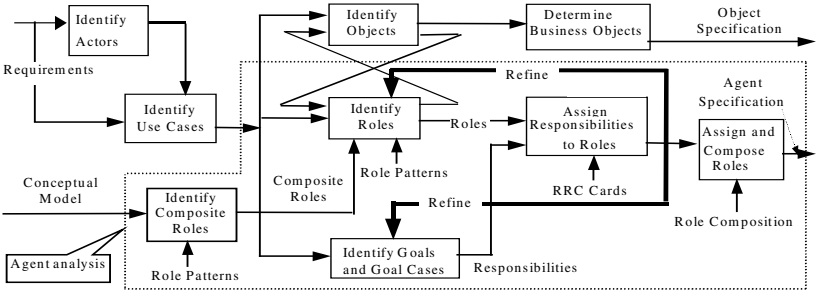


Fig. 1. ICOM diagram for agent-oriented analysis

This figure depicts a process specify and design agents. In this figure, each activity, represented by a solid box, uses several different models, transforming and refining them from activity to activity. The box is given an ICOM (Input, control, output and mechanism) representation adopted from the functional model of IDEF [6], [7]. ICOM and IDEF have been widely used for modeling manufacturing processes. A thick line with an arrow for connecting two activities represents interaction collaboration between these activities. All activities as shown in Fig. 1 can be classified into two categories: agent-oriented analysis (bounded by a dashed polygon) and object-oriented analysis including activities (not bounded by the dashed polygon). Object-oriented analysis consists of four activities: "Identify Actors", "Identify Use Cases", "Identify Objects" and "Determine Business Objects" where traditional methods developed by Jacobson et al [19] are used.

The identified use cases are also fed to the activity "Identify Goals and Goal Cases" in the agent-oriented analysis process. A goal is an objective or a desired state that can be achieved or ascertained by an agent. A goal case is a goal-based use case that is defined as a collection of scenarios about the agent's interaction with a system. Each scenario describes a sequence of events that the agent initiates. The result of the sequence of events has to be something of use to the initiating agent, or to another agent. An agent can start a goal case when the corresponding goal is triggered. Use of goal cases is for traceability because they link to system requirements. We use the exiting method from [26] to identify goals and their goal cases and present them in a hierarchical diagram.

Roles can be identified from the use cases that are output of the activity "Identify Use Case" by using role model catalogs (refer to "Identify Roles"). A role model catalog resembles a set of patterns [16] or a pattern language [3], [15], [25] and it is being used at British Telecom (BT) for downstreaming agent technology [27]. This catalog developed at BT includes approximately 60 role models, including those from agent enhanced workflow, flexible manufacturing, electronic commerce, agent based information management and retrieval, and the FIPA protocols [12] (contract net, auctions, etc.). Role models are patterns of interaction. The patterns in the BT catalog were mined from numerous existing agent frameworks or specifications. This includes Zeus [31], FIPA, and Single Function Agents [13]. Roles and role models are also considered in intelligent systems, such as those found in KADS [18]. Role models that commonly occur can then be documented as role patterns. To document role models of agent systems [22], one important method is to use role, responsibility and collaboration (RRC) cards to specify responsibilities and collaborations of agents. It is essential in the early phase of agent-oriented software development. A template for a RRC card is shown in Table 1.

Table 1. Template to represent RRC card

Role model name		
Role type	**Responsibility**	**Collaborator**
Names of Roles	List all responsibilities	List all collaborators

The following steps are outlined to identify roles:

- Instance composite roles if they are available (described in next sections).
- Examine role patterns from the existing role patterns. The determined role patterns can specify types of interaction and collaboration of the role with other roles.
- Partition goals to form roles if they are not patterns.
- Determine all roles for the identified interactions and collaborations.

To start at the bottom of the hierarchy goal diagram, the activity "Assign Responsibilities to Roles" can determine goal cases that each role can achieve and then assign them to responsibilities of that role. Once responsibilities are assigned to roles in a multi-agent system analysis, we can partition and compose identified roles into different categories for agent design (refer to "Assign and Compose Roles"). This role model composition is the integration of the relevant role models in an application [1]. This composition is more than just the sum of the constituent patterns. It captures the synergy arising from the different roles an agent plays in the overall composition structure and follows some constraints that ensure the proper semantics of the composite are maintained. These constraints are called composition constraints that are the elements of the set of pairwise relationships between the roles that objects can play [32]. To assign and compose roles to agents, we should:

- Design agents in appropriated size i.e. to assign appropriated number of roles to each agent.
- Compose the roles for the agents with differentiation emphasizing the specialization that is goal oriented. Split an agent if it has too many responsibilities for the different sub-goals. Merge agents if they have similar responsibilities.
- Compose the roles for the agents with good quality of collaboration, for which some aspects are essential such as cohesion, lower coupling, and minimum need for communication.

3 Layered Conceptual Model

Before applying the general system analysis approach above, we need to develop a conceptual model to organize our system in a structure with the functionality to be best supported. The Layered Architecture pattern [8] has been widely accepted as a standard in network design and software engineering. This architecture reduces the coupling between the complexity of individual problems. It helps to structure applications that can be decomposed into groups of subtasks, in which each group of subtasks is at a particular level of abstraction. In this paper, information gathering domain is classified into different layers within a structure. Each layer provides a level of abstraction and certain services to the layer above it, while hiding the implementation of its services from the higher layer. Also, each layer passes both data and control information to its corresponding neighbors. It is similar to communication within an organization in real world. Therefore, we propose a layered conceptual model for an agent-based information gathering system as shown in Fig. 2.

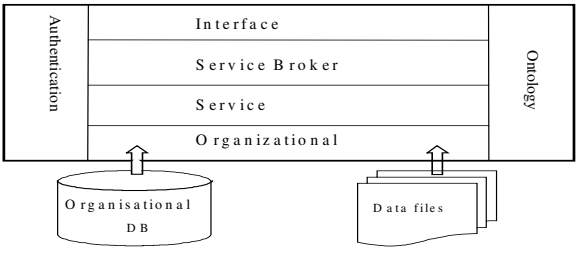


Fig. 2. Conceptual model of information gathering system

In this figure, the ontology layer collectively maintains a knowledge base of the different terminology and concepts that are employed over the whole organization. A given ontology describes language that would be used for specifying requests for information. The individual and unique ontology is used to express the meaning of the requesters, so the ontology layer is required for translation. The ontology layer adapts a request to other modules in different layers. The authentication layer is responsible for the system security.

The interface layer is tied closely to an individual human's goals and it can be used to predict the user's intentions and to request services provided by the remaining modules. This layer acts on behalf of users to relay specifications and obtain results. The service broker layer predicts or models the intentions of the overall organization and then provides services to users via the interface layer. Negotiation may occur between the interface and service broker layers. In order to provide better services to users, the service broker layer should gather information from other layers, then process the information and send the results to user via the interface layer. The service layer is used to provide services. This layer differs from the organizational layer that controls resources. The service layer represents and provides the high level services of the organization. These services can be formed by encoding expertise and by utilizing the organizational layer. The organizational layer can be used to manage the organizational resources. Its main task is to gather data from the various sources. Data can be recorded in different ways and must be adjusted to be consistent and comparable.

4 Relevant Role Models

Based on the layered conceptual model, we identify the relevant role models in this section. Our system requires the ability to accurately identify the user who is making requests. The user's identity is verified by checking a password typed in during login. The process that verifies and records the user's identity is called authentication, and this is carried out in the authentication layer in Fig. 2. The Bodyguard pattern allows objects to be shared and to control access to them in a distributed environment without system-level support for distributed objects [29]. It provides message dispatching validation and assignment of access rights to objects in nonlocal

environments to prevent incorrect access to an object in collaborative applications. The Bodyguard role model is depicted in Fig. 3.

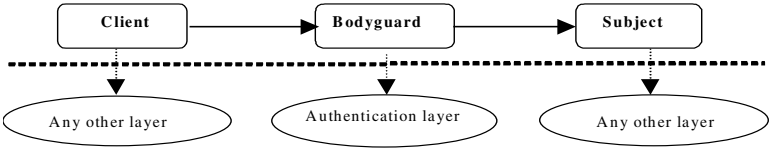


Fig. 3. Bodyguard role model. An arrow between two roles indicates collaboration; the direction of the arrow signifies the messaging path. The bottom half of the figure depicts role assignments in the application. When a dashed line is drawn between a role and a layer, it means that the layer plays the given role.

Three roles are depicted in the top half of the figure: Client, Bodyguard, and Subject. As a result, a layer plays the Client role; the authentication layer plays the Bodyguard role, and the another layer is the Subject. Responsibilities and collaborations for these three roles are shown in Table 2.

Table 2. RRC card for bodyguard role model

Bodyguard		
Role type	**Responsibility**	**Collaborators**
Client	to request the permission of a service	> Bodyguard
Bodyguard	to validate the qualification to notify providing service to provide permission for services	 >Subject >Client
Subject	To provide the service To accept the notification of the service	>Client >Bodyguard

The Adapter pattern aims to convert the interface of a class into another interface that the client expects. We need the Adapter pattern here because there should be a layer to translate and match the meanings between the layer and a user or the other layer. In the Adapter pattern, there are three roles: Client, Adapter/ Wrapper, and Target. In our example, the ontology layer acts as Adapters. All of the other layers may appear in the Client or Target roles. Fig. 4 shows this description and Table 3 shows a RRC card for the Adapter role model.

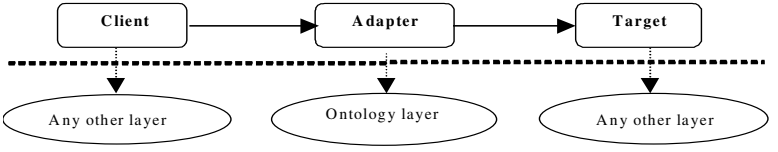


Fig. 4. Adapter role model

Table 3. RRC card for the Adapter role model

Adapter		
Role type	**Responsibility**	**Collaborators**
Client	□ To collaborate with the Target	> (server) Adapter
Adapter	□ Adapt interface expected by the Client to that of the Target	> (server) Target
Target	□ To receive the message sent by the Client □ To perform a task and send a reply	> Adapter/Client

The Observer pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. This pattern can be applied when an abstraction has two aspects, one dependent on the other. Encapsulating these aspects in separate objects lets you vary and reuse them independently. Fig. 5 shows the Observer role model and Table 4 shows a RRC card for this model:

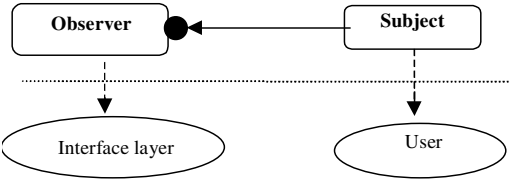


Fig. 5. Observer role model

Table 4. RRC card for Observer role model

Observer		
Role type	**Responsibility**	**Collaborators**
Observer	□ To store information that is consistent with the subject	
Subject	□ Notify observers when its state changes	>Observer

The Manager design pattern encapsulates management of the instances of a class into a separate Manager object [34]. In our system, the organizational layer is responsible for managing the organizational resources. The organizational layer plays the Manager/Owner role, while the actual databases and information sources play the role of Resource. The Broker pattern provides for communication and location transparency for interoperating applications. The service broker layer plays the role of Broker in this Broker role model. The service layer also plays the role of Master in the Master/Slave role model that comes from Master/Slave pattern [2].

5 Role Composition

In terms of the layered conceptual model in Fig. 2, we can characterize composite roles used for identifying agent roles for information gathering. In this characterization, the composite roles with more sophisticated behaviors in the higher levels depend on lower level capabilities, and they only depend on their neighbors. Table 5 shows these composite roles extracted from the relevant role models for the conceptual model.

Table 5. Summary of composite roles

Roles	Types	Description
Interface Role	Observer (Observer), Client-proxy (Broker), Client (Bodyguard), Client & Target (Adapter)	Act on behalf of user; Request to provide services; Obtain result.
Broker Role	Broker (Broker), Client & Subject (Bodyguard), Client & Target (Adapter)	Get user request; Match request to service; Send request to service provider; Reply result.
Service Role	Master (Master/Slave), Server-proxy (Broker), Client & Subject (Bodyguard), Client & Target (Adapter)	Accept the request; Provide service; Distribute work if need; Get final result; Send result.
Organizational Role	Manager (Manager), Slave (Master/Slave), Client & Subject (Bodyguard), Client & Target (Adapter)	Manage information resources; Query information; Reply result.
Authentication Role	Bodyguard (Bodyguard), Client & Target (Adapter)	Verify and validate user; Send out permission
Ontology Role	Adapter (Adapter), Client & Target (Bodyguard)	Get help request; Find alternatives; Translate Terms.

In this table, there are six composite roles identified: Interface Role, Broker Role, Service Role, Organizational Role, Authentication Role, and Ontology Role. These composite roles are used as patterns to identify roles for an agent-based application. For instance, if we assign the service role to an agent who is responsible for estimating life cycle costs, this agent should play the master role that sends requests to organization agents for gathering information. It may also play roles of Client & Target when it requests alternatives for vocabularies that can mark up the gathered information. As a result, we can assign goals and goal cases extracted from uses cases to this agent in terms of responsibilities of Service Role. In a similar manner, we have composed these composite roles to 10 agents [36] that are implemented by using Jack Intelligent Agents [9], [10], [11].

6 Conclusion and Future Work

This paper has proposed a systematic process for developing agent-based system. Based on the layered conceptual model, generic roles have been identified and composed to agents used in LCC information gathering system. This system can be applied to estimating product life cycle costs.

References

1. Andersen, E.: Conceptual Modelling of Objects: a Role Modelling Approach. PhD Thesis, University of Oslo (1997)
2. Aridor, Y., Lange, D.B.: Agent Design Patters: Elements of Agent Application Design. Autonomous Agents (Agents'98), Minneapolis (1998) 108-115
3. Baumer, D., Riehle, D., Siberski, W., Wolf, M.: Role Object. Proceedings of the 4th Annual Conference on the Pattern Languages of Programs, Monticello, Illinois, USA, September 2-5, (1997)
4. Benson, S.: Life Cycle Costs and Dic Pump. http://www.discflo.com/lccart.html(1998)
5. Blanchard, B.S., Fabrycky, W.J.: Systems Engineering and Analysis. Prentice-Hall, Inc., New Jersey, USA (1990)
6. Bravoco, R.R., Yadav, S.B.: Requirements Definition Architecture–An Overview. Computers in Industry 6 (1985) 237-251
7. Bravoco, R.R. and Yadav, S.B.: A Methodology to Model the Functional Structure of an Organisation. Computers in Industry 6 (1985) 345-361
8. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Patter-Oriented Software Architecture: A System of Patterns. Wiley, USA (1996)
9. Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A.: Light-Weight Intelligent Software Agents in Simulation. SimTech 99, Melbourne, Australia (1999)
10. Busetta, P., Ronnquist, R., Hodgson, A., Lucas, A.: JACK Intelligent Agents-Components for Intelligent Agents in Java. Agent Link News 2, January 1999 ISSN 1465-3842 (1999)
11. Cross, M., Ronnquist, R.: A Java Agent Environment for Simulation and Modelling. SimTech 99, Melbourne, Australia (1999)
12. Dickinson, I.: Agent Standards. Agent Technology Group. http://drogo.cselt.stet.it/fipa (1997)
13. Dunskus, B.V, Grecu, D.L., Brown, D.C., Berker, I.: Using Single Function Agents to Investigate Conflicts. In Smith I. (ed.): Conflict Management in Design. AI EDAM, Special Issue. Cambridge UP (1996)
14. Fabrycky, W.J., Blanchard, B.S.: Life-Cycle Cost and Economic Analysis. Prentice-Hall, Inc., New Jersey, USA (1991)
15. Fowler, M.: Dealing with Roles. Pattern Languages Of Programming (PLOP'97), Illinois (1997)
16. Gamma, E.R., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)
17. Hennecke, F.: Life Cycle Costs of Pumps in Chemical Industry. Chemical Engineering and Processing 38 (1999) 511-516
18. Hickman, F. R., Killin, J. L., Land, L., Mulhall, T., Porter, Taylor, R.M.: Analysis for Knowledge-Based Systems: A Practical Guide to the KADS Methodology. Ellis Horwood, Ltd, Chichester (1989)
19. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, J.: Object-Oriented Software Engineering–A Use Case Driven Approach. Addison-Wesley (1992)

20. Jiang, H. C., Mo, J.: Internet Based Design System for Global Concurrent Engineering. Proceedings of the Second International Conference on Managing Enterprises, Newcastle, Australia (1999) 497-501
21. Jin, Y., Levitt, R.E., Christiansen, T.R., Kunz, J.C.: The Virtual Design Team: Modeling Organizational Behavior of Concurrent Design Teams. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 9 (1995) 145-158
22. Kendall, E.A.: Agent Roles and Role Models: New Abstractions for Multi-agent System Analysis and Design. International Workshop on Intelligent Agents in Information and Process Management, Germany, September (1998)
23. Kendall, E.A.: Role Modeling for Agent System Analysis, Design, and Implementation. First International Symposium on Agent Systems and Applications (ASA'99), Third International Symposium on Mobile Agents (MA'99), Palm Springs, October (1999)
24. Kendall, E.A., Malkoun, M.T., Jiang, C.: A Methodology for Developing Agent Based Systems for Enterprise Integration. In: Bernus, P., Nemes, L. (eds.): Modeling and Methodologies for Enterprise Integration. Proceedings of the IFIP TC5 Working Conference on Modeling and Methodologies for Enterprise Integration, Queensland, Australia (1995) 333-344
25. Kendall, E.A., Malkoun, M.T., Jiang, C.: Multi-agent System Design Based on Object Oriented Patterns. The Journal of Object Oriented Programming, June, 1997 (1997) 447-456
26. Kendall, E.A., Palanivelan, U., Kalikivayi, S.: Capturing and Structuring Goals: Analysis Patterns. EuroPlop'98, European Pattern Languages of Programming, Germany, July, (1998)
27. Kendall, E.A., Murali Krishna, P.V., Chirag V., Pathak, C.B.S.: Patterns of Intelligent and Mobile Agents. Agents '98, May, (1998)
28. Li, Y., Huang, B., Wu, C.: Virtual Enterprise Information System. Proceedings of the 1st Asia-Pacific Conference on Intelligent Agent Technology (1999) 493-497
29. Neves, F.D., Garrido, A.: Bodyguard. In R.Marting, D.Riehle, F.Buschmann (eds.): Pattern Languages of Program Design 3. Addison Wesley (1998) 231-243.
30. Nodine, M., Perry, B., Unruh, A.: Experience with the InfoSleuth Agent Architecture. Proceedings of AAAI-98 Workshop on Software Tools for Developing Agents (1998)
31. Nwana, H.S., Ndumu, D.T., Lee, L.C.: ZEUS: An Advanced Tool Kit for Engineering Distributed Multi Agent Systems. Prac. App. Agent Multi- agent Sys (PAAM), March (1998) 377-391
32. Riehle, D.: Describing and Composing Patterns Using Role Diagrams. In: Smolyani, A., Shestialtynov, A. (eds.). WOON'96 (1st Int'l Conference on Object-Orientation in Russia), Conference Proceedings, St. Petersburd, Electrotechnical University (1996)
33. Sommerlad, P.: Manager. In: Marting, R. Riehle, D., Buschmann, F. (eds.): Pattern Languages of Program Design 3. Addison Wesley (1998) 19-28
34. Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and Practice. The Knowledge Engineering Review 10 (2) (1995) 115-152
35. Zammuto, R.F., O'Conner, E.J.: Gaining Advanced Manufacturing Technologies Benefits: the Role of Organizational Design and Culture. Academy of Management Review (1992) 710-728
36. Zhang, T., Kendall, E.A.: Agent-based Information Gathering System for Life Cycle Costs. Proceedings of the 1st Asia-Pacific Conference on Intelligent Agent Technology (1999) 483-487

Teamwork and Adjustable Autonomy in Agents

Nancy E. Reed

Department of Computer and Information Science
Linköping University, SE-581 83 Linköping, Sweden
nanre@ida.liu.se

Abstract. The workshop on teams and adjustable autonomy brought together researchers working on several aspects of agents coordination The workshop activities focused on aspects of autonomy from communication protocols and computational models supporting adjustable autonomy to designing and implementing systems capable of achieving adjustable autonomy through interaction with a human user or other agents.

1 Introduction

Adjustable autonomy allows systems to operate with dynamically varying levels of independence, intelligence, and control [Barber *et al.*, 2000]. A human user, another system, or the autonomous system itself may adjust an agent's "level of autonomy" as required by the current situation. An agent's adjustable autonomy can involve changes in:

- The authority and role the agent has in or outside of a team.
- The goals of the agent and the teams it is a member of.
- The other members recognized as belonging to the same team(s).
- The complexity of the commands executed and the constraints that are enforced.
- The resources (including time) consumed by an agent's operation.
- The circumstances under which an agent will either override or allow manual control.
- The circumstances under which an agent will request user information or control.
- The number of subsystems that are being controlled autonomously and how they communicate and interact.

Three recent workshops have focused on defining the new area of adjustable autonomy, the AAAI '99 Spring Symposium on Agents with Adjustable Autonomy [Musliner and Pell, 1999], the Workshop on Autonomy Control Software at Agents '99 [Hexmoor, 1999] and the Workshop on Adjustable Autonomy Systems at IJCAI '99 [Kortenkamp *et al.*, 1999]. This workshop's aim was to build on the previous work and advance this area with a focus on adjustable autonomy in multi-agent systems and/or systems with people and agents doing cooperative problem solving.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 301–302, 2001.

The implementation of autonomy in a team can be (a) fixed by indirect factors such as roles and obligations, (b) dynamically varying to allow for varying levels of contribution of members toward the team, or (c) learned. The control system may be completely in control, it may be supervising manual control, or it may be somewhere in between. A system that can shift amongst these control extremes in a safe and efficient manner is a system with adjustable autonomy. Adjustable autonomy provides the flexibility to adapt to changing environments, user goals, and resources. This flexibility is important in order to successfully deploy autonomous system solutions for highly complex real-world problems.

This collection includes several papers with research results, an invited talk summary, and a panel discussion summary.

2 Program Committee

Acknowledgements. Thank you to all the workshop participants and authors. A special thanks to the workshop committee, reviewers, and editors at Springer.

References

[Barber *et al.*, 2000] K. Barber,A. Goel, and C. Martin. Dynamic adaptive autonomy in mult-agent systems. *Journal of experimental and theoretical artificiail intelligence,* 12(2):129–148, 2000.

[Hexmoor, 1999] Henry Hexmoor, editor. *Workshop on Autonomy Control Software.* Autonomous Agents 1999, May 1999.

[Kortenkamp *et al.,* 1999] D. Kortenkamp, G. Dorias, and K. Myers, editors. *Proceedings of IJCAI99 Workshop on Adjustable Autonomy Systems,* August 1999.

[Musliner and Pell, 1999] David Musliner and Barney Pell, editors. *Agents with Adjustable Autonomy.* AAAI Spring Symposium, May 1999.

A Communication Protocol Supporting Dynamic Autonomy Agreements in Multi-agent Systems

K. Suzanne Barber, Cheryl E. Martin, and Ryan M. McKay

The Laboratory for Intelligent Processes and Systems, ENS 22
Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712
{barber|cemartin|rmckay}@mail.utexas.edu

Abstract. Given the challenges faced by agent-based systems including dynamically changing environments, uncertainty, and failures, agent research must explore techniques to make these systems ever more flexible and adaptive. Adaptive systems are more robust and often perform better given changing situations. One area of research that offers promise for improved agent-based systems is adaptive autonomy. In general, adaptive autonomy gives agents the ability to seek help to solve problems or take initiative when otherwise they would be constrained by their design to follow some fixed procedures or rules for interacting with other agents. In order to gain run-time flexibility and any associated performance improvements, agents must be able to adapt their autonomy during system operation. This paper provides a mechanism through which agents can achieve this end. The examples presented in this paper focus on adapting one dimension of autonomy, specifically the decision-making framework under which agents solve problems in their system. The communication protocol presented here allows agents to dynamically modify their autonomy at run-time by forming agreements with other agents about autonomy-altering interactions. Conversation state machines and example event traces are provided. This communication protocol forms a solid foundation around which systems capable of dynamic adaptive autonomy can be implemented.

1 Introduction

Agents should be allowed to dynamically adapt their autonomy to their situation during system operation. In general, adaptive autonomy gives agents the ability to seek help when solving problems or take initiative when otherwise they would be constrained by their design to follow some fixed procedures or rules for interacting with other agents. This paper presents a communication protocol through which agents can dynamically agree about autonomy changes during system operation. The examples presented focus on adapting one dimension of autonomy, specifically the decision-making framework under which agents solve problems in their system.

The next section presents the interpretation of "autonomy" used by this research. This interpretation describes what is actually being manipulated through the given communication protocol. The following sections describe the motivation for a communication protocol used to implement dynamic adaptive autonomy and discuss related work. The communication protocol itself is then presented including message

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 303–320, 2001.

types, conversation state machines, and example conversation traces. Finally, the assumptions of the conversation model are explored along with failure modes, and conclusions and future work are discussed.

2 Autonomy as Decision-Making Control

There is no widely agreed-upon definition of agent autonomy. However, in order to motivate and implement adaptive autonomy, a working definition must be presented. This research considers agent autonomy with respect to decision-making control and the authority to assign tasks. "Decision-making control" and "authority-over" dictate how agents interact to determine a solution during collaborative problem solving with respect to a given goal the agents are pursuing. A specification of "decision-making control" dictates which agents make decisions about how to achieve a goal. A specification of "authority-over" dictates to which agents the decision-makers can assign tasks (i.e. which agents the decision-makers have authority over). These relationships are described by a decision-making framework. An agent's degree of autonomy is determined by the decision-making frameworks in which it participates [4]. An agent's degree of autonomy can be described qualitatively along a spectrum as shown in Figure 1. The three discrete autonomy level categories labeled in Figure 1 define salient points along the spectrum, as described below:

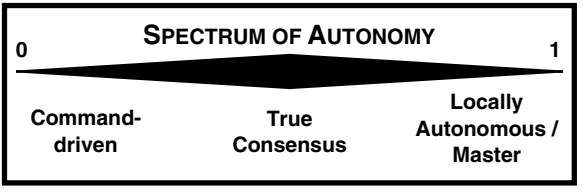


Fig. 1. The Autonomy Spectrum

Command-driven – The agent does not make any decisions about how to pursue its goal and must obey orders given by some other agent(s).

True Consensus – The agent works as a team member, sharing decision-making control equally with all other decision-making agents.

Locally Autonomous / Master – The agent makes decisions alone and may or may not give orders to other (command-driven) agents.

Agents in a multi-agent system can move along the autonomy spectrum during system operation by forming, dissolving, or modifying decision-making interactions with other agents. The research presented here provides a communication protocol through which these autonomy changes can be realized.

A representation of autonomy must be used to allow the agents to converse about autonomy changes. The representation of autonomy as decision-making control has been formalized as a tuple (G, D, C) [4], where G is a set that describes the goal or

goals that are the focus of the decision-making, *D* is a set that identifies the decision-makers and their relative strengths in the decision-making process, and *C* is a set that declares the authority-over constraint. Table 1 gives some simple "Me-You" examples of the use of this representation where "me" is one agent and "you" is another agent. For a complete description of this representation and the classification algorithm used to map the representation to the autonomy spectrum, see [3].

Table 1. Simple examples of autonomy representation instances in "Me-You" terms

As the "me" agent, if my autonomy representation specifies...	Then I am...
G (Focus) = { MyGoal } *D* (Decision-Makers) = { (Me,1) } *C* (Authority-over) = { Me }	**Locally Autonomous**: I make decisions about how to achieve my goal alone with rank 1 (highest ranking decision-maker), and only I am bound to implement the decisions I make.
G = { MyGoal } *D* = { (You,1) } *C* = { Me }	**Command Driven:** You, as a master agent, make decisions about how to achieve my goal, and I am bound to implement the decisions you make.
G = { MyGoal, YourGoal } *D* = { (Me,1), (You,1) } *C* = { Me, You }	**True Consensus:** We are both working together in true consensus (equal ranking decision-makers) to make decisions about how to achieve both of our goals simultaneously. We are both bound to implement the group's decision.

One additional representational element must be established before dynamic autonomy can be realized. This element specifies limits on an agent's ability to modify its own autonomy. Agents should model and enforce commitments to their established autonomy agreements. This "autonomy-level commitment" helps ensure that agents will actually participate as expected in their established decision-making frameworks. In addition, the implementation of such a commitment impacts several other important aspects of agent-based systems capable of adjustable autonomy including trust and stability. Agents explicitly committed to a specific interaction style can be trusted to a greater extent (by designers or users) not to adjust their autonomy in an unpredictable manner. Also, autonomy-level commitment puts a control on how free agents are to continually adjust their autonomy as opposed to actually working to solve a problem at a certain autonomy level. In general, commitments are not unbreakable. Often, there is some price that an agent can pay if it wants to renege on a commitment. In addition, certain conventions [11] allow a commitment to be broken without penalty under certain conditions (e.g. loss of communication). All parties to the commitment should agree a priori on these conventions.

In summary, this research views agent autonomy as decision-making control, and it views dynamic adaptive autonomy as an agent's ability to modify the decision-making frameworks in which it participates, dynamically, during system operation. The authors' previous experiments have provided empirical motivation for this type of adaptation. These experiments have shown that no one decision-making framework performs best across all conditions that may be encountered at run-time

[2]. Given these results, the implementation of dynamic adaptive autonomy should be further explored.

3 Why Use Communication to Adapt Autonomy?

An agent's autonomy must be defined with respect to other agents in the system (in the sense that autonomy is "freedom from intervention" by such agents). The concept of autonomy as decision-making control refers to an agent's ability to make decisions about how to pursue its goals without intervention by any other agent. A *decision-making framework* specifies an agent's degree of decision-making control with respect to a given goal it is pursuing [4]. In this context, an agent's degree of autonomy depends directly on how much decision-making control it retains over its goals compared to the decision-making control exercised by other agents for that goal. In order to change autonomy in a dynamic fashion during system operation, agents must establish or modify decision-making frameworks. Agents must establish coherent views of the framework under which they are operating together in order for this inter-operation to be meaningful. For example, an agent who is considered to be command-driven by some master should not consider itself to be locally autonomous. Therefore, all agents involved in a decision-making framework must agree about autonomy changes before they can be implemented. Agents must establish an agreement to work together under a specified decision-making framework each time more than one agent is involved.

Negotiation through communication is one way to establish such an agreement. Alternatively, agents can be designed such that they are guaranteed to all choose and implement coherent decision-making frameworks independently, similar to the coordination approach used in [16]. If this approach is taken, then communication is not needed. However, negotiation through communication is the preferred way to reach such inter-agent agreements [23]. Using communication and negotiation does not require a hard-coded coordination policy and it does not require that each agent maintain a consistent worldview, which is needed for independent implementation of the same coordination decision. This paper provides a communication protocol to establish agreements about autonomy changes.

Note that establishing one type of decision-making framework (locally autonomous) requires no external autonomy agreements from other agents. The instantiation of all other decision-making frameworks requires an autonomy agreement to be established. The locally autonomous case can be viewed as the absence of an autonomy agreement, and the agent must work alone. However, the agent must still consider the cost of changing its autonomy to form a locally autonomous decision-making framework. For example, if an agent who is command-driven for a particular goal determines that it no longer wants to participate in its master/command-driven relationship, the agent must renege on a previously formed agreement to avoid following the master's orders in the future. If the command-driven agent determines that it wishes to become locally autonomous for its goal, it can establish the new decision-making framework without consulting other agents; however, it must first dissolve its present command-driven decision-making framework and pay the required autonomy-level commitment penalty for doing so.

4 Related Work

Much previous work has analyzed the formal semantics of agent communication languages, messages, and conversation policies [7; 8; 10; 12; 18; 20]. Analyzing communicative acts with respect to a theory of joint intentions has allowed researchers to show that agents are capable using communicative acts to form the necessary mental states to work together as a team [19; 20]. With respect to the autonomy agreement communication protocol, the "joint intention" formed by agents is to work together under the specified decision-making framework.

The protocol presented in this paper extends the authors' previous work on the Goal and Responsibility Allocation Protocol and Language (GRAPL) [13; 14]. GRAPL allows agents to negotiate about goal (or task) allocation and the allocation of decision-making control (responsibility to plan for a goal). GRAPL is based on a classic task allocation protocol – The Contract Net Protocol (CNP) [17; 21]. The CNP is designed to allow one party to announce a task to be done, and receive bids on the "contract" from multiple interested parties. The announcing party can then choose the best bidder and award the contract. In the context of the GRAPL protocol, agents propose and bid on decision-making frameworks. The agents who receive the proposal "bid" if they agree to accept the proposal. For GRAPL, with respect to autonomy agreements, the "bid" is binary (present or absent). This differs from the CNP where the bid contains some information about how much the agent expects to gain by accepting the contract proposal. GRAPL also differs from the CNP in that it introduces a counter-propose mechanism whereby the announcing or bidding agent can modify the proposal under consideration. Agents evaluate the current version of the proposal to determine whether or not to bid on a proposal.

The current protocol differs from GRAPL in several ways. First, because GRAPL is based on the CNP, its focus is tied to a market-based matchmaking process among agents. This is useful when the proposing agent broadcasts a task announcement to several other agents who are indistinguishably qualified to perform the task, and when those agents will "charge" the proposing agent to various degrees for actually performing the task. However, recent investigation and implementation indicates (1) that autonomy agreements are more directed than general task allocation and (2) that the autonomy agreement process is better modeled as a staged negotiation rather than market-based task allocation. In general, autonomy agreements involve directed communication because an agent proposing a particular agreement has very specific desires about which other agents should be involved [3]. In addition, the agents who respond to an autonomy proposal do not respond with a variable bid, they are limited to actions of accept, reject, or proposing an alternative.

Another critical difference between GRAPL and the current autonomy agreement protocol is the newly developed ability to support multi-agent autonomy negotiations. The GRAPL protocol and underlying CNP foundation support conversations between only two agents at once. It is difficult to establish autonomy agreements involving three or more agents using GRAPL because multiple concurrent two-party conversations must be coordinated. The newly developed protocol explicitly models the autonomy agreement conversations as multi-agent interactions, and is scaleable to any number of participants.

5 Autonomy Agreement Communication Protocol

The autonomy agreement protocol presented here has been developed for use by Sensible Agent-based systems [1]. Sensible Agents are agents with a particular architecture that have been designed to reason about autonomy and implement dynamic autonomy assignments. During the *autonomy reasoning* process, a respective Sensible Agent selects the most desired decision-making framework for each of the goals it intends to pursue. If these selections involve other agents, these other agents must agree (if the framework is to be implemented) to participate in collaborative decision-making under the desired decision-making framework.

The autonomy agreement communication protocol is based on a propose/counter-propose paradigm in which agents ask other agents to participate in their desired decision-making frameworks. As shown in Table 2, each proposal or counterproposal contains (1) a unique identifier, (2) a specification of the desired decision-making framework using the autonomy representation ((*G*, *D*, *C*) as described in the "Autonomy as Decision-Making Control" section), and (3) the proposed autonomy-level commitment for each involved agent. For the purposes of the following discussions, an *autonomy proposal* is the communicated representation of the desired decision-making framework, and an *autonomy assignment* is the resulting implementation of that decision-making framework by each agent, should the proposal be accepted. Table 2 gives an example of a possible autonomy proposal and its resulting autonomy assignments. The autonomy proposal and the resulting autonomy assignments made by each agent are similar, however the individual autonomy assignments may not reflect all information contained in the proposal. For example, in Table 2, each agent represents only its own al_commitment in its resulting assignment. Once the autonomy assignments are made, the agents can begin collaborative problem solving under the desired decision-making framework. The "autonomy_agreement_id" identifies the conversation under which an autonomy agreement is formed. This identifier ties all conversation messages together during the autonomy negotiation process, and it later serves as an identifier for agent communications during collaborative problem solving for the associated goals. The "proposal_id" identifies the current proposal being considered in the agreement conversation.

All agents who receive proposals may accept, reject, or counter-propose. In the end, the original proposing agent can confirm or deny the agreement. Once an agreement is confirmed, all involved agents make the specified autonomy assignments to their respective goals. The agent who originally proposes an autonomy agreement will be referred to, from this point forward, as the agreement *manager*. All other agents involved in the agreement conversation will be referred to as *participants*. Any agent may take either role for a given conversation. Any agreement conversation may involve one manager and one or more participants. A critical contribution of the protocol presented here is the ability to handle multiple participants, allowing autonomy agreements and resulting decision-making frameworks to involve more than two agents.

In general, an agent who intends to achieve a goal will attempt to achieve it alone or request help from other agents to achieve it. Therefore, in a master/command-driven relationship, it is likely to be the command-driven agent who originally

Table 2. Example autonomy proposal and resulting assignments

Autonomy Proposal

{ autonomy_agreement_id = A_22;
proposal_id = A_114;
G = { $goalX_{AgentA}$, $goalY_{AgentB}$ };
D = { (AgentA, 1), (AgentB ,1) };
C = { AgentA, AgentB };
al_commitments =
{ (AgentA, 4), (AgentB, 3) };
}

Resulting Autonomy Assignment (Agent A)	Resulting Autonomy Assignment (Agent B)
{ autonomy_agreement_id = A_22; G = { $goalX_{AgentA}$, $goalY_{AgentB}$ }; D = { (AgentA, 1), (AgentB ,1) }; C = { AgentA, AgentB }; al_commitment = (AgentA, 4) }	{ autonomy_agreement_id = A_22; G = { $goalX_{AgentA}$, $goalY_{AgentB}$ }; D = { (AgentA, 1), (AgentB ,1) }; C = { AgentA, AgentB }; al_commitment = (AgentB, 3) }

proposed the decision-making framework (i.e. acted as the manager for the conversation establishing the master/command-driven framework). Before an agreement is established, the command-driven agent is likely to know more about the goal and more about how much help it needs to achieve the goal. In the same way, during the formation of a consensus relationship, one agent is likely to propose a consensus decision-making framework for a goal it intends to achieve, and other agents are likely to add their own goals to the decision-making framework during the counter-proposal phase so the agreement becomes more beneficial to all involved.

5.1 Message Types

Table 3 lists the conversation phases for the autonomy agreement protocol as well as the possible message types seen in each phase. Phase 1 starts the conversation and encompasses only the initial proposal. Phase 2 allows a participant to consider the current proposal and determine whether or not it is willing to commit to the specified decision-making framework. Phase 3 allows the manager to consider counter-proposals from all involved participants, if any have been suggested. If, instead, all participants have accepted the manager's proposal, the manager must then (in Phase 3) determine whether it wants to commit to the specified decision-making framework. The manager does not commit to the proposed agreement with the initial proposal. This gives the manager the flexibility to investigate multiple mutually exclusive possible agreements concurrently. In the end, the manager can commit to only one of these mutually exclusive possibilities. The semantics of this late-commitment paradigm are similar to those formally defined for the "SOFFER" conversation policy by [20]. In this primitive conversation policy, the "offering" agent waits for acceptance from the other agent and then tells that agent whether it is still willing to do the offered action. If the manager decides to commit in Phase 3, it informs the participant(s) that they should implement the agreement. The manager also has the

option to terminate the conversation (deciding not to commit) in Phase 3. Due to the propose/counter-propose cycle, the agents can move back and forth between Phase 2 and Phase 3 as many times as they are willing to consider counter-proposals.

Table 3. Message types and descriptions

Phase 1 (initial proposal):	
PROPOSE	Manager sends initial proposal to participant(s)
Phase 2 (participants' opportunity to commit to agreement):	
P-COUNTERPROPOSE	Participant requests updated or different agreement
ACCEPT	Participant commits to agreement
REJECT	Participant refuses to negotiate further about agreement
FAIL	Manager terminates conversation and informs participant(s) that agreement failed
Phase 3 (manager's opportunity to commit to agreement):	
M-COUNTERPROPOSE	Manager sends updated proposal to participant(s)
CONFIRM	Manager commits to agreement and informs participant(s) to implement
FAIL	Manager terminates conversation and informs participant(s) that agreement failed

5.2 Conversation State Machines

The autonomy agreement protocol proposed by this research uses state machine descriptions to specify allowable behaviors within autonomy agreement conversations. This section describes these state machines. State machines are a useful way to specify the behavior of any system, and have previously been used to model agent communication protocols [5; 23]. The state machines presented here can be implemented directly within agents to guide their determinations of when to perform necessary autonomy reasoning tasks and their determinations of what their next conversation action should be.

Both the manager and participant views are given. Any agent may play either role for a given conversation. A single, global state machine for the conversation is not specified because the agents involved in a given conversation do not have access to global information. Agents could not maintain consistent views of a global conversation state because agents cannot all know the same things at the same times about the conversation. These agents can maintain only their local view of the conversation. These local views should be coherent (i.e. they should not conflict with the views of other agents involved in the same conversation). However, this coherency requirement does not require maintaining a global conversation model wherein each agent models the current state of the other agents involved in the conversation as well as its own current state.

Figure 2 shows the state machine for a 2-agent conversation (1 manager and 1 participant) from the manager's perspective. This state machine enforces a lock-step action for the conversation (manager sends message, participant sends message, manager sends message, etc.). After the initial proposal message, each message acts as an acknowledgement of receipt of the previous message. After sending a given message, an agent (either manager or participant) must wait for a response from the

other agent. (The exception of FAIL messages are discussed below in the section on "Assumptions for Conversation Model"). This "send/ wait-for-acknowledging-response/ send" paradigm ensures that agents maintain non-conflicting local views of a given conversation.

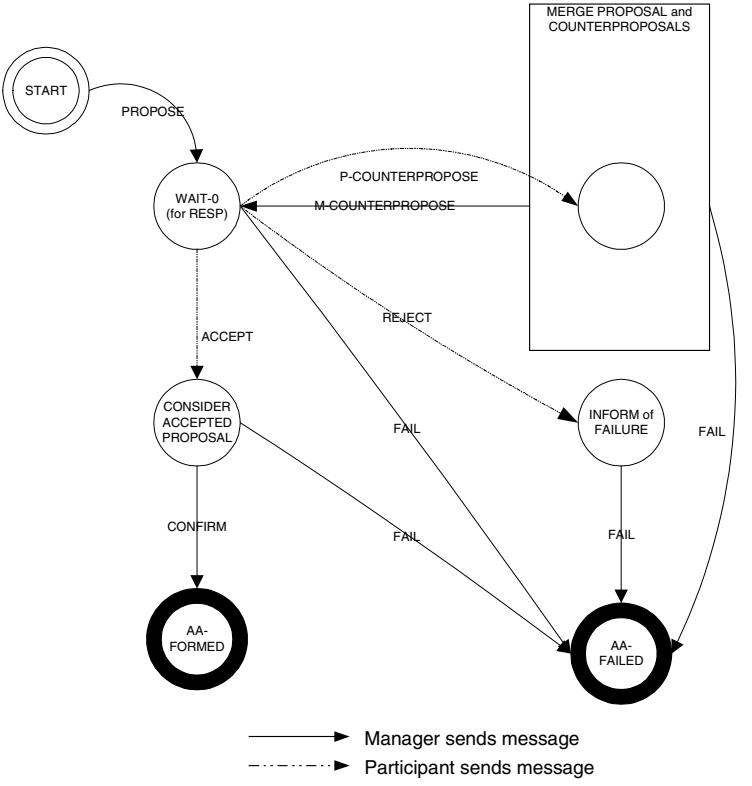


Fig. 2. Manager state machine for 2-agent conversation (1 manager, 1 participant)

As more agents become involved in the decision-making framework, the manager's transitions among states become more complicated. Figure 3 shows the manager state machine for a 3-agent conversation (1 manager, 2 participants). In conversations with multiple participants, the manager can make the transition to the "MERGE PROPOSAL and COUNTERPROPOSALS" meta-state in different ways. If at least one agent counter-proposes and no agent rejects, the manager must attempt to merge all counter-proposals with its original proposal. The manager then re-submits a new counter-proposal to the entire group, keeping every agent "on the same page" for the negotiation. This merging task represents the primary difference between the manager's role and the participants' roles. After every proposal or m-counter-proposal, all agents in the conversation are considering the same decision-making framework. To reach the successful completion of an autonomy agreement conversation, all participants must eventually agree to a proposal sent out by the conversation manager.

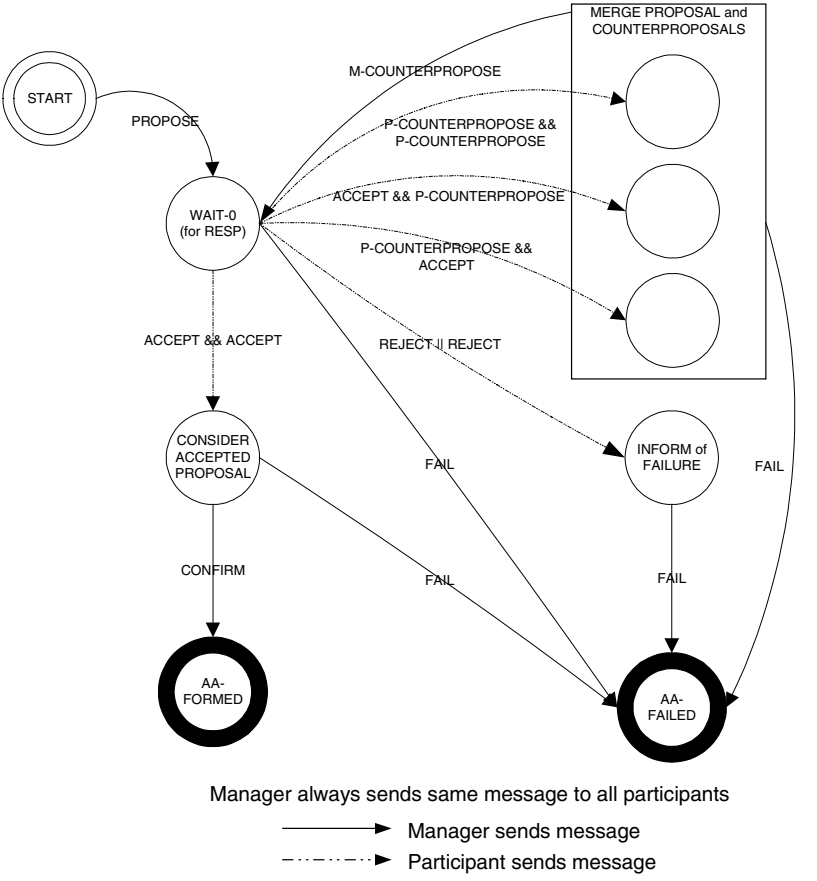


Fig. 3. Manager state machine for 3 agent conversation (1 manager, 2 participants)

Figure 4 shows the participant's state machine. A participant communicates only with the manager and may not know if other participants are involved in the conversation or not. The "CONSIDER PROPOSAL" state is the participant's primary action state. Here the participant decides to accept, reject, or counter-propose in response to the manager's proposal or counter-proposal. Note that if the participant counter-proposes, the manager must also then counter-propose in order to achieve an agreement state. If the participant accepts, it must be prepared to implement the corresponding autonomy assignment in case the manager confirms the agreement. It must also be prepared for the manager to counter-propose even though it has accepted the agreement already. This would occur in cases where there are multiple participants involved in the conversation and at least one of the other participants has counter-proposed rather than accepting the earlier proposal.

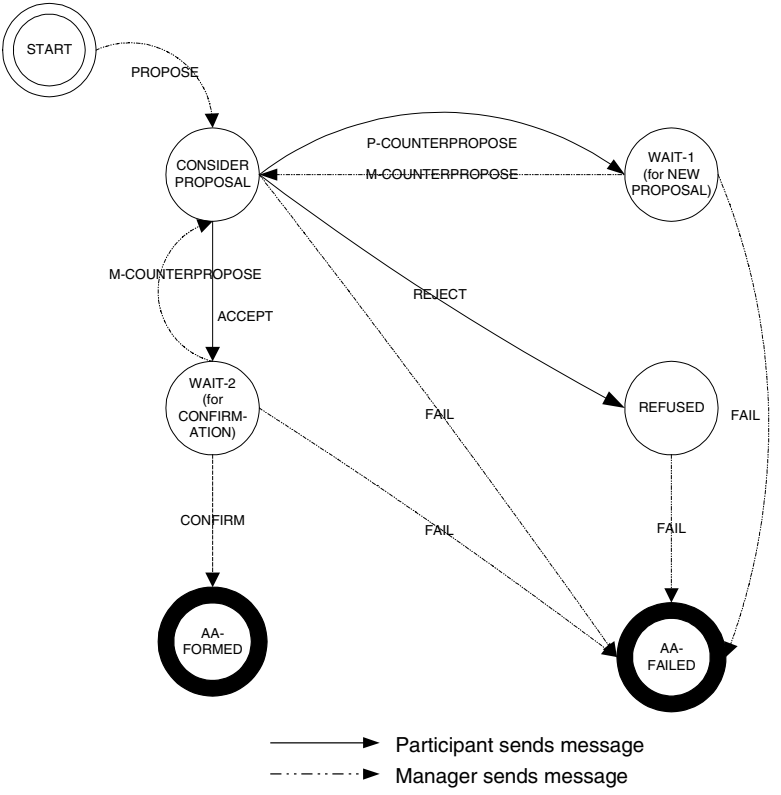


Fig. 4. Participant State Machine

5.3 Example Conversation Traces

In order to give the reader a clearer understanding of this communication protocol, example conversation traces are presented here. Figure 5 shows some example conversation traces for some 2-agent autonomy agreement conversations. Three major conversation types are demonstrated. Figure 5a shows a minimal (in terms of the number of messages required to form an autonomy agreement) conversation between two agents that has a successful outcome. The circles on each event-trace line represent the point at which the corresponding agent reaches a final conversation state for that trace. Figure 5a shows both the manager and participant reaching of final state of "AA-FORMED" (refer back to state diagrams to associate states along each event-trace with messages shown). Figure 5b demonstrates the difference in commitment phases between a manager and a participant. Participants commit to the agreement when they send the AGREE message. However, as described above in Section 5.1, the manager does not commit until it sends the CONFIRM message. Therefore, even if all the participants agree to the proposal, the manager still has the option to fail. Figure 5c shows a type of behavior expected to be relatively common

in autonomy agreement conversations. The counter-proposal process can go on ad infinitum, until one of the parties decides either to accept the latest proposal or to terminate the conversation. Any agent involved in the conversation can terminate a counter-proposal loop. As shown in Figure 2, the manager can FAIL out of the conversation at any point. A regular participant can terminate the conversation only in their CONSIDER PROPOSAL state, by sending a REJECT message to the manager. This second case is shown in Figure 5c. The participant has decided that there is no point to continuing the conversation, because it cannot reach agreement with the manager. As dictated by the state machine in Figure 2, the manager responds by sending out a FAIL message, terminating the conversation.

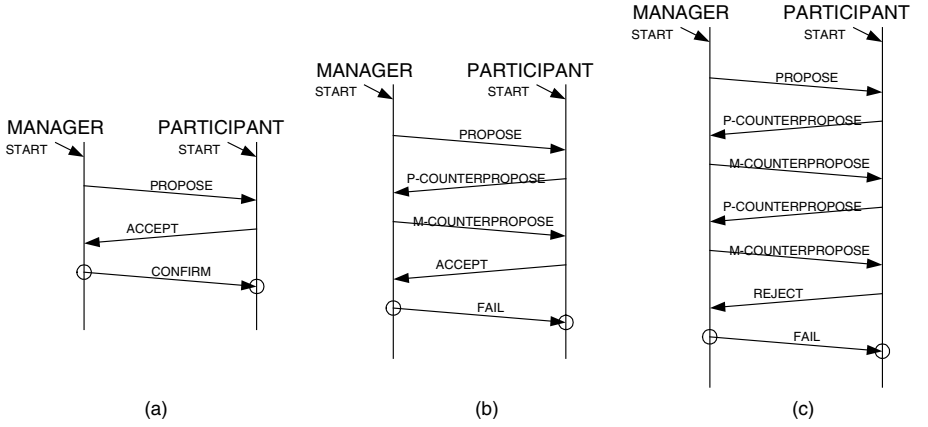


Fig. 5. Sample conversation traces for 2-agent autonomy agreement conversations

Figure 6 shows example conversation traces for some 3-agent autonomy agreement conversations. The conversation shown in Figure 6a is relatively simple, as is any conversation in which one of the participants REJECTs in the first round. Any rejection shunts the conversation to failure. The second conversation, shown in Figure 6b, is more interesting. Although participant1 accepted the proposal, participant2 counter-proposed. When the manager receives any counter-proposals and no rejections, it attempts to merge all of the counter-proposals and the original proposal into a single new proposal. Once the manager has generated what it considers to be the "best" merged proposal, it sends this counter-proposal to all the participants. It cannot simply respond to the counter-proposing agent, because all the participants must be aware of and decide whether to accept the latest proposal issued by the manager. In the example conversation shown in Figure 6b, both agents accept the second proposal issued by the manager. As stated previously, the manager could still fail out of the agreement at this point. However, it decides to commit to the agreement by sending out a CONFIRM message to all the participants.

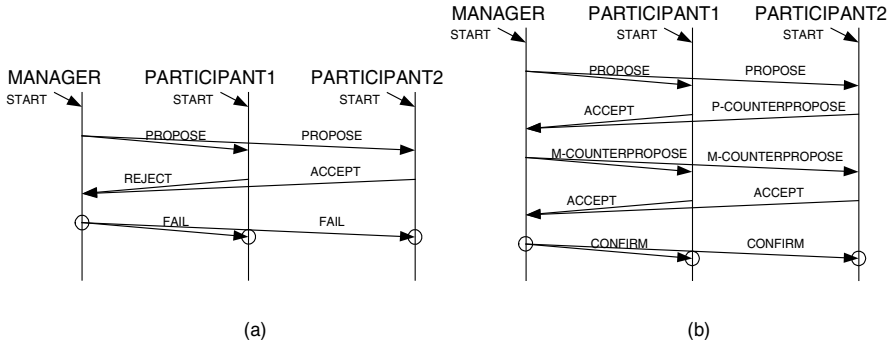


Fig. 6. Example conversation traces for 3-agent autonomy agreement conversations

6 Assumptions for Conversation Model

The conversation protocol presented here assumes asynchronous, reliable point-to-point communication [15]. Communication in the agent-based system is assumed to be asynchronous given the unbounded amount of time an agent may take to respond to a given message. The characteristics of reliable point-to-point communication imply that (1) if an agent sends a message, *m*, then all addressed agents who are functioning correctly will eventually receive and process *m*, (2) no agent receives and processes any message that is not addressed to that agent, (3) if an addressed agent receives and processes message *m*, then all addressed agents who are correctly functioning will eventually receive and process *m*, and (4) for any message *m*, every addressed agent who is correctly functioning receives and processes *m* at most once and only if *m* was previously sent to that agent by *sender*(*m*). Even though communication can fail in the agent-based systems modeled for this research, the transport layer used to send messages still guarantees message delivery if communication ever comes back up.

The autonomy agreement communication protocol is very directed and specific to the autonomy agreement task. Agents are expected to follow the protocol. Received messages are ignored if they do not advance the state of a current conversation via a legal transition or do not start a new conversation with a proposal message. Future work may investigate handling conversational tangents, exceptions, or interrupts via task models that allow agents to interpret the abstract context of the conversation, as shown in [9]. Such problems are often seen in natural-language based systems and human interaction. Currently, the protocol does not support divergent messages or recovery from spurious messages.

Because the communication protocol (in general) allows only one state transition before an acknowledging response is required, a causal ordering constraint is not required. The causal-ordering condition would add the following constraint: for messages *m* and *m*', if *send*(*m*) causally precedes *send*(*m*'), then no agent addressed by both messages who is correctly functioning will receive and process *m*' unless it

has previously received and processed *m*. The remainder of this section discusses the only exception to the requirement for acknowledging responses.

Given the protocol specified by the state machines and the conversation model assumptions, the agents generally proceed in lock-step for the given conversation protocol. First, the manager proposes to all participants, then all participants must respond, then the manager must respond to all participants, etc. The only exception to this procedure may occur in Phase 2, where the manager may terminate the conversation by sending a FAIL message before all participants have responded. This situation is described further by the conversation traces below, and relates to the failure model assumed by this conversation protocol, described here.

This autonomy agreement protocol currently assumes a crash failure model [15]. Either the manager or any number of the participants may fail by halting. Failed agents remain halted (for the purposes of the conversations in which they are participating) and cannot re-enter a conversation at a later time. The fact that an agent has failed may not be detectable by other agents in the system. Because communication is assumed to be asynchronous, it is not possible to distinguish between an agent that has crashed and one that is responding very slowly. Therefore, both possibilities must be considered by the protocol. If the manager fails, the participants currently do nothing, they maintain the conversation and may wait forever to receive a message from the manager. This is not a critical problem because agents may carry on multiple conversations at once, so another decision-making framework can be adopted over the stopped conversation. If the manager is just slow, the participants eventually receive and process messages from the manager as usual. If a participant fails or is very slow to respond in Phase 2, the manager may use a time-out to mark the participant as failed. At this point, the manager uses its Phase 2 option to send out a FAIL message to every participant, terminating the conversation. The manager may also use this Phase 2 option to preemptively terminate a conversation where one participant has refused to continue the conversation (REJECT) and the other participants have not yet responded. The examples below explore these possibilities as well as the scenario in which the participant has not actually failed and does eventually respond to a proposal after the manager has sent the FAIL message.

Figure 7 shows an example of the manager deciding not to wait any longer for a response from one of the participants. In this example, the manager sent a proposal to two agents, and received an acceptance message from one of them, but nothing from the other. After a period of time, the manager gives up on receiving a response from participant2 and fails out of the conversation. Because communication is asynchronous, the algorithm used by the manager to decide how long is “too” long may lead the manager to fail too soon. This possibility is explored in the next example.

In general, if the manager receives a message associated with a particular conversation after it has already reached a final state for that conversation, the manager will ignore that message. Final conversation states are represented on the event trace diagrams by the circles surrounding the endpoints of the CONFIRM and FAIL messages. Any time a manager fails out of a conversation before it has received responses from all participants, the manager can potentially receive the

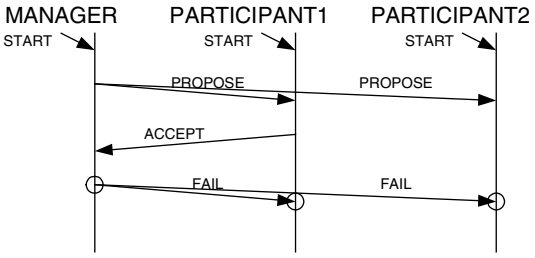


Fig. 7. 3-agent conversation trace in which Participant 2 has crashed

participant's responses after it has reached a final state. Examples of this behavior are shown in Figure 8 and Figure 9. In Figure 8, the participant may have waited too long to respond. Alternatively, the manager may have been involved in multiple conversations about mutually exclusive autonomy agreements. As soon as the manager committed to one of these agreements, it would fail out of the rest. Although the participant does respond in Figure 8, the response arrives at the manager after it has sent out the FAIL message and has reached its final conversation state. The manager ignores the final ACCEPT message from the participant. Note that the participant also reaches a final conversation state as usual. In this case the circled, final state in the event-trace corresponds to the state "AA-FAILED" for each agent involved in this conversation.

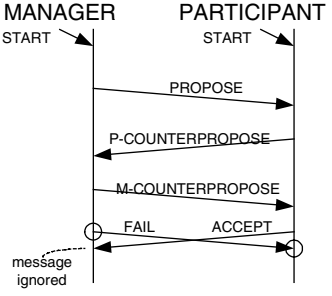


Fig. 8. 2-agent conversation in which Participant responds too slowly

Another motivation for allowing a Phase 2 FAIL message from the master concerns early rejection. In this case a manager receives a rejection from one of the participants before all participants have responded. In order to save time and processing for all the agents involved, the manager can immediately send out the FAIL message. An example of this case is shown in Figure 9. Although participant2 responds, the manager has already reached its final conversation state and will ignore the message. All participants reach a final conversation state as well.

The conversation model, failure assumptions, and recovery options presented here support the basic implementation of dynamic adaptive autonomy. Future work will

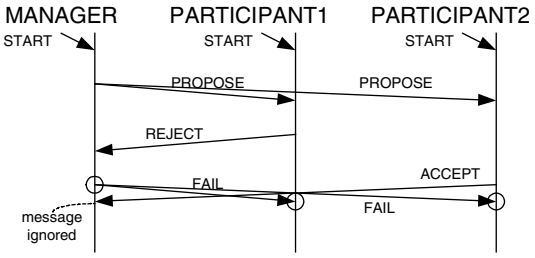


Fig. 9. 3-agent conversation in which Manager sends FAIL message before waiting for all Participant responses.

extend the protocol for cases where the communication medium is lossy or agents may recover and wish to resume a previous conversation.

7 Conclusions

This paper presents a communication protocol that can be used by any number of agents to establish a collaborative decision-making framework during system operation. By allowing agents to change the decision-making frameworks under which they pursue their goals, this communication protocol supports dynamic adaptive autonomy, where autonomy is viewed and represented as decision-making control. The communication protocol presented here provides key advancements over the previously used GRAPL protocol, including a shift to negotiation-oriented interaction rather than contract/bid-oriented interaction and the ability to scale to multi-agent agreements rather than being limited to two-agent agreements. Failure modes for conversations using this communication protocol are examined in this paper using a crash failure model for all involved agents.

Future theoretical work includes specifying formal semantics for the communicative acts encompassed by the proposed protocol. These formal semantics will facilitate analysis and verification of agent behavior. Additionally, using a common formal semantics will allow Sensible Agents using this protocol to interact with other heterogeneous agents in the future. Other potential future work includes generalization of the negotiation process so that multiple different communication protocols could be used and compared.

Parallel research is being conducted regarding the algorithms agents use to choose among the legal transitions at each conversation point in the proposed protocol. For many of the transitions, this requires the ability to generate or evaluate autonomy agreement proposals. Agents must choose, at every point, which autonomy proposals they wish to implement. Quite a bit of research has been done on two-agent utility-theoretic negotiation algorithms, e.g. [6; 22; 24]. It is quite likely that a utility-theoretic approach is feasible for this problem, and this existing work could be applied. However, to do so, these results must be generalized to handle *n*-agent negotiations. Other future work includes implementing the proposed protocol and testing it with respect to run-time performance. Future work will also evaluate the protocol with respect to more complex models of agent failure and communication

failure. Implementation of these research concepts has already begun, and will be tested along with the proposed protocol.

Acknowledgements. This research was supported in part by the Texas Higher Education Coordinating Board (#003658452), a National Science Foundation Graduate Research Fellowship, and the Defense Advanced Research Projects Agency and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-00-2-0588.

References

1. Barber, K. S., Goel, A., Graser, T. J., Liu, T. H., Macfadzean, R. H., Martin, C. E., and Ramaswamy, S.: Sensible Agents. In Proceedings of 1997 IEEE International Conference on Systems, Man, and Cybernetics (Orlando, FL, 1997)
2. Barber, K. S., Goel, A., and Martin, C. E. The Motivation for Dynamic Adaptive Autonomy in Agent-based Systems. In Intelligent Agent Technology: Systems, Methodologies, and Tools. Proceedings of the 1st Asia-Pacific Conference on IAT, Hong Kong, December 14-17, 1999, Liu, J. and Zhong, N., (eds.). World Scientific, Singapore, (1999) 131-140.
3. Barber, K. S., Goel, A., and Martin, C. E.: Dynamic Adaptive Autonomy in Multi-Agent Systems. The Journal of Experimental and Theoretical Artificial Intelligence, Special Issue on Autonomy Control Software, 12, 2 (2000) 129-147.
4. Barber, K. S. and Martin, C. E.: Agent Autonomy: Specification, Measurement, and Dynamic Adjustment. In Proceedings of Autonomy Control Software Workshop at Autonomous Agents (Agents'99) (Seattle, WA, 1999) 8-15.
5. Barbuceanu, M. and Fox, M. S.: COOL: A Language for Describing Coordination in Multi-Agent Systems. In Proceedings of First International Conference on Multi-Agent Systems (San Francisco, CA, 1995) AAAI Press/ The MIT Press, 17-24.
6. Barbuceanu, M. and Lo, W.-K.: A Multi-Attribute Utility Theoretic Negotiation Architecture for Electronic Commerce. In Proceedings of Fourth International Conference on Autonomous Agents 2000 (Barcelona, Catalonia, Spain, 2000) ACM, Inc., 239-246.
7. Cohen, P. R. and Levesque, H. J. Communicative Actions for Artificial Agents. In Software Agents, Bradshaw, J. M., (ed.). AAAI Press/The MIT Press, Menlo Park, CA, (1997) 419-436.
8. Cohen, P. R., Morgan, J., and Pollack, M. E.: Intentions in Communication, The MIT Press, Cambridge, MA, 1990,.
9. Elio, R., Haddadi, A., and Singh, A. Task Models, Intentions, and Agent Conversation Policies. In PRICAI 2000 Topics in Artificial Intelligence. Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence Melbourne, Australia, August/September 2000. Lecture Notes in Artificial Intelligence, Mizoguchi, R. and Slaney, J., (eds.). Springer, Berlin, (2000) 394-403.
10. Finin, T., Labrou, Y., and Mayfield, J. KQML as an Agents Communication Language. In Software Agents, Bradshaw, J. M., (ed.). AAAI Press/The MIT Press, Menlo Park, CA, (1997) 291-316.
11. Jennings, N. R. Coordination Techniques for Distributed Artificial Intelligence. In Foundations of Distributed Artificial Intelligence. Sixth-Generation Computer Technology Series, O'Hare, G. M. P. and Jennings, N. R., (eds.). John Wiley & Sons, Inc., New York, (1996) 187-210.

12. Labrou, Y., Finin, T., and Peng, Y.: Agent Communication Languages: The Current Landscape. IEEE Intelligent Systems, 14, 2 (1999) 45-52.
13. McKay, R. and Barber, K. S.: Goal and Responsibility Allocation in Sensible Agent-based Systems. In Proceedings of Fifteenth National Conference on Artificial Intelligence (Madison, WI, 1998) AAAI Press / The MIT Press, 1195.
14. McKay, R. M. Communication Services for Sensible Agents. Master's Thesis, Electrical and Computer Engineering, University of Texas at Austin, 1999.
15. Mullender, S.: Distributed Systems, , 2nd ACM Press/Addison-Wesley, New York, 1993, 580.
16. Osawa, E.-I.: A Metalevel Coordination Strategy for Reactive Cooperative Planning. In Proceedings of First International Conference on Multi-Agent Systems (San Francisco, CA, 1995) AAAI Press / The MIT Press, 297-303.
17. Sandholm, T. and Lesser, V. R.: Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. In Proceedings of First International Conference on Multi-Agents Systems (San Francisco, CA, 1995) 328-335.
18. Singh, M. P. Multiagent Systems: A Theorical Framework for Intentions, Know-how, and Communications. Springer-Verlag, Berlin, 1994.
19. Smith, I. A. and Cohen, P. R.: Toward a Semantics for an Agent Communication Language Based on Speech-Acts. In Proceedings of Thirteenth National Conference on Artificial Intelligence (Portland, Oregon, 1996) AAAI Press, 24-31.
20. Smith, I. A., Cohen, P. R., Bradshaw, J. M., and Greaves, M.: Designing Conversation Policies Using Joint Intention Theory. In Proceedings of Third International Conference on Multi-Agent Systems (Paris, France, 1998) IEEE, 269-276.
21. Smith, R. G.: The Contract Net Protocol: High-level Communication and Control in a Distributed Problem-Solver. IEEE Transactions on Computers, 29, 12 (1980) 1104-1113.
22. Soo, V.-W. Agent Negotiation under Uncertainty and Risk. In Design and Applications of Intelligent Agents. Third Pacific Rim International Workshop on Multi-Agents, PRIMA 2000 Melbourne, Australia, August 2000. Lecture Notes in Artificial Intelligence, Zhang, C. and Soo, V.-W., (eds.). Springer, Berlin, (2000) 31-45.
23. von Martial, F. Coordinating Plans of Autonomous Agents. Springer-Verlag, Berlin, 1992.
24. Zlotkin, G. and Rosenschein, J. S.: Negotiation and Task Sharing Among Autonomy Agents in Cooperative Domains. In Proceedings of Eleventh International Joint Conference on Artificial Intelligence (Detroit, Michigan, 1989) IJCAI, Inc., 912-917.

Designing Human-Centered Autonomous Agents

Gregory Dorais[1] and David Kortenkamp[2]

[1] NASA Ames Research Center, Moffett Field CA, 94035
gadorais@ptolemy.arc.nasa.gov
[2] NASA Johnson Space Center – ER2, Houston, TX 77058
kortenkamp@jsc.nasa.gov

1 Introduction

Human-centered automation (HCA) maximizes the goals of humans by supporting a full range of interactions between humans and autonomous systems. The key goal of this research is to minimize the *necessity* for human interaction, but maximize the *capability* to interact. Within HCA we define *adjustable autonomy* as the ability of autonomous systems to operate with dynamically varying levels of independence, intelligence and control. HCA encompasses adjustable autonomy and interfaces it with Human-Computer Interaction (HCI).

The motivations for human-centered automation are many. They include technical issues such as an inability to automate certain aspects of a task because of its complexity or because of an uncertain or changing operating environment. They also include non-technical issues such as a desire to allow for human intervention even when full autonomy is possible. These latter motivations may include safety, training, maintenance or calibration.

The benefits of human-centered automation include the ability to partially automate a system when full automation is not possible. Other benefits are lower costs because difficult-to-automate parts of the system can be left to humans and increased operator acceptance of an autonomous system.

Early work in human-centered autonomous systems has been conducted at NASA Johnson Space Center [Bonasso *et al.*, 1997, Kortenkamp *et al.*, 2000], at NASA Ames Research Center [Dorais *et al.*, 1998], at the Honeywell Technology Center [Musliner and Krebsbach, 1999] and the University of Texas [Barber *et al.*, 1999].

2 A Simple Example

To demonstrate adjustable autonomy, let's take the example of a tank that needs to be kept at a specific pressure. The tank has two ways of pressurizing: a motor that is controlled electrically and a crank that is turned manually. The tank has two sensors: an analog pressure gauge that must be read by a human and a digital pressure gauge that can be read by a computer. Finally, the system has a controller. The system is summarized as follows:

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 321–324, 2001.

- Controller
  - human who decides whether pressure should be increased
  - computer that decides whether pressure should be increased
- Actuator
  - pump that human cranks
  - pump that motor activates increased
- Sensor
  - analog pressure sensor that human reads
  - digital pressure sensors that computer reads

This translates into eight different autonomy modes. For example, a fully autonomous system would use computer control with the motorized pump and the digital pressure sensor. A fully manual system would use the human controller, hand-cranked pump and analog pressure sensor. However, in addition to these two extremes there are six other partially autonomous configurations. For example, the computer could decide when the human should turn the crank by watching the digital pressure sensor. In this case, the computer is the controller and the human is the actuator. Or the human could read the analog pressure sensor and enter the value into the computer, which would do the control. From this simple example you can see that adjustable autonomy is the ability of the system to move amongst these different configurations as required.

3 Requirements

Building an adjustably autonomous control system to support HCA places severe requirements on the underlying control system. Without a properly designed control system, adjustable autonomy can be ineffective and even dangerous. It can be difficult to "retrofit" adjustable autonomy into existing autonomous or tele-operated control systems. There are several key requirements for a well-designed adjustably autonomous control system:

- A human (or other agent) who wants to adjust autonomy needs to know what the control system knows and what it is doing. This includes the following:
  - *State*: The state is a set of values that represent the current abstraction of the system (internal state) and its environment (external state). The human should be able to read and update the internal states of the control system and the control system's perceived state of the world.
  - *Models*: Models define the set of possible states and their relationships. Models should be presented in a way that is easily understood by humans.
  - *Goals*: A goal is a desired set of states. The user needs to know the system's current goals and its progress in achieving those goals. The system may need to explain non-linearities, e.g., backtracking.
  - *Tasks:* The tasks are the actions the system is currently taking to achieve its goals. The human needs to be able to see those tasks, adjust them and add to them if necessary.

- A system can be adjustably autonomous only if it can be commanded by an outside agent. Commanding can take many forms, including physical actuation, setting goals, changing models or executing procedures.
- Adjustable autonomy only applies when there are multiple methods (paths) to accomplish system tasks. If there are no choices then there is no autonomy to adjust.
- The human (or agent) that is adjusting autonomy must have knowledge of the capabilities of the other agent(s) and be able to recognize success and failure.
- The protocol for changing responsibility (or the level of autonomy) must be clear and must support both requesting a change in autonomy and accepting a change in autonomy.

4 Conclusions

We have developed the following list of questions that must be asked (and answered) when developing a human-centered autonomous system.

- What tasks can be done only by humans? Only by automation? By both?
- Who can set the level of autonomy for a task? Can the level of autonomy change at any time or only under certain circumstances?
- What are the timing issues with respect to a change in autonomy?
- Can an autonomy setting at one level of a hierarchical task be applied to all descendants?
- What are the possible autonomy level transitions? What transitions are not permitted?
- Is information necessary to control the system available to the user or to other agents?
- Are there multiple ways to accomplish the same task? Are they selectable by the user? By a planner?
- What parts of the system are commandable from the outside?
- How is success and failure of other agents recognized?

References

[Barber *et al.*, 1999] Suzanne Barber, Anul Goel, and Cheryl Martin. The motivation for dynamic adaptive autonomy in agent-based systems. In *Proceedings of the First Asia-Pacific Conference on Intelligent Agent Technology (IAT '99*, 1999.

[Bonasso *et al.*, 1997] R. Peter Bonasso, David Kortenkamp, and Troy Whitney. Using a robot control architecture to automate space shuttle operations. In *Proceedings of the 1997 Innovative Applications of Artificial Intelligence Conference*, 1997.

[Dorais *et al.*, 1998] Gregory Dorais, R. Peter Bonasso, David Kortenkamp, Barney Pell, and Debra Schreckenghost. Adjustable autonomy for human-centered autonomous systems on mars. In *Proceedings of the Mars Society Conference*, 1998.

[Kortenkamp *et al.*, 2000] David Kortenkamp, Debra Keirn-Schreckenghost, and R. Peter Bonasso. Adjustable control autonomy for manned space flight. In *Proceedings of the IEEE Aerospace Conference*, 2000.

[Musliner and Krebsbach, 1999] David Musliner and Kurt Krebsbach. Adjustable autonomy in procedure control for refineries. In *Working Notes of the AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.

A Cognitive Model of Situated Autonomy

Henry Hexmoor

Computer Science & Computer Engineering Department
Engineering Hall, Room 313
Fayetteville, AR 72701
hexmoor@uark.edu

Abstract. We introduce situated autonomy and present it as part of the process of action selection. We then discuss the cognitive ingredients of situated autonomy and derive a degree of situated autonomy.

1 Situated Autonomy and Action Selection

Autonomous agents have been defined to be agents with self-generated goals using the agent's motivations [4]. Such agents perform action selection, which is the function of selecting the most relevant and meaningful action [9], entirely for selfish reasons [7]. We believe circumstances of everyday agents provide opportunities for reasoning about relative levels of autonomy. Instead of being autonomous in the general sense, we will focus on the notion of autonomy in the context of a situation and in a team with other agents. We will consider agents able to perform autonomy considerations very fast while they are in the situation. Imagine in a game of basketball, the agent who is in the midst of running anticipates a block and reflects about whether to pass the ball or to run with it. Here, autonomy is a split-second situated assessment.[1] The player considers choosing each of the behaviors "pass the ball" and "run with the ball." The agent's considerations of autonomy involve the higher-level goals of scoring or driving the ball into the opponent zone. The agent decides between its orientation to "pass the ball" which means sharing its autonomy toward scoring/driving with another player or its orientation to "run with the ball" which means self-autonomy. Situatedness is to consider situations in the environment as integral component of the agent's process of deliberation or reactive response generation. Situation is a state of the world as it pertains to a problem. **We define situated autonomy as an agent's stance, as well as the cognitive function of forming the stance, toward assignment of the performer of a goal at a particular moment when facing a particular situation.** Assumption of individual versus social rationality affects the cognitive function. At a coarse level the agent's orientation toward the goal will be whether to abandon it or to decide its overall position toward the goal: to make it an entirely personal goal, to make a goal for another agent, to consider the goal a collaborative effort, or to consider an inclination for the goal that

[1] Assessment of autonomy is either a deliberative process or an automatic association of a stance that might be a historic stance or based on the agent's personality.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 325–334, 2001.

is less than totally self-directed. Here, we are not concerned about responsibility for a goal, which is the amount of effort or commitment an agent is willing to spend on seeing to its accomplishment. At a finer level the agent's stance will go beyond an overall position to include a degree of situated autonomy. In this paper, the degree of autonomy will be a measure of the agent's deliberateness over its autonomy decision. Two useful measures of autonomy beyond the scope of this paper are (1) degree of relative dependence on environmental factors such as other agents, and (2) degree of control (or influence) an agent has over a goal. Generally determining degree of autonomy is more time-consuming than determining an overall position. In our discussion of situated autonomy we will not care whether the goals are internally generated or externally imposed.

Action selection generates an action in response to a new situation. An important step in action selection is choosing among possible plans or possible primitive actions. We propose that situated autonomy can be used in this decision. Given that an agent may have several alternative plans and actions to achieve a goal with each alternative appropriate at different commitment level, an agent's situated autonomy can be used as the context for arbitration.

For a cognitive agent, action selection is affected by the frequency of encountering new situations. We describe an agent's assessment of its situated autonomy that is also affected at varying situation frequencies.

At the highest frequency, the agent may produce reflex-like actions. Such agents have no time to account for their situated autonomy. At a relatively fast frequency, the agent may produce reactive actions with minimal time to account for situated autonomy. Such situated autonomy assessment will consider pre-disposition toward the goal. Pre-disposition here is taken as "an evaluative tendency that is expressed by evaluating a particular entity with some degree of favor or disfavor" [3, p. 693]. An agent's pre-disposition toward a goal is based on semi-permanent beliefs and goals about enabling factors for the goal. Our understanding of pre-disposition is a cognitive function that operates on the agent's weak beliefs and unmotivated goals. **Enabling factors for a goal are subset of the elements of the situation that are either necessary or facilitating conditions for attempting the goal.** We consider individual, social, and physical enablers with different origins: (a) entirely endogenous, (b) exogenous and social, and (c) exogenous and physical in nature. The individual enablers are the agent's competencies. The social enablers are the social influences and motivators. The physical enablers are the physical conditions, physical resources, and physical tools. We will further discuss these types of enablers in a later part of this paper. An agent may have a model of other agents as well as its own. We will use prefixes "Other-" and "Self-" to distinguish between an agent's model of other agent's enablers and its own. For instance, Other-Social-Enabler will denote an agent's model of another agent's social enablers and Self-Physical-Enabler will denote an agent's model of its own physical enablers.

At slower frequencies, the agent will have more time to assess the quality and quantity of the enabling factors. Situated autonomy at that level will be based on dispositions toward the goal. Perkins, Jay and Tishman [8], define dispositions as "people's tendencies to put their capabilities into action" (p. 75). At yet slower frequencies, the agent will have time to consider going beyond dispositions derived

from enabling factors and include motivations. Human motivations are a collection of psychogenic needs, which guides behavior [6]. At the slowest frequency, the agent may consider long-term ramifications of its options in order to produce optimal actions. In this paper we consider goal-oriented social agents in domains with relatively fast situation frequency. Such agents may have limited time to consider situated autonomy. They may just have enough time for assessing an overall position.

Consider action selection as a linear process where somehow the agent's action selection has settled on a goal. The next step and the focus of this paper are the agent's reflections on its autonomy with respect to the goal at hand. Finally, the agent uses the results of its introspection and renders a decision about action(s).

Figure 1 summarizes the types of action generated by action selection that is at different frequencies of Situations. Reflex actions are generated without much deliberation for situated autonomy. Other than reflex actions, situated autonomy consideration for actions generated to the right are coarser than for action to the left.

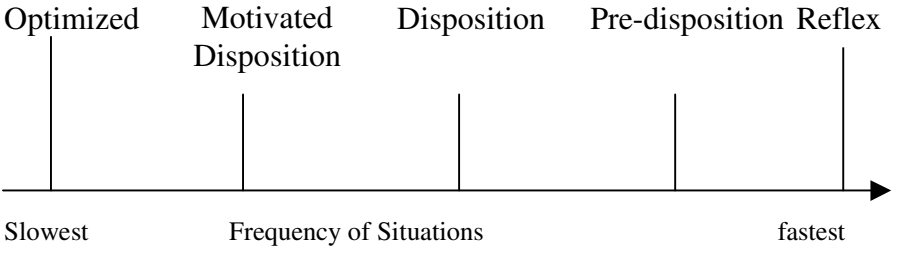


Fig. 1. Action selection at different frequencies of Situations

Given a goal, the agent's assessment of situated autonomy is a cognitive process that is comprised of several stages, Figure 2. The process begins by the agent's consideration of pre-dispositions only. If the agent has a habit of disfavoring the goal, it will decide to ignore it for no other reason other than its habit and considers itself Not-autonomous with respect to that goal. The agent who habitually favors the goal and favors itself to be the executor of the goal will consider itself to be Self-autonomous. The agent who habitually favors the goal and favors itself not to be the executor of the goal will lean toward delegation and will consider itself to be Del-autonomous.

The agent with some more time may consider the goal further and form dispositions toward it. If the agent perceives the goal to be impossible, the agent forms a mental state of Not-autonomous. If the agent perceives that the goal is doable either exclusively by the agent alone or by delegation, it will stop further considerations of situated autonomy. If such an agent solely using its dispositions considers itself to be the executor of the goal, it will consider itself to be Self-autonomous. When we say an agent is autonomous with respect to a goal, we may mean one of two things about its disposition toward the goal. We may mean the agent is self-reliant in the sense that it is not affected by any exogenous sources such as social or physical. Alternatively, we

may mean the agent can bring about the desired effect given its access to its exogenous sources such as other agents or resources or it can do it itself. If it considers other agents to be executors of the goal, it will consider itself to be Del-autonomous. If the goal is deemed clearly appropriate for delegation due to the agent's inability to perform the goal itself, the agent is considered Del-autonomous and subsequently a communicative act will be generated.

An agent who has formed a disposition toward its goal that has not resulted in determination of either the agent or others being the exclusive executors may further use its motivations. Moreover, motivations can modify a decision that is previously determined based on disposition. We will consider motives to be captured by a policy that produces a preference to favor/disfavor the goal as well as the executor of the goal. If a goal is deemed inappropriate due to the agent's motivation policy, the initial commitment is revised and the agent is considered to be Not Autonomous with respect to that goal. If a goal is deemed feasible for self-performance due to the agent's disposition and additionally appropriate due to the agent's motivation, the agent is considered to Self-autonomous and the goal might be sent to the motoric system.

If the agent has not determined exclusive execution, the agent is slated to perform the goal with other agents and its autonomy is classed as Semi-autonomous or Shared-autonomous. Shared-autonomous implies getting assistance from another agent or reliance on some environmental elements such as tools or resources, or offering help to some other agent who will be the primary executioner of the goal. Shared autonomy implies more demand for the agent than semi-autonomy. With semi-autonomy, the agent knows it is dependent on outside sources to perform the goal. With shared autonomy the agent knows furthermore that there are one or more agents that complement its autonomy. An example of shared autonomy is shown between a human air traffic controller and a collision-detection agent [5].

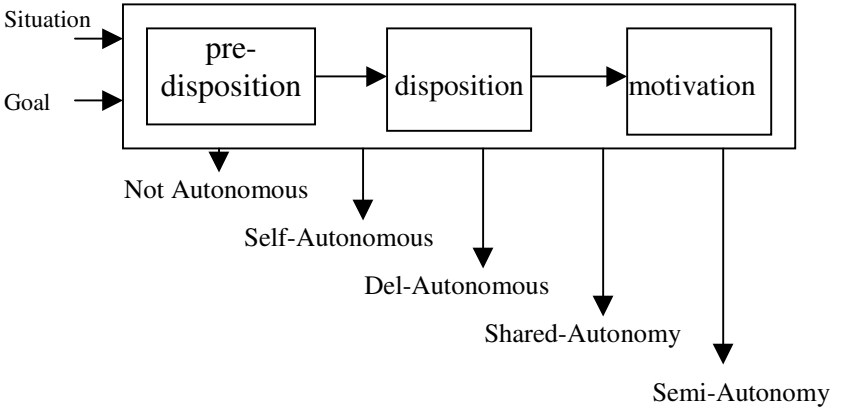


Fig. 2. Situated Autonomy as part of the process of action selection

An agent may perceive the goal to be shared by a team. Such an agent will be motivated to help the team and may override its determination of shared-autonomy or semi-autonomy based on its dispositions in favor of self-autonomy.

2 A BDI Model of Autonomy: Necessary Cognitive Components for Situated Autonomy

In this section we will provide a predicate calculus account of autonomy. We do not claim our characterization to be complete but we believe it covers the major human-intuitive aspects of situated autonomy.

We refine our earlier use of pre-disposition with a combination of a weak belief in the enabling factors, and an unmotivated goal. An agent's belief is weak B^w when it is not carefully formed and it is premature. An agent may have little time to form its belief or the agent senses uncertainties in the object of the belief. An agent may form sketchy beliefs solely based on prior experience and otherwise not have substantiated support for holding the belief. In contrast to B^w, we define B^s as a strong belief when the agent has inductive or deductive reasons for holding the belief. This style of treating belief differs from explicit/declarative evaluation. Instead they are closer to that in [1].

A goal is unmotivated G^u when the process of adopting the goal has suffered from weaknesses in related prior intentions or beliefs. G^m is defined as a motivated goal when the agent has (a) inductive or deductive reasons for adopting the goal including goals that the agent shares with a team [2], and (b) a wish for expecting it. Otherwise, an agent's goal is unmotivated. An unmotivated goal is adopted out of habit but fails to have justification and the agent's wish.

We list a few factors that contribute to forming a motivated self-directed goal. We will not use any notation for brevity. In the following, a goal is owned by the agent if another agent did not suggest the goal. I.e., the agent has endogenous reasons for the goal.

- x *owns the goal* to bring about that goal.
- x is *not the owner of the goal* but the goal is owned by the team and x perceives a great deal of personal *commitment and responsibility* for that goal.
- x perceives itself to be the *only agent who can do the goal*.

One or more of the following mental states may support motivated goal that is other - directed:

- x *owns the goal* to bring about that goal
- x is *not the owner of the goal* but is owned by the team and x perceives a great deal of *commitment and responsibility* for the delegation of that goal.
- x *believes* it does not have individual enablers.

We are now ready to define Self-autonomy in terms of pre-disposition.

Definition 2.1 Self-autonomous^p
Agent x is self-autonomous based on its pre-dispositions with respect to goal g in situation s iff x can perform an act to bring about g with permission but however, it

has a weak belief in situation s about Self-Enablers and it has g as an unmotivated goal in situation s.

(Self-autonomous^{p} x s g) ≡ ∃ α (Agts α x) ∧ (Achvs α g) ∧ (Per x α) ∧ (B^{w} x s Self-Enablers) ∧ (G^{u} x s g)

The predicate "Per" stands for the deontic notion of permission for individuals (See [10] for typical usage of permissions) and "G-Per" (used later in this paper) stands for group permission. Note that individuals and groups typically do have the same permissions and that group permissions cannot be reduced to the permissions of the individuals forming the group. The notations "Agts" and "Achvs" stands for "agent of" and " Achieves" respectively and are similar to their usage in [11].

We consider disposition to be a combination of an unmotivated goal G^{u} and a strong belief B^{s} in the enabling factors. Using disposition, we define Self-autonomy again, Self-autonomous$^{D.}$

Definition 2.2 Self-autonomous^{D}

Agent x is self-autonomous based on its dispositions with respect to goal g in situation s iff x can perform an act to bring about g with permission, it has a strong belief in situation s about Self-Enablers but however, it has g as an unmotivated goal in situation s.

(Self-autonomous^{D} x s g) ≡ ∃ α (Agts α x) ∧ (Achvs α g) ∧ (Per x α) ∧ (B^{S} x s Self-Enablers) ∧ (G^{u} x s g)

We define motivated disposition as a combination of a goal that is motivated G^{m} with motivation, and a strong belief B^{s} in the enabling factors. We define Self-autonomy this time based on motivated disposition, Self-autonomous$^{MD.}$

Definition 2.3 Self-autonomous^{MD}

Agent x is self-autonomous based on its motivated dispositions with respect to goal g in situation s iff x can perform an act to bring about g with permission, it has a strong belief in situation s about Self-Enablers, and it has g as a motivated goal in situation s.

(Self-autonomous^{MD} x s g) ≡ ∃ α (Agts α x) ∧ (Achvs α g) ∧ (Per x α) ∧ (B^{S} x s Self-Enablers) ∧ (G^{m} x s g)

Next, we define Del-autonomy in terms of pre-disposition.

Definition 2.4 Del-autonomous^{p}

Agent x is del-autonomous based on its pre-dispositions with respect to goal g in situation s iff there is an agent y (other than x) that can perform an act to bring about g with permission but however, x has a weak belief in situation s about Other-Enablers and it has g as an unmotivated goal in situation s.

(Del-autonomous^{p} x s g) ≡ ∃ αy (Agts α y) ∧ (Achvs α g) ∧ (Per y α) ∧ (B^{w} x s Other-Enablers) ∧ (G^{u} x s g)

Definition 2.5 Del-autonomous^{D}
Agent x is del-autonomous based on its dispositions with respect to goal g in situation s iff there is an agent y (other than x) that can perform an act to bring about g with permission, x has a strong belief in situation s about Other-Enablers but however, it has g as an unmotivated goal in situation s.
(Del-autonomous^{D} x s g) $\equiv \exists\ \alpha y$ (Agts α y) $\wedge$ (Achvs α g) $\wedge$ (Per y α) $\wedge$ (B^{S} x s Other-Enablers) $\wedge$ (G^{u} x s g)

Definition 2.6 Del-autonomous^{MD}
Agent x is del-autonomous based on its motivated dispositions with respect to goal g in situation s iff there is an agent y (other than x) that can perform an act to bring about g with permission, x has a strong belief in situation s about Other-Enablers, and it has g as a motivated goal in situation s.
(Del-autonomous^{MD} x s g) $\equiv \exists\ \alpha y$ (Agts α y) $\wedge$ (Achvs α g) $\wedge$ (Per y α) $\wedge$ (B^{S} x s Other-Enablers) $\wedge$ (G^{m} x s g)

We believe Shared_autonomy requires more than a pre-disposition.

Definition 2.7 Shared-autonomous^{D}
Agent x is shared-autonomous based on its dispositions with respect to goal g in situation s iff x is a part of a group of agents t, where t can perform an act to bring about g with permission, x has a strong belief in situation s about Group-Enablers but however, the group has g as an unmotivated goal in situation s.
(Shared-autonomous^{D} x s g) $\equiv \exists\ \alpha t$ (x $\in$ t) $\wedge$ (Agts α t) $\wedge$ (Achvs α g) $\wedge$ (G-Per t α) $\wedge$ (B^{S} x s Group-Enablers) $\wedge$ (G^{u} t s g)

Definition 2.8 Shared-autonomous^{MD}
Agent x is shared-autonomous based on its motivated dispositions with respect to goal g in situation s iff x is a part of a group of agents t, where t can perform an act to bring about g with permission, x has a strong belief in situation s about Group-Enablers, and the group has g as a motivated goal in situation s.
(Shared-autonomous^{D} x s g) $\equiv \exists\ \alpha t$ (x $\in$ t) $\wedge$ (Agts α t) $\wedge$ (Achvs α g) $\wedge$ (G-Per t α) $\wedge$ (B^{S} x s Group-Enablers) $\wedge$ (G^{u} t s g)

We believe Semi_autonomy requires more than a pre-disposition but typically does not change with motivation.

Definition 2.8 Semi-autonomous^{D}
Agent x is shared-autonomous based on its dispositions with respect to goal g in situation s iff x is a part of a group of agents t, where t can perform an act to bring about g with permission, However, x has a weak belief in situation s about the Group-Enablers, and the group has g as an unmotivated goal in situation s.
(Semi-autonomous^{D} x s g) $\equiv \exists\ \alpha t$ (x $\in$ t) $\wedge$ (Agts α t) $\wedge$ (Achvs α g) $\wedge$ (G-Per t α) $\wedge$ (B^{w} x s Group-Enablers) $\wedge$ (G^{u} t s g)

Finally, an agent is not autonomous if no other form of Autonomy holds.

Definition 2.8 Not-autonomous
Agent x is not-autonomous with respect to goal g in situation s iff there are no acts that x has both permission to perform and perform to achieve g.
(Semi-autonomous^{D} x s g) ≡ $\neg\exists$ α (Agts α t) $\wedge$ (Achvs α g) $\wedge$ (Per x α)

Not-autonomous necessarily presupposes that x does not have any Self-, Del-, Shared-, or Semi-autonomy.

Our notations so far can only help with a coarse reasoning about situated autonomy for the agent. We have identified four categories of situated autonomy. Self-Autonomous gives the agent the most choices to complete its action selection. An agent's action selection must decide on method of delegation with Del-Autonomous. Semi-autonomous is the least clear and the agent's action selection may use other consideration for action selection. With Shared-autonomous, the action selection must consider the other agents sharing the autonomy over the goal for an action. If the agent is Not_autonomous, its action selection can terminate and the agent will not rationally perform that goal.

3 Degree of Situated Autonomy

In the previous section we presented several categories of situated autonomy. Stances based on pre-disposition are weaker than the ones based on disposition. Stances based on disposition are weaker than the ones that include motivation. We propose that the strength of an agent's stance is the basis for its degree. For example, an agent's stance Del-autonomous^{MD} is stronger and has a higher degree than the agent's stance Self-autonomous^{D}. The degree of situated autonomy within each stance is a function of (a) the agent's subjective certainty in the agent's beliefs about enabler ingredients, (b) the perception of the absolute quantity of the enabler components that gives the agent a sense of liberty, and (c) the strength of positive mental states due to situation at-hand and lack of strength in the negative mental states due to the situation at-hand. First, we will briefly discuss the liberties an agent senses with regard to three main enabling quantities. This will be used in defining degrees of Self-, and Del-Autonomy.

The physical enablers. For example, the road condition and mechanical integrity of a vehicle can be considered environmental elements for a driving goal. An agent's perception of the quantitative condition of the road and the car combined with its certainty about its own beliefs forms the overall physical enabler component of its autonomy. Colloquially, we may think of the agent's Faith or Trust in the physical enablers or Faith/Trust in its perception or its beliefs. To the extent the agent experiences freedom from concerns about physical enablers, the agent has physical liberties.

The social enablers. Other agents may provide positive or negative social influences. Other agents may facilitate achievement of the goal or detract from it. The perception of the quantity of social elements affecting the goal as well as the agent's belief in

such elements makes up the contribution of the social enablers in determining a degree of situated autonomy. For example, a friend who is familiar with the driving directions for a driving goal can make the goal easier. The agent's perception of the degree of its dependence on such a friend for help contributes to the agent's degree of social enablers. To the extent the agent is invested in the favorable social elements and is ready to guard against unfavorable elements, the agent has social liberties.

The individual enablers. The agent's perception of the degree of its competencies as well as the certainty of beliefs in them makes up the contribution of individual enablers in determining a degree of situated autonomy. To the extent the agent is self-confident about the goal the agent has individual liberties.

The necessary conditions for Self-Autonomy require an agent to sense high levels of Individual, Social, and Physical liberties with respect to the goal. Whether such an agent has a high degree of Self-autonomy depends on the strength of its mental states due to the situation at-hand.

The necessary condition for Del-Autonomy requires an agent to sense a high level of Social liberty of other agents with respect to the goal. Whether such an agent has a high degree of Del-autonomy further depends on (a) the strength in one or more of the following positive mental states:

- x *owns the goal* to bring about the goal

and (b) lack of strength in the following negative mental state:

- x *believes* it does not have individual enablers.

We treat the degree of an agent's Shared-Autonomy to be its relative autonomy with respect to other agents for a given goal in a given situation. Since other agent's autonomy and the situation affect Shared-Autonomy, it is more dynamic than Self-, and Del-Autonomy. The necessary condition for Shared-Autonomy requires an agent to sense that there are other agents who share the goal and complement its autonomy with respect to the goal. Whether such an agent has a high degree of Shared-autonomy depends on (a) the strength in the following positive mental state:

- x *owns the goal* to bring about the goal.

and (b) lack of strength in one or more of the following negative mental states:

- x *believes* it does not have individual enablers,

and (c) the extent to which,

- relative to other agents with whom x shares the goal, x perceives a relatively large level of personal power and control over the goal.

For an agent to sense a high degree when it is Semi-Autonomous depends on the intensity of the agent's mental state that "x *owns the goal* to bring about that goal" combined with the agent's mental state that "x *believes* it does not have adequate individual enablers."

Finally, an agent that is Not Autonomous has the least degree of situated autonomy.

4 Conclusion

We have discussed the idea that situated autonomy is at the core of an agent's action selection. It is responsible for the agent's mental state about how to relate to a goal in the context of relevant enabling factors in an environment that includes other agents. We have presented central cognitive ingredients that constitute notions of autonomy of self, delegation, and sharing. Degree of autonomy as a measure of the agent's deliberativeness of its decision is then presented. It is argued to be dependent on the qualities of enabling factors and the strength of the agent's beliefs in them.

We plan to refine these notions and to implement agents that exhibit dynamic changes in their autonomy. Previously we presented some quantified results of different levels of autonomy [5]. Our future plans include extending our empirical studies with implementations of situated autonomy.

References

1. Castelfranchi C., Falcone R., (2000), Trust and Control: A Dialectic Link, *Applied Artificial Intelligence Journal*, Special Issue on "Trust in Agents" Part 1, Castelfranchi C., Falcone R., Firozabadi B., Tan Y. (Editors), Taylor and Francis 14 (8).
2. Cohen, P. R., and Levesque, H. J. (1990). Persistence, intention, and commitment, Cambridge, MA: MIT Press,1990, Cambridge, MA: MIT Press.
3. Eagly, A.H. (1992). Uneven Progress: Social Psychology and the Study of Attitudes. Journal of Personality and Social Psychology 63(5): 693-710.
4. d'Inverno M. and Luck M., (1996). Understanding Autonomous Interaction, in ECAI '96: Proceedings of the 12th European Conference on Artificial Intelligence, W. Wahlster (ed.), 529-533, John Wiley and Sons.
5. Hexmoor, H.. (2000). Case Studies of Autonomy, In proceedings of *FLAIRS 2000*, J. Etherege and B. Manaris (eds), p. 246- 249, Orlando, FL.
6. McAdams. D.P. (1989). The Development of a Narrative Identity. In D.M. Buss and N. Cantor (Eds.). Personality Psychology: Recent Trends and Emerging Directions. New York: Springer-Verlag. pp. 160-176.
7. Luck M. and d'Inverno M., (1995). A Formal Framework for Agency and Autonomy, in Proceedings of the First International Conference on Multi-Agent Systems, 254-260, AAAI Press / MIT Press.
8. Perkins, D., Jay, E., and Tishman, S. (1993). New Conceptions of Thinking: From Ontology to Education. Educational Psychologist 28(1): 67-85.
9. Pirjanian, P., (2000). Multiple Objective Behavior-Based Control, *Robotics and Autonomous Systems*, Volume 31, Issue 1-2, 30 April 2000, pp. 53-60.
10. Wieringa R. J. and Ch. Meyer, J.-J. (1993). Applications of Deontic Logic in Computer Science: A concise view, In *Deontic Logic in Computer Science,* 17 – 40, John Wiley & Sons, Chichester, England.
11. Wooldridge, M. (2000). *Reasoning about Rational Agents*, The MIT Press, Boston.

Designing an Architecture for Adjustably Autonomous Robot Teams

David Kortenkamp

NASA Johnson Space Center – ER2
Houston, TX 77058, USA
kortenkamp@jsc.nasa.gov

1 Introduction

We are beginning a project to develop fundamental capabilities that enable multiple, distributed, heterogeneous robots to coordinate in achieving tasks that cannot be accomplished by the robots individually. The basic concept is to enable the individual robots to act fairly independently of one another, while still allowing for tight, precise coordination when necessary. The individual robots will be highly autonomous, yet will be able to synchronize their behaviors, negotiate with one another to perform tasks, and "advertise" their capabilities. This architectural approach differs from most other work in multi-robot systems, in which the robots are either loosely coupled agents, with little or no explicit coordination [1,4,5], or else are tightly coordinated by a highly centralized planning/execution system [3]. Our proposed architecture will support the ability of robots to react to changing and/or previously unknown conditions by replanning and negotiating with one another if the new plans conflict with previously planned-upon cooperative behaviors. The resulting capability will make it possible for teams of robots to undertake complex coordinated tasks, such as assembling large structures, that are beyond the capabilities of any one of the robots individually. Emphasis will be placed on the reliability of the worksystem to monitor and deal with unexpected situations, and flexibility to dynamically reconfigure as situations change and/or new robots join the team.

A main technical challenge of the project is to develop an architectural framework that permits a high degree of autonomy for each individual robot, while providing a coordination structure that enables the group to act as a unified team. Our approach is to extend current state-of-the-art hierarchical, layered robot architectures being developed at NASA JSC (3T) [2] and CMU (TCA) [6] to support distributed, coordinated operations. Our proposed architecture is highly compatible with these single-agent robot architectures, and will extend them to enable multiple robots to handle complex tasks that require a fair degree of coordination and autonomy.

As second technical challenge is to use distributed techniques to provide coordinated control of complex, coupled dynamic systems. For example, a mobile manipulator may have many degrees of freedom and controlling them all from a single controller would be complicated and computationally expensive. However,

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 335–338, 2001.

by breaking the complicated control problem into several simpler control problems and then having the simpler control problems coordinate and cooperate with each other to achieve a task we can reduce complexity and computational requirements. This approach will require the architectural support described in the previous paragraph.

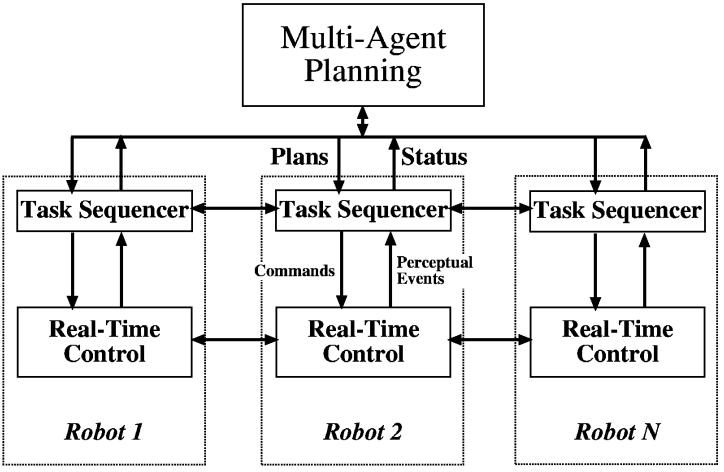


Fig. 1. A distributed, multi-layered robot architecture.

2 Approach

Our basic approach to multi-robot architectures is to distribute the behavior and sequencing layers of the three-tiered architectural approach, while maintaining a centralized planner (Figure 1). The centralized planner sends high-level, abstract plans to individual robots, where the plans include goals to be achieved and temporal constraints between goals. The task sequencer then decomposes the goals into subtasks, and issues commands to the behavior layer. The behavior layer executes the commands, sending data back to the sequencer so that it can monitor task execution. Occasionally, status information is sent back to the planner, especially when the robots encounter problems they cannot solve.

3 Preliminary Work

Our project testbed will be a multi-robot construction scenario (see Figure 2). Our most significant achievement to date is the development of distributed visual servoing, using a roving eye and fixed manipulator (see Figure 3). The servoing system uses a pair of color stereo cameras to provide a 6DOF pose that is the

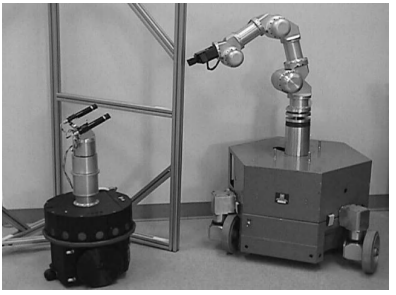


Fig. 2. Mobile manipulator and roving camera performing construction task.

Fig. 3. Fixed manipulator and roving camera perform servoing. Colored fiducials are used for vision.

difference between two colored fiducials. This difference is used to drive the arm. The servoing continues until the target fiducial reaches the desired pose. The roving eye drives around the workspace to keep the targets in sight and centered in the image, and it moves back and forth to ensure that the targets fill most of the camera field of view. The roving eye and arm are completely distributed and autonomous. They use a distributed version of 3T's Skill Manager to coordinate activities. This work was performed jointly by NASA JSC and CMU graduate student David Hershberger, who worked in the NASA JSC labs over the summer. This use of a roving eye, completely separated from the arm it is guiding, is a novel approach to visual servoing and has many applications in construction and manufacturing. We are currently performing experiments to measure quantitatively the precision obtained by this approach.

4 Adjustable Autonomy Issues

The work we discuss in this paper has not yet directly addressed adjustable autonomy. This section introduces some adjustable autonomy issues and possible solutions.

- **Teaming**: Our approach will allow robots to create dynamic and ad hoc teams to accomplish tasks. Sometimes this will require several robots to become "subservient" to other robots while members of a team. For example, if two robots are moving a long beam, one of the robots may be designated the lead robot and it will pass commands directly to the other robot, which will execute them with limited autonomy. During the course of performing many different tasks, robots may sometimes be in the leader role and sometimes in the follower role. So, they will need to adjust their autonomy level to reflect their role in the team.
- **Operator interaction**: The goal of our research is to develop remote colonies of robots on planetary surfaces. Because of limited bandwidth communication, operator interaction with the robots will be limited. However,

there may be times when direct operator control of an individual robot or a team of robots is required. Traded control options will need to be built into our architecture.
- **Human/robot teams**: We also want to allow for the possibility that human crew members could be working along side robotic crew members in construction tasks. While this will require significant human/robot interaction advances (for example in natural language and vision), the adjustable autonomy aspects should not be much different than in the first bullet of this section.

Acknowledgements. The development of this proposed architecture has been a collaborative process with Reid Simmons of Carnegie Mellon University and Robert R. Burridge of NASA Johnson Space Center. CMU graduate student David Hershberger implemented the system described in Section 3 while working at NASA Johnson Space Center.

References

[1] Tucker Balch and Ron Arkin. Behavior-based formation control for multiagent robot teams. *IEEE Transactions on Robotics and Automation*, 14(6), 1998.
[2] R. Peter Bonasso, R. J. Firby, E. Gat, David Kortenkamp, David P. Miller, and Marc Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1), 1997.
[3] O. Khatib. Force strategies for cooperative tasks in multiple mobile manipulation systems. In *Proceedings of the International Symposium of Robotics Research*, 1995.
[4] Maja J. Mataric. Using communication to reduce locality in distributed multiagent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(2):357–369, 1998.
[5] Lynne Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 1998.
[6] Reid Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1), 1994.

Making Adjustable Autonomy Easier with Teamwork

Paul Scerri and Nancy E. Reed

Department of Computer and Information Science
Linköping University, SE-581 83 Linköping, Sweden
pausc@ida.liu.se | nanre@ida.liu.se

Abstract. Adjustable Autonomy (AA) is the name given to a variety of approaches to the task of giving outside entities the ability to change the level of autonomy of agents in an autonomous system. The idea presented in this paper is to leverage the flexibility and robustness of agent teams to develop flexible and safe AA mechanisms. We argue that the fundamental properties of teamwork are very useful for providing the robustness to change and flexibility that an implementation of AA requires. We present the EASE system which leverages teamwork as the basis for a powerful and robust AA system for intelligent actors in interactive simulation environments.

1 Introduction

Despite our best efforts, so called "intelligent" actors sometimes don't do precisely what we expect or intend them to do. Coffee lounges are full of anecdotes of actor designers watching in horror as at a critical moment in an important demonstration an "intelligent" actor does something extremely stupid. Giving designers some control over actors at runtime will increase their utility - as well as reducing stress [Blumberg and Galyean, 1995]. In actors deployed over long periods of time, with safety critical missions, perhaps on spacecraft, it may be essential that the actor's behavior can be modified without restarting it [Dorais, 1998, Schooley *et al.*, 1993, Kortenkamp *et al.*, 2000].

Adjustable Autonomy (AA) is the name given to a variety of approaches to achieving the goal of allowing the parts of a system to change levels of autonomy [Kortenkamp *et al.*, 1999b, Musliner and Krebsbach, 1999, Reed, 2000, Hexmoor, 1999]. In this paper we are primarily concerned with mechanisms giving humans some control over autonomous actors in simulation environments. This means giving the user the ability to change aspects of the agent's behavior at runtime. The behavior should be able to be changed at all levels of abstraction and at any time during the simulation. The more flexibility the AA gives to a user the better the AA serves its purpose.

However, AA is far from trivial to build into an actor. Two general approaches are immediately obvious. Firstly, at design time the designers could work out the different controls the user will need at runtime and build them into the actor.

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 339–352, 2001.

This approach is taken in systems like HCSM [Cremer *et al.*, 1995] and Improv [Perlin and Goldberg, 1996]. An advantage of this approach is that the designers can ensure a priori that the actions the user takes at runtime will be safe and sensible. A limitation of the approach is that the designer needs to correctly identify all the actions the user might want to take, a priori. This is problematic as it seems that one of the important uses of AA is for graceful handling of situations that were *unexpected* at design time.

A second general approach to AA is to augment the agents with mechanisms that allow users to flexibly manipulate the internals of the actor. This is the approach taken by [Kortenkamp *et al.*, 1999b] and [Blumberg and Galyean, 1995] among others. Such mechanisms, if done well, can provide the user with an almost unlimited range of options, allowing them to make whatever behavior changes are required at runtime. One downside of such approaches is that, because the designer gives up some control over the internal running of the actor, it becomes difficult to guarantee that the actor's behavior will be reasonable, safe or graceful. Even carefully designed specifications can easily fall apart when a user starts making ad hoc changes.

The general approach of providing mechanisms to allow a user to manipulate the internal structure of an actor is appealing because it allows great flexibility. However, the promise can only be realized if the actor can handle fairly ad hoc changes to its structure robustly.

Most agent systems are not designed to allow ad hoc changes to their composition. However, there is one type of system that *is* specifically designed to be flexible and robust to changes to its components – multi-agent systems (MASs) [Tambe, 1996]. When the agents work together as a *team* towards common goals they can be especially robust and flexible. Part of the reason for the flexibility is that interactions between the components (i.e., agents) of the system are at an abstract level. The relationships between agents are explicitly represented and understood so when one part changes (or breaks) the rest of the team can reconfigure smoothly around the changed or broken agent.

We leverage the flexibility and robustness of agent teams to develop flexible and safe AA mechanisms. The agent teamwork implements the same type of guarantees, e.g., safe and reasonable, that a designer might implement if developing hard-coded AA controls - but in a way that is flexible at runtime. In particular a single agent is composed of a complete MAS. (For clarity we refer to the overall agent (composed of a MAS) as an actor, and the components of the actor (team members in the MAS) as agents.) The AA mechanism gives the user ad hoc control over the MAS. The teamwork between the agents in the MAS helps make the overall behavior "graceful" and safe.

A prototype of this idea has been implemented. The real-time control prototype, nicknamed "The Boss" is a sub-system of an actor specification system called EASE (End-user Actor Specification Environment) [Scerri and Reed, 2000b]. EASE is used to develop actors which are internally controlled by a multi-agent system (MAS). The Boss works by adding agents to, removing agents from or suspending agents in the multi-agent system which

in turn dictates the behavior of the actor. These facilities give the user a large amount of control over the actor at runtime. Importantly the user is not required to take back complete control of the actor, only manipulate those aspects of the behavior that they want or need to change. Because the addition or removal of agents is done in accordance with the team conventions and within the team model, the overall behavior of the actor usually continues to be reasonable while the user makes their desired changes. That is, the normal teamwork mechanisms serve to provide the robustness required in the face of these ad hoc changes.

2 EASE

EASE is an integrated graphical development system for specifying and running intelligent actors for simulation environments. Specific design tools support all aspects of development from information processing specification through to mission specification. For more details on EASE see [Scerri and Reed, 2000b], in this paper we describe only enough of EASE to support the AA.

An actor created with EASE has a complete multi-agent system that performs the task of action selection. The basic idea is conceptually similar to the "Society of Mind" ideas of Minsky [Minsky, 1988] and that of *behavior-based architectures* [Mataric, 1994]. Each agent in the system is responsible for a specific aspect of the actor's overall behavior. For example, in a simulated pilot actor there might be one agent for avoiding the ground and another for heading towards some way point. The teamwork is fairly simple – the agents form *contracts* with each other in a hierarchical structure. Contracts correspond closely to the idea of delegation [Falcone and Castelfranchi, 1999]. The contracted agent is assigned a specific aspect of the contractor's task. For example, a contractee of a *Safety* contractor might be responsible for avoiding a particular aircraft. The contracts, including the specific agent to be contracted, are determined at design time by the designer. Coupled with the notification of success or failure and negotiation amongst agents, the contracts form the core of the teamwork mechanisms.

Agents at the bottom of hierarchies in the MAS are directly involved in negotiations about the actor's actions. Each agent involved in the negotiation has a function which it uses to decide its *satisfaction* with different proposed actions for the actor. A *factory* administers the negotiation. A factory continuously selects values from the domain of possible actions and suggests the values to the agents in the negotiation. The agents calculate their satisfaction value for the suggested action and return it to the factory. The factory compares the responses of the agents to the new suggestion and their responses to the previous most favored suggestion. The factory keeps a record of the negotiation and periodically executes the favored action. The factory's algorithm for choosing between suggestions is an anytime version of a fuzzy behavior fusion algorithm [Pirjanian and Mataric, 1999]. A snapshot of the Negotiation Viewer tool showing the progress of a negotiation is shown in Figure 1.

The basic idea is that each engineer argues selfishly (i.e., it argues for actions that will lead to its goal being achieved) and the negotiation protocol tries to ensure that a fair (i.e., taking into account all the agents wishes) outcome is reached. The *priority* level of an agent affects how much weight the agent has in the negotiation. The way such a process leads to good overall decisions is described mathematically in Ossowski [Ossowski and García-Serrano, 1999]. The agent's *satisfaction* with an action is proportional to the relative reduction in the distance between the current state and its ideal state(s), that can be expected from that action as compared to other possible actions. So actions that are expected to lead to the biggest reduction in distance to the ideal states will *completely satisfy* the engineer. On the other hand, any actions expected to increase the distance to ideal states will *completely dis-satisfy* the engineer. All other actions, i.e., those that are expected to reduce the distance to the ideal state(s) but not so much as some other actions will satisfy the engineer to a proportional level.

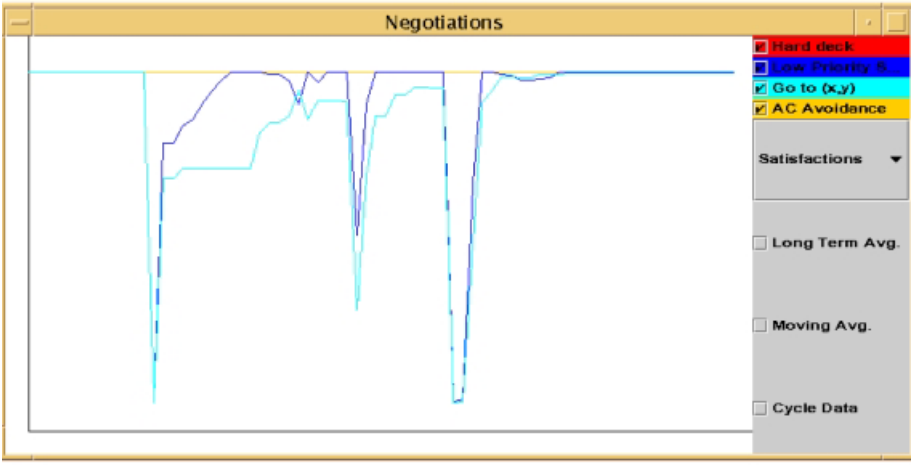


Fig. 1. A snapshot of the Negotiation Viewer showing each agent's satisfaction levels during the successful avoidance of an aircraft. Four agents are involved in the negotiation. Two of the agents, "Hard deck" (red) and "AC Avoidance" (yellow) are always completely satisfied (100% – top line). The "Go to waypoint" agent (light blue) is unsatisfied more than the "Smooth" agent (dark blue) (both lines go near the bottom of the window).

In the current EASE prototype, agents are internally controlled by single-layered state machines. Each state in the state machine may be a *success* state, a *failure* state or a regular (intermediate) state. A designer should design the agent so that when it is in a success state, it is achieving (or has achieved) the task assigned to it and when it is in a failure state, it is unable to achieve (temporarily or permanently) its assigned task. The success or failure information is

communicated to the contractor agent which can use the status information in it's decision making. Figure 2 shows a snapshot of the end-user tool for creating state machines.

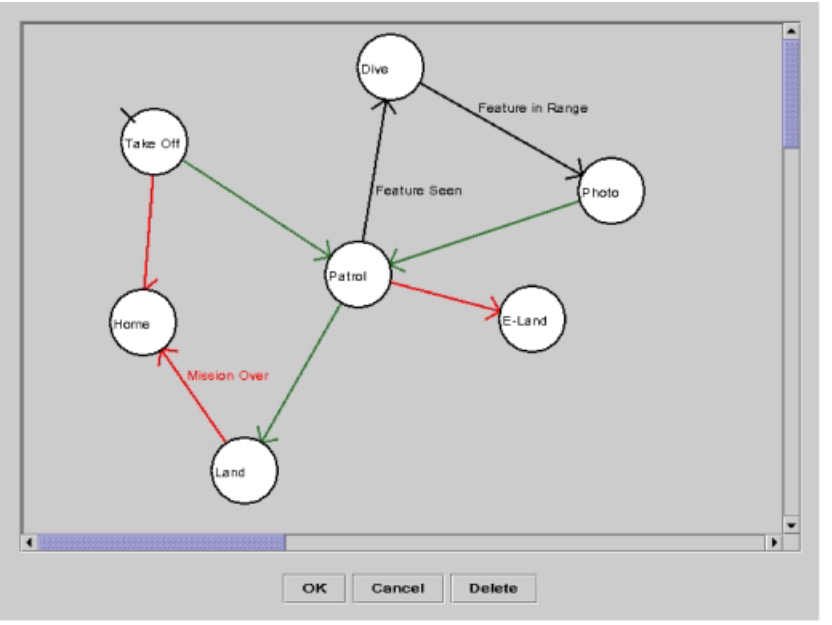


Fig. 2. The end-user state machine specification tool, showing a specification of a state machine for a simple patrol agent. Circles represent states, while arrows represent transitions. The state in the top left corner, i.e., take-off, with an extra line is the start state. Success transitions are lightly colored while normal transitions are annotated with the condition for their traversal.

Two key aspects of the EASE agents' team behavior facilitate effective AA. Firstly, an agent informs its contractor of the success or failure of its task. This allows contractors to react intelligently to the success or failure of sub-goals and take appropriate subsequent actions. The contractee informs its contractor whether its goal is currently being achieved (or maintained), the goal cannot be achieved, or normal progress is being made towards the achievement of the goal (i.e., there is no reason to believe the goal cannot be achieved but it is not yet achieved). These fairly abstract messages are the only communication that occurs between agents once a contract is made. The messages allow contractors to make decisions based on the status of sub-goals. Although some aspects of the organization's functioning might be improved if more status information were shared, the simplicity of the interconnections between agents makes the addition and removal of agents to and from the organization easier. Being able to add and

remove agents easily is critical when the *user* wants to add and remove agents, i.e., add or remove goals, at *runtime*.

At runtime the AA utilizes the success and failure messages to invoke the normal team mechanisms of the actor to smoothly integrate the changes specified by the user. In particular, when the user breaks a contract, i.e., removes an agent, the same messages are passed to the agent's contractor as if the agent "normally" failed or succeed. Hence, when the user takes an action the contractor agent has the required mechanisms to handle the situation and move on appropriately. This means that although the user is making "ad hoc" actions, the designer can, a priori, design the specification in such a way that the overall behavior of the actor should remain within reasonable bounds.

The second key mechanism of the team framework that facilitates effective AA is the low level negotiation mechanism. Behavior based architectures, like EASE, are intrinsically robust to failure [Goldberg and Mataric, 2000, Parker, 1998], due primarily to the flexibility of their behavior arbitration mechanisms. All "horizontal" interaction between agents, i.e., goal conflict resolution, takes place within the negotiation framework. The negotiation algorithm, being an adaptation of a behavior fusion algorithm, is well suited to having agents added and removed at unpredictable times. This allows agents to be removed from the MAS without serious disruption to other parts of the behavior.

2.1 Example

In this section a detailed example is given of the specification and running of a simple EASE actor. This example was demonstrated at Agents 2000 [Scerri and Reed, 2000a]. In the next section, this example is used to show how a user might intervene and take control, modifying the originally specified behavior of the actor online.

The example actor we describe is a pilot in an air-combat simulation. When the simulation begins, the aircraft is in the air. The pilot should fly the aircraft through three way points and then to a home position. Any other aircraft in the air should be avoided. For simplicity and clarity we consider the case of an aircraft with only two degrees of freedom: heading and altitude.

A design-time view of the agent hierarchy is shown in Figure 3. The *mission* agent has four states. In each of its states, the mission agent contracts a single *way point* agent. State transitions between the states of the *mission* agent occur when the *way point* agent it contracts reaches its way point (the *way point* agent will go into a success state and inform the mission agent of its success). There are three instances of a *way point* agent. Each of them has two states, a start state (fly) and a success state (done). In the start state the agent will negotiate using a satisfaction function that gives high satisfaction for heading suggestions toward the way point and low satisfaction for heading suggestions that are away from the way point. The transition between the states will occur when the agent reaches the way point.

The *smooth* agent, responsible for keeping the aircraft on a relatively straight course, has only a single state. In that state it has a satisfaction function that

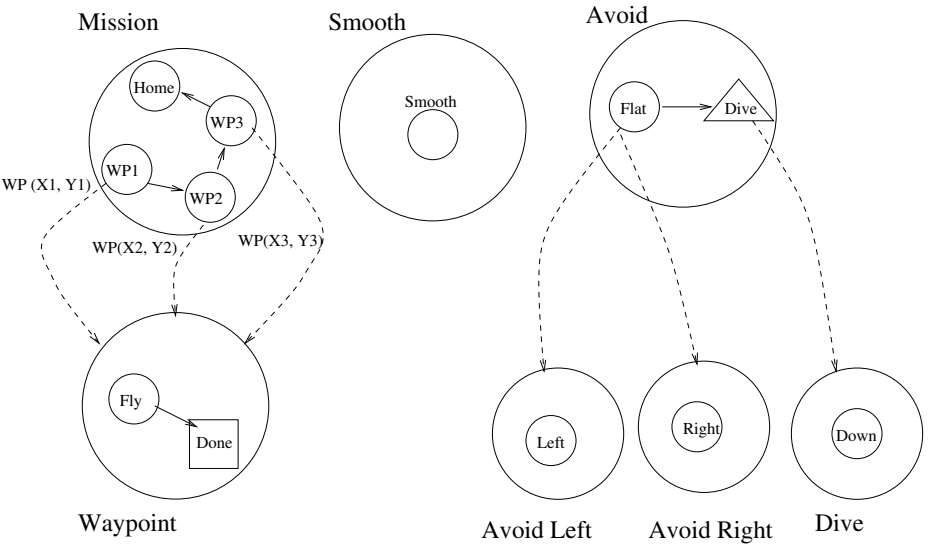


Fig. 3. The mission design specification for the example. There are three agents when the actor starts: "Mission", "Smooth", and "Avoid" (shown at the top of the figure). Contracts between agents are shown as dashed lines and the state machines are shown inside the agents.

prefers headings that are close to the current heading, i.e., it prefers slow turns, indirectly keeping the line of the aircraft smooth. This agent has a lower priority so that if a safety critical action cannot be done simultaneously with keeping the heading smooth the smooth agent will be overridden.

An *avoid* agent is responsible for avoiding a single aircraft (a special agent called a *list manager agent* is responsible for contracting one avoid agent for each detected aircraft). The satisfaction function in the starting state of an avoid agent is very simple. It returns high values when suggestions are made that lead to a relative heading of greater than 30 degrees from the obstacle aircraft and satisfaction values of zero otherwise. If the aircraft gets within a set distance of the obstacle aircraft a transition is made into the second state. The second state is a failure state. In this state the satisfaction function will argue for altitude values that will lead the aircraft to dive. In this state the agent's priority is high, giving it the most weight in the negotiations.

When the actor starts there are three agents active, the *mission* agent, *smooth* agent, and *avoid* agent (see Figure 3). As soon as the actor starts, the *mission* agent will contract a *way point* agent to go toward the first way point. When the aircraft reaches the first way point, that *way point* agent will transition to its success state. Because the *mission* agent has a transition dependent on contractee success it will transition to its next state. In the next state, it contracts another *way point* agent to get it to the next way point. As soon as another

aircraft is detected, an *avoidance* agent would be contracted specifically to avoid the detected aircraft. One such agent will be contracted for each aircraft detected.

3 Online Control of Agents

In the previous sections, the multi-agent system (MAS) that provides the action selection mechanism for an actor was described. In this section, the tools that allow runtime manipulation of the MAS, and hence, runtime control of the actor, are described. The tools can be separated into two categories, tools for observing the behavior of the MAS and tools for manipulating the MAS (to change the actor's behavior).

3.1 Observing the Actor's Multi-agent System

The main tool for observing the MAS is "The Boss" shown in Figure 4. A familiar tree layout shows all the currently active agents. Each node shows the name and current state of one of the agents. The layout of the tree reflects the hierarchical arrangement of contractor and contracted agents in the MAS. It is easy to see from the tree which agent is contracted by which other agent and, more generally, how a particular abstract task is currently broken down into parts and the overall state of the actor. Different types of agents are shown in different colors. Agents negotiating directly with a factory (at leaves of the hierarchy) are shown in red and agents contracting other agents (non-leaves in the hierarchy) are shown in green.

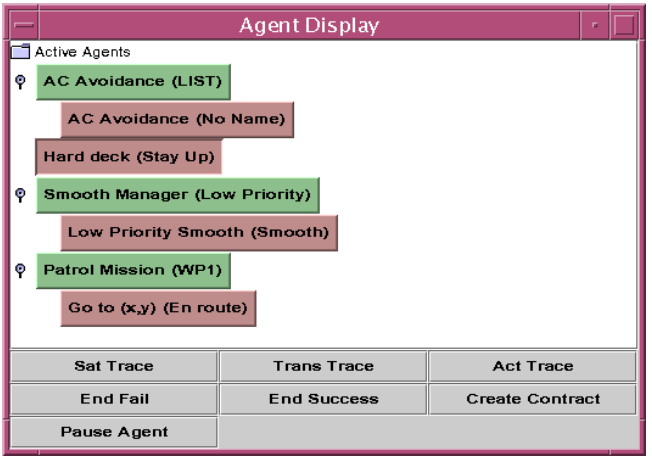


Fig. 4. The Boss display lists all active agents in an actor (top) and allows tracing, termination and suspension of a selected agent, and the creation of new agents (bottom buttons).

The top three buttons in the bottom of The Boss window (Figure 4) allow the user to view different aspects of the system. For agents involved in negotiations, a snapshot of the satisfaction function can be shown (see Figure 5). The snapshot shows, in great detail, the calculation the agent has performed in order to produce its satisfaction value for the current factory suggestion. For all the agents, traces of the calculations of their priority and state transition conditions can also be shown. This mechanism was designed primarily for debugging purposes, i.e., the designer can check the calculations at runtime to determine why agents are performing each action, but this information also aides the AA. For example, the calculation traces may allow a user to detect early on that the agent is about to do something undesirable or understand why it is doing something incorrectly.

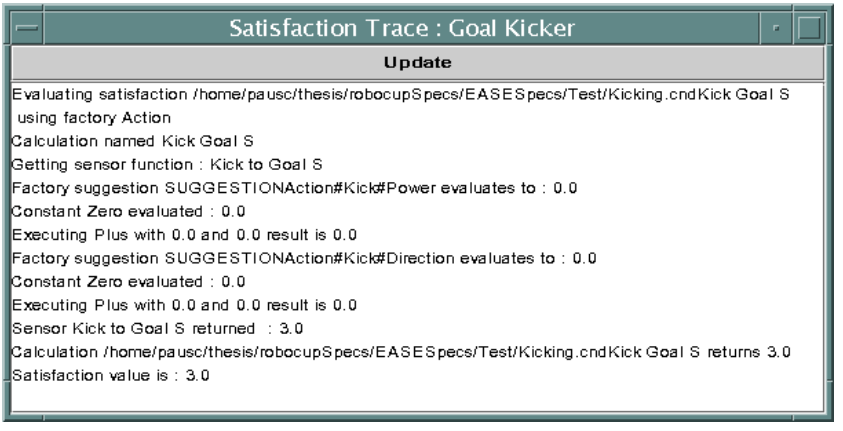


Fig. 5. The Calculation Trace window shows the details of the calculations one of the agents is performing. Each line shows the result of computing the value of one cell. The labels of the cells are printed with the corresponding intermediate results. The bottom line in the window shows that the final satisfaction value calculated is 3.0.

3.2 Manipulating the Actor's Multi-agent System

There are three ways that the user can manipulate the MAS: by breaking existing contracts, by creating new contracts and by suspending active agents. Each of the mechanisms is accessed through the same interface shown in Figure 4 (the same interface used to observe the agents as described in the previous section).

To break a contract between contractor and contractee, the user selects the agent and then selects either the "end fail" button or the "end success" button at the bottom of the window. The agent whose contract is broken ends the contracts of all agents it contracted. That is, breaking a contract with an agent that has contractees results in the whole hierarchy of agents headed by that agent being terminated.

If the "end failure" button was selected the selected agent sends a message to the contractor agent informing the contractor that its contractee has failed. Conversely, if the "end success" button is used a message indicating success is sent from the agent being terminated to its contractor. The contractor handles the termination of the contract as it would handle any success or failure of a contractee, i.e., by taking the same actions that it would have taken if the contractee had really succeeded or failed.

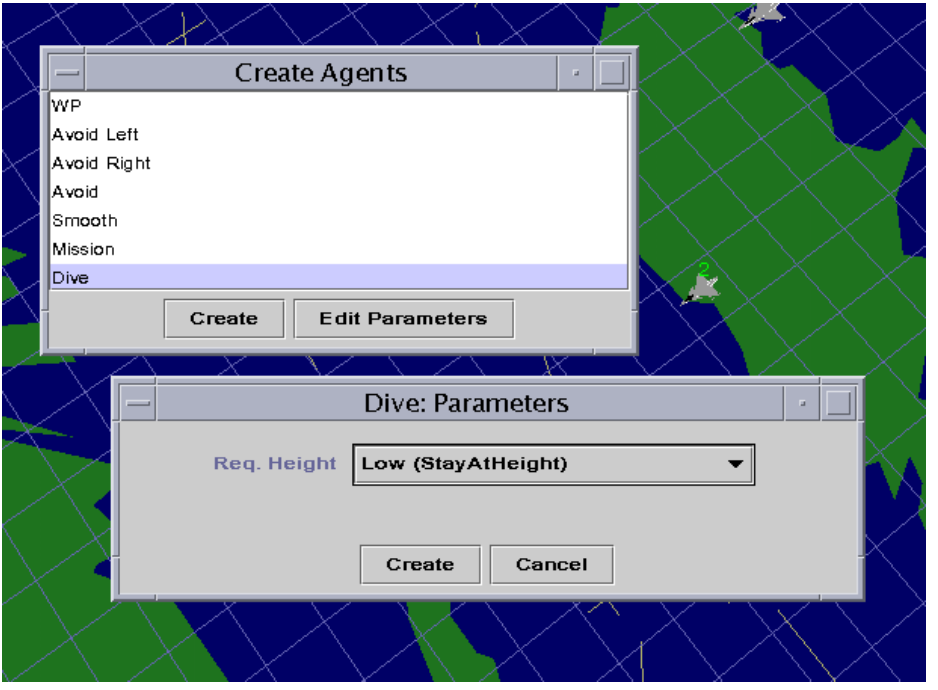


Fig. 6. Creation of new agents. In the example, the user wants to make the aircraft dive to a specified height. The user selects "Dive" for a dive agent in the upper window and then selects the "create" button. This leads to the lower window opening for the user to enter the dive altitude, in this case the user enters the parameter *low* (from the file "StayAtHeight").

For example, consider the actor specification given in the previous section. If a user decided that the aircraft should skip the first way point and go directly to the second, this could be achieved by selecting the *way point* agent (contracted by the *mission* agent) then selecting the "end success" button. The *way point* agent will leave the negotiations and inform the contractor (the *mission* agent) of the successful completion of its task. When the *mission* agent receives the

message it will make a success state transition and contract a new *way point* agent for going to the second way point.

The second way a user can manipulate the MAS is by contracting agents themself, i.e., by adding new agents to the MAS. Figure 6 shows another EASE tool, this tool allows agents to be created and contracted by the user. The "Create Agents" window comes up when the "create contract" button in the "Boss" window is selected. To create an agent, the user selects the agent's name and clicks on the "create agent" button. The agent will pursue a particular goal in the actor. If the agent contract has uninstantiated parameters (for example the location of a way point) a dialog box pops up allowing the user to instantiate appropriate values. In the example shown in the figure a *dive* agent has been created and the user is instantiating the parameter "Req. Height" (indicating the level to dive to) with the value "Low" (which has been defined as some value in the file "StayAtHeight"). The created agent becomes a part of the MAS and, just like other agents, contracts other agents or enters into negotiations over actor actions. Newly created agents are always top-level agents.

The third type of control the user has is the ability to suspend (pause) an agent within the actor. This is done by selecting the agent and then selecting the "pause agent" button shown at the bottom of The Boss tool (see Figure 4). All agents contracted, directly or indirectly (i.e., contracted by contractees), by the suspended agent are also suspended. No status messages are sent to the contracting agent of the suspended agent, who assumes that the execution of the task is ongoing. The "pause" button works as a toggle, i.e. a suspended agent can be restarted by selecting it again and clicking "pause agent".

4 Evaluation

AA is difficult to usefully test in a structured way because AA is designed to handle situations unanticipated by the designer – experiments created by a designer cannot test things they did not anticipate. Any structured experiments will likely be biased towards the features developers have included in the system. Hence, the best testing of AA is use in real world projects by real users. This is one approach being taken for the evaluation of EASE.

The original domain that EASE was used in was air-combat simulation with Saab's TACSI system [Saab, 1998]. During one early demonstration of EASE the potential for genuinely useful AA was unexpectedly tested. The author was demonstrating a simple air-combat mission to engineers from Saab Aerospace. Mid-way through the demonstration it became clear that there was an error in the aircraft avoidance specification created for that demonstration and that a mid-air collision was likely. Realizing this, a "height" agent was temporarily contracted which caused the EASE controlled aircraft to change altitude, avoiding the imminent collision. The AA turned a potentially disastrous demonstration into a success. The most pleasing part of the incident was that the AA system was able to handle a clearly unexpected situation involving a newly introduced bug in another part of the system.

EASE was also used in the development of a RoboCup simulation team [Noda, 1995]. During the development process for the Headless Chickens IV RoboCup football team [Scerri *et al.*, 2001], AA was extensively used. The information required for doing AA reasoning is similar to that required for debugging, hence the same interfaces used to present information for AA also provided useful information for debugging. However, the ability to see inside the agent for debugging was a small bonus compared to the ability to change the players' behavior while they ran. The functionality provided by the AA was very useful for experimenting with the behavior of the agent. The task of setting up specific test scenarios was made significantly easier because the agent could be manipulated at runtime, instead of the more standard process of stopping the game and changing the player specifications. Furthermore, players with considerable gaps in their functionality, e.g., not stopping when a goal was scored or not listening to referee calls, could still be effectively tested because the developer could come in and "help out" via AA, when the player encountered a situation it did not yet have the functionality to handle.

Other application areas, including disaster management simulations, command and control training and network management simulation are currently being investigated using EASE.

5 Conclusions

We used a team infrastructure to implement a flexible, robust AA mechanism for controlling actors in simulation environments. The user manipulates a multi-agent system controlling the actor, thereby changing the behavior of the actor.

The team infrastructure brings flexibility and adaptability to the architecture, allowing it to quickly adapt to changes made by the user. When the user changes one part of the multi-agent system, the team mechanisms coordinate to produce the new behavior. This means that the user needs to pay less attention to the details of the changes they intend to make, thus making their task easier. Furthermore, the team style of partioning the functionality of the system often makes it easier for the user to isolate the aspects of the actor's behavior that they want to observe and possibly change. Utilizing a flexible team infrastructure as the basis for AA seems to offer significant promise as a mechanism for making the AA more robust and easier to build.

Acknowledgments. This work is supported by The Network for Real-Time Research and Education in Sweden (ARTES), Project no. 0055-22, Saab Corporation, Operational Analysis Division, the Swedish National Board for Industrial and Technical Development (NUTEK) under grants IK1P-97-09677, IK1P-98-06280 and IK1P-99-6166, and the Center for Industrial Information Technology (CENIIT) under grant 99.7.

References

[Blumberg and Galyean, 1995] Bruce Blumberg and Tinsley Galyean. Multi-level control of autonomous animated creatures for real-time virtual environments. In *Siggraph '95 Proceedings*, pages 295–304, New York, 1995. ACM Press.

[Cremer *et al.*, 1995] James Cremer, Joseph Kearney, and Yiannis Papelis. HCSM: A framework for behavior and scenario control in virtual environments. *ACM Transactions on Modeling and Computer Simulation*, pages 242–267, 1995.

[Dorais *et al.*, 1998] G. Dorais, R. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost. Adjustable autonomy for human-centered autonomous systems on mars. In *Proceedings of the first international conference of the Mars society*, pages 397–420, August 1998.

[Falcone and Castelfranchi, 1999] R. Falcone and C. Castelfranchi. Levels of delegation and levels of help for agents with adjustable autonomy. In *Proceedings of AAAI Spring symposium on agents with adjustable autonomy*, pages 25–32, 1999.

[Goldberg and Mataric, 2000] D. Goldberg and M. Mataric. Robust behavior-based control for distributed multi-robot collection tasks. Technical Report IRIS-00-387, Institute for Robotics and Intelligent Systems, University of Southern California, 2000.

[Hexmoor, 1999] Henry Hexmoor, editor. *Workshop on Autonomy Control Software*. Autonomous Agents 1999, May 1999.

[Kortenkamp *et al.* 1999a] D. Kortenkamp, G. Dorias, and K. Myers, editors. *Proceedings of IJCAI99 Workshop on Adjustable Autonomy Systems*, August 1999.

[Kortenkamp *et al.*, 1999b] David Kortenkamp, Robert Burridge, Peter Bonasso, Debra Schrenkenghoist, and Mary Beth Hudson. An intelligent software architecture for semi-autonomous robot control. In *Autonomy Control Software Workshop, Autonomous Agents 99*, pages 36–43, 1999.

[Kortenkamp *et al.*, 2000] D. Kortenkamp, D. Keirn-Schreckenghost, and R. P. Bonasso. Adjustable control autonomy for manned space flight. In *IEEE Aerospace Conference*, 2000.

[Mataric, 1994] Maja Mataric. *Interaction and Intelligent Behavior*. PhD thesis, Massachusetts Institute of Technology, 1994.

[Minsky, 1988] Marvin Minsky. *The Society of Mind*. Simon and Schuster, 1988.

[Musliner and Krebsbach, 1999] D. Musliner and K. Krebsbach. Adjustable autonomy in procedural control for refineries. In *AAAI Spring Symposium on Agents with Adjustable Autonomy*, pages 81–87, Stanford, California, 1999.

[Noda, 1995] Itsuki Noda. Soccer server: A simulator of RoboCup. In *Proceedings of AI Symposium'95*, Japanese Society for Artificial Intelligence, December 1995.

[Ossowski and García-Serrano, 1999] S. Ossowski and A. García-Serrano. *Intelligent Agents V : Agent Theories Architectures and Languages*, chapter Social Structure in Artificial Agent Societies: Implications for Autonomous Problem Solving Agents, pages 133–148. Springer, 1999.

[Parker, 1998] Lynne E. Parker. Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[Perlin and Goldberg, 1996] Ken Perlin and Athomas Goldberg. Improv: A system for scripting interactive actors in virtual worlds. *Computer Graphics*, 29(3), 1996.

[Pirjanian and Mataric, 1999] Paolo Pirjanian and Maja Mataric. A decision theoretic approach to fuzzy behavior coordination. In *Proceedings, IEEE International Symposium on Computational Intelligence in Robotics & Automation (CIRA-99)*, Monteray, CA, Nov. 1999.

[Reed, 2000] N. Reed, editor. *Proceedings of PRICAI Workshop on Teams with Adjustable Autonomy*, Melbourne, Australia, 2000.

[Saab, 1998] Saab. *TACSI - User Guide*. Gripen, Operational Analysis, Modeling and Simulation, 5.2 edition, September 1998. in Swedish.

[Scerri and Reed, 2000a] P. Scerri and N. Reed. Real-time control of intelligent agents. In *Technical Abstracts of Technical Demonstrations at Agents 2000*, 2000.

[Scerri and Reed, 2000b] Paul Scerri and Nancy Reed. Creating complex actors with EASE. In *Proceedings of Autonomous Agents 2000*, pages 142–143, 2000.

[Scerri *et al.*, 2001] Paul Scerri, Nancy Reed, Tobias Wiren, Kikael Lönneberg, and Pelle Nilsson. Headless Chickens IV. In Peter Stone, Tucker Balch, and Gerhard Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, pages 493–496. Springer Verlag, Berlin, 2001.

[Schooley *et al.*, 1993] L. Schooley, B. Zeigler, F. Cellier, and F. Wang. High-autonomy control of space resource processing plants. *IEEE Control Systems Magazine*, 13(3):29–39, 1993.

[Tambe, 1996] Miland Tambe. Teamwork in real-world, dynamic environments. In *Proceedings of the Second International Conference on Multi-agent Systems*, Kyoto, Japan, 1996.

Dimensions of Adjustable Autonomy

K. Suzanne Barber[1], Cheryl E. Martin[1], Nancy E. Reed[2], and David Kortenkamp[3]

[1]Laboratory for Intelligent Processes and Systems, Electrical & Comp. Eng.,
The University of Texas at Austin, Austin TX, USA 78712,
{barber|cemartin}@mail.utexas.edu

[2]Dept. of Computer & Info. Sci, Linköping University, Sweden SE-58183,
nanre@ida.liu.se

[3]NASA Johnson Space Center – ER2, Houston TX, USA, 77058
korten@smtp.traclabs.com

Abstract. This panel discussion focused on the dimensions of autonomy that are defined and adjusted by the research of the workshop participants. In particular, the groups were able to explain and relate their work in adjustable autonomy to each other's work, although the topics and application areas are very diverse. We describe three dimensions of an agent's autonomy that are adjustable - the agent's *independence* in how it carries out its goals, its *control* over choosing between options to carry out its goals, and the set of *goals* it has to achieve. Each of these dimensions of autonomy can be adjusted over time.

1 Introduction

The panel discussions at the First Workshop on Teams with Adjustable Autonomy, held at PRICAI 2000, focused on identifying the primary dimensions of agent autonomy that are adjusted by the research work of the various workshop participants. Although this workshop and three previous workshops (Musliner and Pell 1999, Hexmoor 1999, Kortenkamp 1999) have addressed the topic of adjustable autonomy, this topic area is still in its infancy and its boundaries and core ideas have yet to be precisely defined. The work presented at all these workshops has varied widely with respect to the interpretation of "agent autonomy" and "adjustable autonomy." As a result, it can be difficult to relate the different research approaches in the papers and presentations to one another. The panel discussion at this workshop gave researchers the opportunity to ask and answer the question, "How does each piece of research presented here connect to other pieces of research in the context of adjustable autonomy?" This paper describes the framework that the workshop participants developed to compare and contrast their work on adjustable autonomy.

The concept of agent autonomy is broad, thus no widely accepted definition for this concept exists in the field of agent-based systems. This poses a fundamental challenge for developing a consensus about "adjustable autonomy." Nevertheless, it is well accepted among workshop participants that "autonomy" can be adjusted and that this endeavor can lead to the development of systems with useful agent behavior. Therefore, the unifying model of adjustable autonomy sought by this panel discussion

R. Kowalczyk et al. (Eds.): PRICAI 2000 Workshop Reader, LNAI 2112, pp. 353–361, 2001.

is developed from descriptions of these useful adjustments rather than from controversial previously proposed definitions of autonomy. The autonomy adjustments considered by this workshop lie in various dimensions. The proposed model of adjustable autonomy attempts to relate these dimensions to one another.

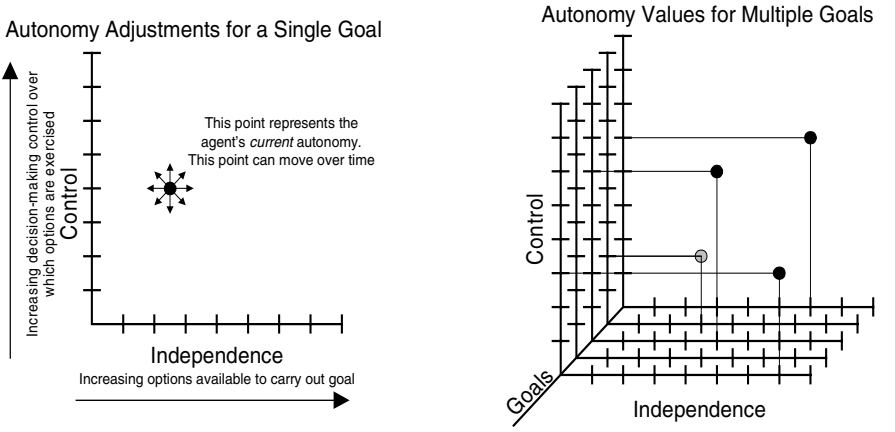


Fig. 1. Relationships of Dimensions of Adjustable Autonomy

As exemplified by the research presented at this workshop, most instances of "adjustable autonomy" research actually adjust different things. This is the fundamental problem faced when relating each piece of research to its counterparts. We identified each of the "characteristics" that were being adjusted as different dimensions of autonomy. Three primary dimensions were identified and related to one another to form a unified context for discussions of adjustable autonomy.

The three primary dimensions of adjustable autonomy we identified are as follows:

(1) **Independence** – changing the number and kinds of options an agent can exercise in carrying out a particular goal. As the number of exercisable options increases, the agent's autonomy increases.
(2) **Control** – changing the degree of decision-making control an agent has over which options it exercises in carrying out a particular goal. As the degree of decision-making control an agent has increases, the agent's autonomy increases.
(3) **Goals** – changing the goals (and/or their priorities) that an agent is pursuing over time. For each goal an agent pursues, the first and second dimensions determine the amount of autonomy the agent has, with respect to that goal.

Each of these three dimensions relates to the others, as shown in Figure 1. An agent is assumed to have one or more goals at all times. For each goal, an agent has a particular degree of control over the decision-making process that determines how that goal should be carried out (autonomy in the decision-making process). For each goal, an agent has a particular set of options or alternatives available for carrying out that goal (autonomy with respect to independence). The following example highlights these three dimensions.

Let *Broker* be an agent whose goal is to "make money for *Investor* over the long-term horizon." Let *Investor* be a person who uses *Broker* to invest money and who can adjust the autonomy of *Broker* (or on a constituent basis, the autonomy per goal, held by *Broker*). Assume *Broker* has the capability, in general, to invest money in bonds, mutual funds, individual stocks, and precious metals.

Investor can adjust the autonomy of *Broker* by placing constraints on the options *Broker* can exercise to make money. For example, *Investor* can limit *Broker* to investments of only precious metals and bonds. Alternatively, *Investor* can allow *Broker* to invest in bonds, mutual funds, and individual stocks. Here, *Investor* is adjusting *Broker*'s autonomy along the first dimension, independence. *Investor* could allow *Broker* to have maximum autonomy along this dimension by allowing *Broker* to exercise any or all of its possible buying capabilities.

Investor can adjust the autonomy of *Broker* along the second dimension by taking over some or all of the decision-making control for how the money is invested. For example, *Investor* could allow *Broker* to have maximum decision-making control by simply transferring the investment money to *Broker* and letting *Broker* decide where and when to invest it. Alternatively, *Investor* can take away some of the decision-making control from *Broker* and make *Broker* consult with *Investor* about every transaction. At the other extreme, *Investor* could allow *Broker* to have no decision-making control and simply order *Broker* to make each individual transaction that *Investor* wants.

Investor could also give *Broker* another goal or goals, which may or may not conflict with the goals *Broker* already has. For example, *Investor* could give *Broker* an additional goal of "generate monthly income for *Investor*." The options and decision-making control that *Broker* can exercise for this goal may vary.

These dimensions help describe what is being adjusted as adjustable autonomy is realized. Different dimensions may be adjusted, but in each case, these adjustments occur dynamically, over time, as a system operates. Therefore, in the example, *Investor* may change *Broker*'s goals, control, and independence multiple times in the course of a week or a month. This time-scale may actually become very short for some other types of systems. To realize adjustable autonomy, *Investor* must be able to both increase and decrease *Broker*'s autonomy as the system operates and changes.

The following sections describe research contributions of the workshop participants addressing the realization of these adjustments. Each contribution is discussed within the context of the dimensions of adjustable autonomy introduced above.

2 Independence

Independence refers to the number of different ways that an agent is allowed to try to achieve its goals. The more options an agent has the more autonomous it can be said to be. An example is the 3T architecture developed at NASA JSC (Bonasso *et al* 1997a). The 3T architecture consists of a low-level set of skills that perform actions in the environment, a mid-level reactive planner (Firby's RAPs system (Firby 1987)) that activates and deactivates sets of *skills* and a high-level planner that places reactive plans (RAPS) on the agenda and monitors resources and time. This architecture allows for independence in many different ways. First, the RAPS system

allows the developer to encode different *methods* to accomplish the same tasks. The RAPS system can choose amongst these methods depending on sensed data or previous experience. Typically one of those methods is to ask a human to perform the task (see Bonasso *et al* 1997b). Second, the planner replans when the current plan is failing and attempts to come up with an alternate path to the goal situation. Finally, by breaking the agent's sensing and acting capabilities into low-level primitive skills the architecture allows for easy recombination of these skills to accomplish tasks in novel ways.

Adjusting an agent's level of autonomy with respect to independence can take many different forms. For example, at the planning stage a knowledgeable user can restrict the search space of the planner. Various forms of mixed-initiative planning (Burstein and McDermott 1996) do just that – use human knowledge to guide the planning activities of an agent. This restricts the agent's independence in choosing plans and, therefore, adjusts the agent's level of autonomy. Or a human user can restrict the agent from using a particular method or from using a particular action. For example, we have an autonomous control system for an advanced water recovery system (see Bonasso 2001) and we can tell the control system not to use particular pumps or valves when accomplishing a task.

In general, restricting an agent's independence is a good way to guide it in the right direction without needing to take direct control. This allows the agent to maintain awareness of the situation and react to new data, while still allowing the human to use his or her capabilities. Any adjustably autonomous agent needs to have mechanisms for restricting its actions and guiding its deliberations.

3 Control

The dimension of autonomy described in this section considers autonomy as decision-making control. This viewpoint holds that an agent's degree of autonomy, with respect to some goal that it actively uses its capabilities to pursue, is the degree to which the decision-making process, used to determine how that goal should be pursued, is free from intervention by any other agent (Barber and Martin, 2001a).

The capability of adjustable autonomy with respect to decision-making control is realized in Adaptive Decision-Making Frameworks (ADMF)(Barber et al. 2001a). The following subsection presents a representation of autonomy as decision-making control that is based on the representation of agent decision-making frameworks and decision-making interaction styles. A decision-making framework (DMF) specifies the set of interactions exercised by members of an agent group as the group determines how a goal or set of goals should be achieved. The interactions specified by a decision-making framework are (1) decision-making control relationships and (2) authority-over relationships. A specification of decision-making control dictates which agents make decisions about how to achieve a goal. A specification of authority-over dictates to which agents the decision-makers can assign tasks (i.e. which agents the decision-makers have authority over). An agent's individual decision-making interaction style describes how that agent participates in the overall framework. Agents adopt a distinct decision-making interaction style for each goal

they pursue. Agents' decision-making interaction styles can be described informally along a spectrum as shown in Fig. 2.

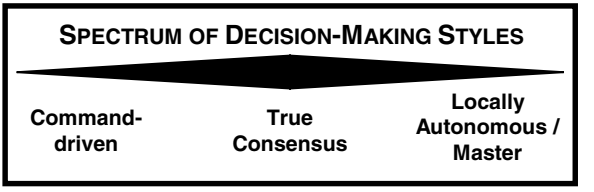


Fig. 2. The spectrum of agent autonomy with respect to decision-making interaction styles.

The three discrete categories of decision-making interaction styles, which define salient points along the spectrum, are labeled in Fig. 2:

- **Command-driven –** The agent does not make any decisions about how to pursue its goal and must obey orders given by some other agent(s).
- **True Consensus –** The agent works as a team member, sharing decision-making control equally with all other decision-making agents.
- **Locally Autonomous / Master –** The agent makes decisions alone and may or may not give orders to other agents.

A decision-making framework is composed of a coherent set of individual decision-making styles for all participating agents (e.g. a Master/Command-driven framework, an All Consensus framework, etc.).

A computational representation of agent decision-making interaction styles is needed in order to assign, evaluate, and modify these concepts in an automated fashion (i.e. to realize the capability of ADMF). Such a representation gives the agent or its designer something to set, a "knob to turn" so to speak, allowing decision-making control to be assigned and adjusted.

Decision-making interactions can be represented by the tuple (*D*, *G*, *C*) (Barber et al., 2000), where *D* identifies the decision-makers and their relative strengths in the decision-making process, *G* identifies the set of goals that are the focus of the decision-making framework, and *C* declares the authority-over constraint. Table 1 presents the specification for this representation, and each component is explained in detail in the following paragraphs. An algorithm for classifying assignments to the (*D*, *G*, *C*) representation according to the decision-making interaction styles defined in Fig. 2 is introduced in (Barber et al., 2000).

The set *D* identifies which agents make decisions about how the intended goals listed in *G* should be pursued. *D* also describes the relative strength of these decision-making agents in the decision-making process. The evaluation of the relative strength of any agent in the decision-making process is based on an analogy to a simple voting process in which every vote must be cast. In this process, every decision-making agent receives an integer number of votes, greater than or equal to one. In *D*, the tuple (a_x , v_{a_x}) represents an agent who is making decisions about how to pursue the goal(s) in *G* along with the number of votes that agent can cast to determine the

Table 1. Specification for Decision-Making Interaction Representation

Decision Making Interaction Representation (G, D, C)		
D	decision-makers	$\{(a_0, v_{a_0})[, (a_1, v_{a_1}), ..., (a_n, v_{a_n})]\}$, or $\{(a_1, v_{a_1})[, ..., (a_n, v_{a_n})]\}$
G	focus	$\{g_i^{a_0}[, g_j^{a_1}, ..., g_k^{a_n}]\}$ or $\{g_j^{a_1}[, ..., g_k^{a_n}]\}$
C	authority-over constraint	$\{a_0[, a_1, ..., a_n]\}$ or $\{a_1[, ..., a_n]\}$

overall decision of the group. Each agent in the set D may play an equal part in determining how to pursue G, or some agents may have more votes than others.

As the focus of the decision-making framework, G, identifies the goal(s) about which agents are making decisions. Any agent may make decisions for goals it intends to achieve as well as for goals that other agents intend to achieve. Additionally, agents may combine their goals for concurrent solution in a "you scratch my back, I'll scratch yours" fashion. The set G may either contain a goal intended by the self-agent (a_0), plus any number of goals intended by other agents (1 per other agent), or G may contain only goals intended by other agents and no goal intended by the self-agent. The set G must identify at least one goal intended by some agent. If the number of elements in G is greater than one ($|G| > 1$), then the decision-making agents must find a solution for all constituent goals concurrently.

The set C simply lists the agents who are bound to carry out the decisions made by the decision-makers identified in the set D. The decision-makers are said to have authority over the agents in C because these agents have previously committed to the decision-making framework, thereby committing to accept task assignments from the decision-makers that are required to carry out the goal(s) identified by G. The authority-over constraint, C, ensures that some agent(s) will carry out the decisions of the decision-making group. If an agent listed in C fails to accept its required task assignment, it must pay a penalty for breaking its commitment to the decision-making framework (Martin, 1997).

Previous experiments exploring the Adaptive Decision-Making Frameworks (ADMF) capability have shown that no one decision-making framework performs best across various situations that may be encountered at run-time (Barber et al., 1999). In fact, the performance of agents operating under a given decision-making framework differs greatly across run-time situations. Given these differences, further research has been undertaken to show that agents capable of ADMF (who implement the decision-making framework that performs best for every different situation encountered) perform better overall than agents who are not able to adapt to various situations in this manner (Barber and Martin, 2001b). Overall these experiments

show that implementing the capability of Adaptive Decision-Making Frameworks is a useful method of realizing adjustable autonomy in agent-based systems.

4 Goals

This section considers the adjustment in autonomy of an agent produced by changes to its set of goals or objectives (and/or their priorities). For each goal an agent pursues, the independence and control dimensions described above determine the amount of autonomy the agent has with respect to each goal.

An agent development environment called EASE (End-user Actor Specification Environment) (Scerri and Reed, 2000a, Scerri and Reed, 2000b) allows a user to adjust the autonomy of an actor interacting with a simulation environment by changing the goals the agent is pursuing or can pursue. Examples of actors are a pilot controlling an aircraft in a tactical simulation or a football player in a football simulation. Each actor moves in its environment and its actions are produced by the reasoning of a hierarchy of agents. Each agent in this hierarchy is relatively simple and pursues one goal or objective. An agent accomplishes its goal by contracting other agents (which are then instantiated and fall just below the agent in the hierarchy) to satisfy parts of its goal or objective. A user is able to add and remove agents in the hierarchy, thereby changing the goals or objectives the agent can pursue (adjusting the autonomy of the actor). A user may also change the priorities of goals or change the way a goal is pursued.

The user can make two types of changes in the autonomy of an actor - temporary ones that modify specific agents in the actor, and permanent ones that modify a generic agent or some other variables. The temporary modifications are ones done to the agents that are currently negotiating contracts within an actor. These modifications are adding agents (goals), removing agents (goal and all subgoals), and suspending agents (temporarily inactivating the goal and its subgoals from the reasoning process).

The persistent modifications a user can make include adding new or removing existing types of agents to the pool of available generic agents that can be contracted, adding capabilities to an existing generic agent, removing or modifying the capabilities of an agent, and modifying the importance (priority) of the agent. Since these changes are done to the generic agents instead of to particular instances of agents, the changes affect every agent created from the generic agent in future simulations. A user can also modify the values of constants that are used by agents in their decision-making. This will affect all future decisions of any agent using the constant in its reasoning. This has the potential to affect decision-making control and make alternative solutions available to solve goals.

Certainly there are other ways to change the autonomy of an actor by changing its goals. EASE is a prototype implementation of one set of ways for doing the adjusting.

5 Discussion

Falcone & Castelfranchi (2000) describe levels of delegation and adoption as they relate to adjustable autonomy. They describe two categories of meanings for autonomy. The first is self-sufficiency - not being dependent on others for one's goals. The second is performance or executive autonomy, which the authors divide into 2 subcategories, planning autonomy, how much of the goal planning the agent does for itself, and goal autonomy, whether the agent is allowed to have/find goals.

The dimensions of autonomy described in the previous sections relate to the types and levels of delegation described in Falcone & Castelfranchi (2000) as follows. They describe an n-dimensional space of delegation types to characterize the autonomy of an agent. In their example (n = 3) they show the types: 1) delegation of control, 2) specification-based kinds of delegation and 3) interaction-based kinds of delegation. Their delegation of control dimension ranges from full control to no control. This is approximately the same as the control dimension described above.

Secondly, their specification-based kinds of delegation ranges from closed delegation (complete specification of the task) to open delegation (less complete specification of the task). This is similar to the independence dimension above.

Finally, their interaction-based delegation ranges from strict delegation (contracts) to weak delegation, where an agent believes another agent will accomplish a desired goal on its own. This dimension overlaps both the control dimension and the goal dimensions above. Control: "Who" has control over what goals the agent should achieve? Goals: Can a goal be "forced" upon an agent?

6 Conclusions

The panel discussion was useful in identifying common features within the research presented as well as identifying some differences. This paper also ties the discussion to other work in adjustable autonomy. It seems we have only touched the surface of the definition and use of adjustable autonomy and teamwork in agent systems.

Acknowledgements. Part of this work was supported by the Swedish National Board for Industrial and Technical Development (NUTEK), the network for Real-time education and research in Sweden (ARTES), and the Swedish Foundation for Strategic Research (SSF).

References

Barber, K. S., Goel, A., and Martin, C. E. 1999. The Motivation for Dynamic Adaptive Autonomy in Agent-based Systems. In *Intelligent Agent Technology: Systems, Methodologies, and Tools. Proceedings of the 1st Asia-Pacific Conference*

Barber, K. S., Goel, A., and Martin, C. E. 2000. Dynamic Adaptive Autonomy in Multi-Agent Systems. *The Journal of Experimental and Theoretical Artificial Intelligence, Special Issue on Autonomy Control Software* 12(2): 129-147.

Barber, K. S. and Martin, C. E. 2001a. Dynamic Adaptive Autonomy in Multi-Agent Systems: Representation and Justification. To appear in the *International Journal of Pattern Recognition and Artificial Intelligence. Special Issue on Intelligent Agent Technology*.

Barber, K. S. and Martin, C. E. 2001b. Dynamic Reorganization of Decision-Making Groups. *Fifth international Conference on Autonomous Agents(Agents 2001)*. J. Muller, E. Andre, S.Sern and C. Frasson, Eds., Montreal, Canada, May 30 - June 1, ACM Press, New York, 513-520.

Bonasso, R.P., 2001. "Intelligent Control of a NASA Advanced Water Recovery System," *Proceedings of the Sixth International Symposium on Artificial Intelligence in Space* (i-SAIRAS 2001).

Bonasso, R. P., Firby, R.J, Gat, E., Kortenkamp, D., Miller, D., and Slack, M. 1997a. "Experiences with an Architecture for Intelligent, Reactive Agents," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol.9, No. 1, 1997.

Bonasso, R.P., Kortenkamp, D. and Whitney, T. 1997b. "Using a Robot Control Architecture *on IAT, Hong Kong, December 14-17, 1999*, Liu, J. and Zhong, N., Eds. Singapore: World Scientific, 131-140.

Burstein, Mark and McDermott, Drew 1996. "Issues in the Development of Human-Computer Mixed-Initiative Planning," *Cognitive Technology: In Search of a Humane Interface*, eds. B. Gorayska and J. L. May, Elsevier Science, Amsterdam, 1996.

Falcone, R. and Castelfranchi, C. 2000. "Levels of Delegation and Levels of Adoption as the basis for Adjustable Autonomy". *Advances in Artificial Intelligence '99*, Lamma, E. and Mello, P., Eds., LNAI 1792, Springer-Verlag, Berlin, 273-284.

Hexmoor, H., 1999, Editor, Workshop on Autonomy Control Software, The Fourth International Conference on Autonomous Agents (Agents 99), Seattle, WA, May 1.

Kortenkamp, D., 1999. Editor, Workshop on Adjustable Autonomy Systems, The Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden, August 1.

Martin, C. E. 1997. Representing Autonomy in Sensible Agent-based Systems. Master's Thesis, Electrical and Computer Engineering, University of Texas at Austin.

Musliner, D. and Pell, B., 1999, Editors, AAAI Spring Symposium on Agents with Adjustable Autonomy, Stanford, CA, March 22-24.

Scerri, P. and Reed, N., 2000a, "Creating Complex Actors with EASE", *Fourth International Conference on Autonomous Agents (Agents 2000)*, C. Sierra, M. Gini and J. S. Rosenschein, Eds., June 3-6, ACM Press, New York, 142-143.

Scerri, P. and Reed, N., 2000b, "Real-time Control of Intelligent Agents", Technical Summaries of the Software Demonstration Sessions, *Fourth International Conference on Autonomous Agents (Agents 2000)*, Josep Puyol-Gruart, Ed., June 3-6, ACM Press, New York, 28-29.

Author Index