CEPIS, Council of European Professional Informatics Societies, is a non-profit organisation seeking to improve and promote high standards among informatics professionals in recognition of the impact that informatics has on employment, business and society.

CEPIS unites 36 professional informatics societies over 32 European countries, representing more than 200,000 ICT professionals.

CEPIS promotes


UPGRADE is the anchor point for UPENET (UPGRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
- Mundo Digital, digital journal from the Italian CEPIS society AICA
- Novática, journal from the Spanish CEPIS society ATI
- Pliroforiki, journal from the Cyprus CEPIS society CCS
- Pro Dialog, journal from the Polish CEPIS society PTiPiPS

Publisher


UPGRADE was created in October 2000 by CEPIS and was first published by Novática and INFOMATIK/INFORMATIQUE, bimonthly journal of SVfISI (Swiss Federation of Professional Informatics Societies, <http://www.svfisi.ch/>).

Editorial Team

Chief Editor: Rafael Fernández Calvo, Spain, <rfcalvo@ati.es>
Associate Editors:
- François Louis Nioieto, Switzerland, <nioieto@acm.org>
- Roberto Carmel, Italy, <rcarmel@dgt.unipv.it>
- Zakaria Maamar, Arab Emirates, <zakaria.maamar@zu.ac.ae>
- Soraya Kousaïd Mostefaoui, Switzerland, <soraya.kousaid@unifr.ch>

Editorial Board
Prof. Woolfied Stucky, CEPIS Past President
Prof. Nello Scambrato, CEPIS Vice President
Fernando Piera Gómez and Rafael Fernández Calvo, ATI (Spain)
François Louis Nioieto, NICT (Switzerland)
Panicos Mavrou (<Pliroforiki, Cyprus>)
Andrzej Marciniak (<Pro Dialog, Poland>)

English Editors: Mike Anderson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennermore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Redman.

Cover page designed by Antonio Crespo Foix, © ATI 2004
Layout: Pascale Schönmann

Editorial correspondence: see "Editorial Team" above
Advertising correspondence: <novatica@ati.es>

Upgrade Newslist available at <http://www.upgrade-cepis.org/pages/editinfo.html#newslist>

Copyright © Novática 2004 (for the monograph and the cover page)
© CEPIS 2004 (for the sections MOSAIC and UPENET).
All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, contact the Editorial Team. The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5235

Next issue (October 2004): "Software Process Technologies"
(The full schedule of UPGRADE is available at our website)

2 From the Editors’ Desk
Worth To Be Mentioned
The Editorial Team of UPGRADE communicates that (1) Pliroforiki, journal published by the Cyprus CEPIS society CCS, has joined UPENET, and (2) a Call for Contributions for our MOSAIC section has been released.

A World of Agents

Guest Editors: Pedro Cuesta-Morales, Zahia Guessoum, Juan Carlos González-Moreno, and Juan Pavón-Mestras

Joint monograph with Novática*

3 Presentation
Agent Technologies at Work – Pedro Cuesta-Morales, Zahia Guessoum, Juan Carlos González-Moreno, and Juan Pavón-Mestras

The authors describe trends and challenges that will need to be addressed over the next ten years in order to progress in the field of Agents Technologies.

6 Challenges for Agent Technology Moving towards 2010 – Michael Luck and Peter McBurney

10 Open Directions in Agent-Oriented Software Engineering – Franco Zambonelli and Andrea Omicini

This paper identifies key open research directions in the development process of Agent-Oriented Software Engineering (AOSE.)

15 Verification and Validation Techniques for Multi-Agent Systems – Rubén Fuentes-Fernández, Jorge J. Gómez-Sanz, and Juan Pavón-Mestras

This article gives an overview of, and presents some of the new methods used in, Multi-Agent Systems verification and validation.

20 Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities of Services – Marco Aiello and Paolo Giorgini

This article gives an overview of, and presents some of the new methods used in, Multi-Agent Systems verification and validation.

27 Developing a Multi-Agent System Using MaSE and JADE – Pedro Cuesta-Morales, Alma-Maria Gómez-Rodríguez, and Francisco J. Rodríguez-Martínez

An example of how to develop a Multi-Agent System using currently available methods and tools is described in this paper.

33 Engineering Multi-Agent Systems as Electronic Institutions – Carles Sierra, Juan A. Rodríguez-Aguilar, Pablo Nortiega Blanco-Vigil, Josep-Lluís Arcos-Rosell, and Marc Esteva-Vivancos

This paper introduces an integrated development environment that supports the engineering of a particular type of distributed systems, namely multi-agent systems, ‘as electronic institutions’.

40 The Baghira Multiaagent Learning Environment: An Educational Community of Artificial and Human Agents – Sylvie Pesty and Carine Webber

This paper presents a practical application of an agent-based architecture, which has been developed using the methodological framework defined by case-based reasoning systems.

56 Integration and Reuse Based on Web Technology and Aspect Object Model – Yauheni Veryha and Eckhard Krause

This paper presents an object-oriented integration framework that uses Web technology and Aspect Object Model to integrate applications based on different component models.

61 News & Events: News from CEPIS, ECDL, and EUCIP; QofIS 2004 (Fifth International Workshop on Quality of Future Internet Services)

**UPENET (UPGRADE European NETwork)**

63 From Pro Dialog (Poland):
Large-Scale Software Development in a Multinational Software Corporation – Jacek Czerniak and Wojciech Śpiewak

This paper presents a process of code management based on the software integrity inspection process performed in R&D departments of a large company.

68 From Novática (Spain):
How will we computer professionals earn a living? (And why don’t you teach for free?) – Ricardo Galli-Granada

In this article the author, a university lecturer in Computer Science, takes an unorthodox look at the computer profession and responds to sharp questions posed by an opponent of free software / open source.

* This monograph will be also published in Spanish (full issue printed; summary, abstracts and some articles online) by Novática (<http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática)) and in Italian (online edition only, containing summary abstracts and some articles) by the Italian CEPIS society ALSI (Associazione nazionale Laureati in Scienze dell’informazione e Informatica) and the Italian IT portal Tecnoteca at <http://www.tecnoteca.it/>.
Worth to Be Mentioned

Dear Readers,

Since our June 2004 issue the following newsworthy events have taken place.

“Pliroforiki”, Journal of the Cyprus CEPIS Society CCS, Joins UPENET

Pliroforiki (Greek for “Informatics”), quarterly journal published, in English, by the Cyprus CEPIS society CCS (Cyprus Computer Society, <http://www.ccs.org.cy/about/>), has joined UPENET (UPGRADE European NETwork). It was founded in 2002 and its editor is Panicos Masouras.

Efprosdektos / Welcome!
Pliroforiki is thus the fourth journal to join UPENET, in addition to Mondo Digitale, digital journal of the Italian CEPIS society AICA; Pro Dialog, journal of the Polish CEPIS society PTI-PIPS; and Novática, journal of the Spanish CEPIS society ATI.

More information about UPENET can be found at <http://www.upgrade-cepis.org/pages/upenet.html>.

MOSAIC’s Call for Contributions

In order to expand the scope of our section MOSAIC section, which was created in January 2004 with a view to complementing our usual monographs with articles about various ICT (Information and Communication Technologies) matters, a Call for Contributions has been released. Those interested in submitting papers can find there information about aims, topics, formats and procedures of MOSAIC.

The text of the Call for Contributions is available at <http://www.upgrade-cepis.org/pages/edintinfo.html>.

Cordially,
The Editorial Team of UPGRADE

Rafael Fernández Calvo, <rfcalvo@ati.es>, Chief Editor and Editor of NOVÁTICA

Associate Editors:
Francois Louis Nicolet, <nicolet@acm.org>; Roberto Carniel, in charge of the Italian version, <rcarniel@dgt.uniud.it>; Zakaria Maamar, <Zakaria.Maamar@zu.ac.ae>, and Soraya Kouadri Mostéfaoui <soraya.kouadrimostefaoui@unifr.ch>, in charge of MOSAIC.
Presentation

Agent Technologies at Work

Pedro Cuesta-Morales, Zahia Guessoum, Juan-Carlos González- Moreno, and Juan Pavón-Mestras

1 Introduction: The Agent Paradigm

During the last decade, Agent-Oriented Software Engineering (AOSE) has attracted the attention of a large community of researchers from many different fields, including artificial intelligence and distributed systems. This interest has been motivated by the potential benefits of the agent paradigm, which needs to be integrated in software engineering practices in order to be applicable in the software industry as a whole.

Although there are many definitions of the agent concept (see, for instance, those included in Michael Wooldridge’s book “An Introduction to Multiagent Systems”, John Wiley & Sons, 2002), most of them identify their distributed nature, autonomy, sociability (hence the term Multi-Agent System, MAS, as agents normally collaborate within organizations to achieve common goals), adaptability to the environment, and even mobility through a network of computer resources.

Agents have been applied with different purposes and in different environments: for personal assistants, to providing support for collaborative work, for trading and negotiation in e-markets, in huge social simulation systems, for web information systems, for e-games, etc.

A common classification scheme of agents is the weak and strong notion of agency. In the weak notion of agency, agents have their own will (autonomy), they are able to interact with each other (social ability), they respond to stimuli (reactivity), and they take the initiative (pro-activity). In the strong notion of agency, in addition to the characteristics displayed by the weak notion of agency, agents can also move around (mobility), they are truthful (veracity), they do what they’re told (benevolence), and they will perform in an optimal manner to achieve goals (rationality). Due to the fact that existing agents have more in common with software than with intelligence, they will be referred to as software agents or agents in this context.

2 Bringing Agent Technology to Market

Successfully bringing agent technology to market requires techniques that reduce the perceived risk inherent in any new technology, by presenting the new technology as an incremental extension of known and trusted methods, and by providing explicit engineering tools to support proven methods of technology deployment. Applied to agents, these insights imply an approach that:

The Guest Editors

Pedro Cuesta-Morales is Associate Professor at Universidad de Vigo, Spain, since 1992. Nowadays he is a member of the Intelligent Agent research group (see <http://gwai.ei.uvigo.es>). He has supervised 6 research projects with companies and he has participated in four national research projects funded by the Spanish Government. He has cooperated in the editing of two books, and is author of several chapters of books and articles in journals. He is author of more than 25 publications in national and international conferences. His current research interests include: intelligent agent, multi-agent systems, agent-oriented software engineering and information retrieval. <pcuesta@uvigo.es>

Zahia Guessoum is Assistant Professor in Computer Science and a member of the Objects and Agents for Simulation and Information Systems (OASIS) research theme of the Laboratoire d’Informatique de Paris 6 (LIP6). She received her doctorship (1996) in Computer Science from Université Paris VI, France. Her research interests are about modular and reflective agent architectures, agent-oriented software engineering, fault-tolerant mechanisms for large-scale multi-agent systems, and multi-agent applications (simulation of economics models, simulation of cell population models,...). <Zahia.Guessoum@lip6.fr>

Juan-Carlos González-Moreno holds a PhD degree in mathematics from Universidad Complutense de Madrid, Spain (1994) and since 1994 he has been an Associate Professor, firstly at the above-mentioned university and since 2000 at Universidad de Vigo, Spain. He has supervised and participated in several research projects with companies or funded by the European or the Spanish government. He has cooperated in the edition of a number of books and is also the author of many chapters of books and articles in journals. He has authored more than 50 publications in national and international conferences. He is currently leader of the Intelligent Agent research group (see <http://gwai.ei.uvigo.es>) at the Universidad de Vigo. <jcmoreno@uvigo.es>

Juan Pavón-Mestras holds a PhD degree in Computer Science from the Universidad Politécnica de Madrid, Spain, (1988) and since 1997 is an Associate Professor at the Universidad Complutense de Madrid. He has been working in the R&D departments of Alcatel in Spain, France, and Belgium, and in Bellcore, USA, especially involved the development of software component architectures for distributed systems and their application for multimedia services on broadband networks and UMTS. Currently he is the leader of the Grasias research group at the Universidad Complutense de Madrid, working on the development of multi-agent systems for web service personalisation and knowledge management. He is also working on the definition of a methodology and the tools for the development of multi-agent systems, INGENIAS (more information about Grasias can be found at <http://grasias.fdi.ucm.es/jpavon>). <jpavon@sip.ucm.es>
introduces agents as an extension of active objects: an agent is an object that can say "go" (flexible autonomy as the ability to initiate action without external invocation) and "no" (flexible autonomy as the ability to refuse or modify an external request);

promotes the use of standard representations for methods and tools to support the analysis, specification, and design of agent software.

As pointed out by some of the articles appearing in this monograph, until recently developing a MAS has been more of an art than a structured discipline. We can currently find tools able to produce complete MAS from a specification, libraries of components that deal with concrete MAS issues (distributed planning, reasoning, learning), and theories that describe MAS behaviour and properties. Knowing all of them requires a great effort. There are surveys which facilitate the task, but it is hard to give an overall view of what software, theories, methodologies exist, and how they are applied to MAS development.

What Is This Monograph about?

Having into account all the above, we have selected a set of papers that address some of the most important aspects and issues of this promising field.

To begin with, Michael Luck’s and Peter McBurney’s paper “Challenges for Agent Technology Moving towards 2010” gives us an answer to the question about how Agent Technology is evolving and summarises the current state-of-the-art in this field, identifying trends and challenges that will need to be addressed over the next 10 years in order to progress in the field and reap the benefits. Similarly, but in this case specifically oriented towards the MAS developing process and methodologies, the paper by Franco Zambonelli and Andrea Omicini “Open Directions in Agent-Oriented Software Engineering”, aims to identify key open research directions in the development process of AOSE. Also related with the development process is the contribution from Rubén Fuentes-Fernández, Jorge J. Gómez-Sanz, and Juan Pavón-Mestras “Verification and Validation Techniques for Multi-Agent Systems”, which gives an overview of, and presents some of the new methods used in, MAS verification and validation. Following the same line of argument, the work by Marco Aiello and Paolo Giorgini “Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities”, proposes the use of the agent-oriented methodology Tropos for the analysis of web service requirements, and describes how it can be used to model quality of service requirements.

An example of how to develop a MAS using currently available methods and tools is presented by the contribution from Pedro Cuesta-Morales, Alma-Maria Gómez-Rodríguez, and Francisco J. Rodríguez-Martínez “Developing a Multi-Agent System Using MaSE and JADE”. The paper “Engineering Multi-Agent Systems as Electronic Institutions” by Carles Sierra, Juan A. Rodríguez-Aguilar, Pablo Noriega Blanco-Vigil, Josep-Lluís Arcos-Rosell and Marc Esteva-Vivancos introduces an integrated development environment that supports the engineering of a particular type of distributed systems, namely multi-agent systems, as electronic institutions.

The monograph closes with some articles related to the development of practical and real MAS systems. The first one, “The Baghera Multiagent Learning Environment: An Educational Community of Artificial and Human Agents”, by Sylvie Pesty and Carine Webber, focuses on the multiagent learning environment named Baghera, built on a two-level multiagent architecture. The second, “Management of a Surveillance Camera System Using Software Agents”, by Jesús García-Herrero, Javier Carbó-Rubiera and José M. Molina-López, shows a MAS system that applies data mining techniques to learn, from real situations, how to extract knowledge from the environment in order to detect conflictive situations in a distributed surveillance camera system, and to improve the cooperation between cameras. The last article, “An Agent-Based Architecture for Developing Internet-Based Applications”, by Juan M. Corchado-Rodriguez, Rosalía Laza-Fidalgo and Luis F. Castillo-Ossa, presents a practical application of an-agent based architecture, which has been developed using the methodological framework defined by case-based reasoning systems.

Let us finally express our thanks to all the authors for their valuable collaboration and also to the editors of UPGRADE and Novática for the opportunity they have given us to guest-edit this monograph, with the hope that its contents will be both interesting and useful to readers of the two journals.

Translation by Steve Turpin
Useful References about MAS

For those interested in obtaining more detailed information about Multi-Agent Systems and Agent Technologies the following sources complement the references provided by the authors of the papers included in this issue.

Books
J. Ferber.
Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence, Addison-Wesley, 1999.
M. N. Huhns, M. P. Singh, L. Gasser.
M. Klusch.
M. Luck, R. Ashri and M. d’Inverno.
M. Luck, V. Marik, O. Stepankova, R. Trappl (eds.)
G. Weiss (ed.).
M. Wooldridge.

Events

• AOSE (Agent-Oriented Software Engineering Workshops):

• AOIS (Agent-Oriented Information Systems):

• CIA (Cooperative Information Agents Workshops):

• ESAW (Engineering Societies in the Agents World Workshop):
  <http://www.irit.fr/ESAW04>.

• EUMAS (European Workshop on Multi-Agent Systems):

Journals
• Autonomous Agents and Multi-Agent Systems (Kluwer Academic Publisher).

• The Journal of Artificial Societies and Social Simulation.
  <http://jasss.soc.surrey.ac.uk/JASSS.html>.

Web Sites


• AgentLink: <http://www.agentlink.org>.

• Agents Portal: <http://aose.ift.ulaval.ca>.

• ASM (Agentes y Sistemas Multiagentes):
  <http://agentes.ei.uvigo.es> (in Spanish.)


• FIPA (Foundation for Intelligent Physical Agents):

• ISOA (Ingeniería de Agentes Orientada a Objetos):
  <http://ma.ei.uvigo.es> (in Spanish.)


• UMBC AgentWeb: <http://agents.umbc.edu>.
Challenges for Agent Technology Moving towards 2010

Michael Luck and Peter McBurney

While there are many real successes of agent technologies, there is still much to be done in research and development for the full benefits to be achieved. This is especially true in the context of environments of pervasive computing devices that are envisaged in coming years. This paper summarises the current state-of-the-art of agent technologies and identifies trends and challenges that will need to be addressed over the next 10 years to progress the field and realise the benefits. It outlines a roadmap that identifies successes and challenges, and points to future possibilities and demands, and suggests that agent technologies are fundamental to the realisation of next generation computing.

Keywords: Agents, Applications, Challenges, Multi-agent Systems, Roadmap, Technology.

1 Introduction

An agent is a computer system capable of flexible autonomous action in a dynamic, unpredictable and open environment. Agent technologies are a natural extension of current component-based approaches, and have the potential to greatly impact the lives and work of all of us and, accordingly, this area is one of the most dynamic and exciting in computer science today. The impact of agent technologies will occur in a number of ways: firstly, as a metaphor for the design of complex, distributed computational systems; secondly, as a source of technologies for such computing systems, and thirdly, as models of complex real-world systems, such as those found in biology and economics. This paper examines these various impacts and presents a status report of current research and application developments.

As the computing landscape moves from a focus on the individual standalone computer system to a situation in which the real power of computers is realised through distributed, open and dynamic systems, we are faced with new technological challenges and new opportunities. The characteristics of dynamic and open environments in which, for example, heterogeneous systems must interact, span organisational boundaries, and operate effectively within rapidly changing circumstances and with dramatically increasing quantities of available information, suggest that improvements on the traditional computing models and paradigms are required. In particular, the need for some degree of autonomy, to enable components to respond dynamically to changing circumstances while trying to achieve over-arching objectives, is seen by many as fundamental.

1.1 Agents as Design

The use of agents as an abstraction tool, or a metaphor, for the design and construction of systems provided the initial impetus for developments in the field. On the one hand, agents offer an appropriate way to consider complex systems with multiple distinct and independent components. On the other, they also enable the aggregation of different functionalities that have previously been distinct (such as planning, learning, coordination, etc) in a conceptually embodied and situated whole. Thus these notions provide a set of technology areas that relate directly to these abstractions in the design and development of large systems, of individual agents, of ways in which agents may interact to support these concepts, and in the consideration of societal or macro-level issues such as organisations and their computational counterparts. Current efforts span diverse areas including agent-oriented software engineering; agent architectures; mobile agent systems; agent infrastructure; and electronic institutions.

Michael Luck is a Professor in the Intelligence, Agents, Multi-media Group in the School of Electronics and Computer Science at the University of Southampton, United Kingdom. He has worked in the field of agent technology and multiagent systems for more than 10 years, and is a member of the Advisory Boards of FIPA (the agent standards body), EUMAS (the European agent workshop), and CEEMAS (the Central and Eastern European agent conference). From 2000 to 2003, he was the Director of AgentLink II, the European Network of Excellence for Agent-Based Computing, and is currently a member of its Executive Committee. <mml@ecs.soton.ac.uk>

Peter McBurney is a Lecturer in the Department of Computer Science at the University of Liverpool, UK. He has a First Class Honours degree and University Medal in Mathematical Statistics from the Australian National University, Canberra, and a PhD in Computer Science from the University of Liverpool. He is co-Director of AgentLink III, and is the author of a number of papers on multi-agent systems in journals and at major conferences. With Michael Luck, he was a member of the core team that produced the AgentLink II Roadmap, which outlined a strategy for research and deployment in agent technologies in Europe over the next decade. Previously, McBurney has been a management consultant, advising global telecommunications companies on market entry strategy and implementation. <p.j.mcburney@csc.liv.ac.uk>
1.2 Agent Technologies

Agent-based approaches have been a source of technologies to a number of research areas, both theoretical and applied. These include distributed planning and decision-making, automated auctions, and information systems. The challenge is to create effective solutions that can be applied to the development and deployment of multi-agent systems, but they also need to be augmented to suit the differing demands of this new paradigm.

- **Provide effective agreed standards to allow open systems development**

In addition to standard languages and interaction protocols, open agent societies will require the ability to collectively evolve languages and protocols specific to the application domain and to the agents involved. Some work has commenced on defining the minimum requirements for a group of agents with no prior experience of each other to evolve a sophisticated communications language, but this work is still in its infancy. Research in this area will draw on linguistics, social anthropology, biology, the philosophy of language and information theory.

- **Provide semantic infrastructure for open agent communities**

At present, information agents exist in academic and commercial laboratories, but are not widely available in real-world applications. The move out of the laboratory is likely to happen in the next ten years, but requires: a greater understanding of how agents, databases and information systems interact; investigation of the real-world implications of information agents (for example, including the economic effects of shopbots); and development of benchmarks for system performance and efficiency. In order to support this, further needs include: new web standards that enable structural and semantic description of information; and services that make use of these semantic representations for information access at a higher level. The creation of common ontologies, thesauri or knowledge bases play a central role here, and merits further work on the formal descriptions of information and, potentially, a reference architecture to support the higher level services mentioned above.

- **Develop reasoning capabilities for agents in open environments**

At present, organisational approaches do not adequately handle the issues inherent in open multi-agent systems, namely heterogeneity of agents, trust and accountability, failure handling and recovery, and societal change. The next challenge for agent-based computing is to develop appropriate representations of analogous computational concepts to the norms, legislation, authorities, enforcement, etc., that can underpin the development and deployment of dynamic electronic institutions. Similarly, virtual organisations involve dynamic coalitions of small groups that can provide more services and make more profits than an individual group. Moreover, such coalitions can disband when they are no longer effective. At present, coalition formation for virtual organisations is limited, with such organisations largely static. The automation of coalition formation will save both time and labour, and may be more effective at finding better coalitions than humans in complex settings. Related issues include negotiation and argumentation, and domain-specific models of reasoning, both of which may be used to form such groups of agents in open environments.

2 The Agent Technology Roadmap

Based on the current status of agent-based computing, we envision four major phases of agent research and development over the next decade [1]. Current deployments typically centre on closed agent systems with ad-hoc designs, predefined communications protocols and scalability only in simulations. Note that this can have a negative impact on interoperability with external non-agent legacy systems (which currently comprise the vast majority of existing computer systems), and on the question of how systems and solutions can migrate to incorporating agent-based concepts. If agents are to succeed, we cannot afford to start from scratch, but need to show how existing solutions and systems can migrate towards agent-based systems and solutions.

The second phase, covering the period to 2006, will increasingly see the use of semi-structured agent communications languages (such as FIPA ACL, <http://www.fipa.org>), top-down design methodologies such as GAIA [2], and scalability extended to predetermined and domain-specific environments. The third phase, covering approximately 2006–2008, will see the use of agreed protocols and languages, and of standard, agent-specific design methodologies in open agent systems in specific domains (such as those in bioinformatics and e-commerce). More general scalability, to include arbitrary numbers and diversity of agents, in each such domain will likely be achieved by this phase, with bridging agents translating between domains. The final stage, from 2010 or so onwards, will see truly-open and fully-scalable multi-agent systems, across domains, with agents capable of learning appropriate communications protocols upon entry to a system, and with protocols emerging and evolving through actual agent interactions. This bears strong similarities to IBM’s ongoing research project on autonomic computing and existing Semantic Web objectives.
research include developing distributed models of profile management, as well as more general distributed agent learning techniques rather than just single agent learning in multi-agent domains. Developing approaches to personalisation that can operate in a standards-based, pervasive computing environment presents many interesting research challenges, including how to integrate machine learning techniques (for profile adaptation) with structured XML-based profile representations. Another area deserving of greater activity is that of distributed profile management – a task for which the agent paradigm should be well suited. The impact of the emerging Semantic Web on approaches for wrapper induction and text-mining also requires careful study.

- **Develop agent ability to adapt to changes in environment**
  Even though learning technology is clearly crucial for open and scalable multi-agent systems, it is still in early development. While there has been progress in many areas, such as evolutionary approaches and reinforcement learning, these have still not made the transition to real-world applications. Reasons for this can be found in problems of scalability and in user trust in self-adapting software. In the longer term, learning techniques are likely to become a central part of agent systems, while the shorter term offers application opportunities in areas such as interactive entertainment, which are not safety-critical.

- **Ensure user confidence and trust in agents**
  Collaboration of any kind, especially in situations in which computers act on behalf of users or organisations, will only succeed if there is trust. For this trust to be given requires a variety of factors to be in place. First, a user must have confidence that an agent or group of agents which represents them within an open system will act effectively on their behalf – it must be at least as effective as the user would be in similar circumstances. Second, agents must be secure and tamper-proof, and must not reveal information inappropriately (e.g., bank account details). There is much work on system security, cryptography and privacy which can be exploited and adapted for use in agent technology. Finally, if a user is to trust the outcome of an open agent system, they must have confidence that agents representing other parties or organisations will behave within certain constraints. Mechanisms to do this include: reputation mechanisms; the use of norms (social rules) by all members of an open system; self-enforcing protocols, which ensure that it is not in the interests of any party to break them; and electronic contracts.

2.2 Challenges for the Agent Community

Achieving this vision will require considerable investment in research and development in a number of areas of applied and pure research, and in commercial deployment and implementation. To achieve the full potential of agent approaches and technologies from our current position, we recommend that research and development resources be focused along several key directions, as follows.

- **Leveraging underpinning work in related areas of Computer Science, such as software engineering, distributed computing and object technologies. This is likely to become increasingly important as agent technologies converge with other approaches, such as Web Services, semantic web technologies, P2P computing, and so on.**
- **Strengthening links with other areas of Computer Science working on different problems, such as the communities active in artificial life, traditional mathematical modelling & simulation, semantic web activities, pervasive computing and uncertainty in AI.**
- **Strengthening links with other disciplines, particularly Economics, Management Sciences, Marketing, Logic, Philosophy, Biology, Sociology and Political Science.**
- **Encouraging industry adoption and commercial development of agent technologies, by identifying the obstacles to take-up and developing appropriate training and support mechanisms, software tools, prototypes, and case studies to overcome these obstacles.**
- **Assessment of non-functional requirements such as dependable agent systems addressing issues including trustworthiness and dynamics of adaptable complex systems.**
- **Relating agent standards, tools and concepts to industrially accepted standards for development and middleware.**

3 Application Opportunities

There are a number of existing and emerging application domains for agent technologies and multi-agent systems. We present a brief description of several of these domains, to demonstrate their wide range and diversity. They indicate the potential impact of agent-related technologies on human life and society. Many other very exciting and potentially very large areas for application are already being investigated, such as health care and manufacturing, but we will not examine them further here.

3.1 Ambient Intelligence

The notion of Ambient Intelligence has largely arisen through the efforts of the European Commission in identifying challenges for European research and development in Information Society Technologies [3]. Aimed at seamless delivery of services and applications, it relies on three identified pillars of ubiquitous computing, ubiquitous communication and intelligent user interfaces, yet it offers perhaps the strongest motivation for, and justification of, agent technologies. The Ambient Intelligence (AmI) vision describes an environment of potentially thousands of embedded and mobile devices (or software artefacts) interacting to support user-centred goals and activity. This suggests a component-oriented view of the world in which the artefacts are independent and distributed. The consensus is that autonomy, distribution, adaptation, responsiveness, and so on, are the key characterising features of these Ambient Intelligent artefacts, and in this sense they share the same characteristics as agents.

In particular, these Ambient Intelligence artefacts are likely to be function-specific (though possibly configurable to tasks) and will, of necessity, need to interact with numerous other
A World of Agents

Artifical artefacts in the environment around them in order to achieve their goals. Interactions will take place between pairs of artefacts (in one-to-one cooperation or competition), between groups of artefacts (in reaching consensus decisions), and between artefacts and the infrastructure resources that comprise their environments (such as large-scale information repositories, or other supporting resources, possibly through agent encapsulation). Interactions like these enable the establishment of electronic institutions or virtual organisations, in which groups of agents come together to form coherent groups able to achieve some overarching goals.

Importantly, interactions will also occur between artefacts and users, potentially requiring greater sophistication in interface issues [4], and in user understanding (and modelling). Also, the openness of the system, and its heterogeneity, will require the employment of learning and adaptation techniques, since many properties of the environment and other agents cannot be known at design time.

3.2 Grid Computing

The high-performance computing infrastructure, known as the Grid, for supporting large-scale distributed scientific endeavour has recently gained heightened and sustained interest from several communities, as a means of developing eScience applications such as those demanded by e.g., the bioinformatics scenarios described above, the Large Hadron Collider facility at CERN (Conseil Européen pour la Recherche Nucléaire – European Laboratory for Particle Physics, Geneva, Switzerland), engineering design optimisation, and combinatorial chemistry. Yet it also provides a computing infrastructure for supporting more general applications that involve large-scale information handling, knowledge management and service provision.

It is natural to view large systems in terms of the services they offer, and consequently in terms of the entities providing or consuming services. Grid applications, in which typically many services may be involved, spread over a geographically distributed environment, which new services join and existing ones leave, thus very strongly suggest the use of agent-based computing. In this view, agents act on behalf of service owners, managing access to services, and ensuring that contracts are fulfilled. They also act on behalf of service consumers, locating services, agreeing contracts, and receiving and presenting results. Just as in the Ambient Intelligence vision, agents will be required to engage in interactions, to negotiate, and to make pro-active run-time decisions while responding to changing circumstances. In particular, agents will need to collaborate and to form coalitions of agents with different capabilities in support of new virtual organisations. Such virtual organisations have been identified by Foster [5] as the tool with which to unwrap the power of the Grid.

Initially geared towards high performance computing, Grid computing is now being recognised as the future model for service-oriented environments, within and across enterprises, especially through Grid Services, which bring Grid concepts to Web Services in a common framework [6]. The impact will be larger than just virtual organisations – a global company is much like such a virtual organisation and will require similar technology.

3.3 Bioinformatics

Another area of application in biology is in Bioinformatics. With the information explosion caused by genomics and proteomics there is a great need for automated information-gathering and information-inference tools. Information-gathering agents may provide assistance to human researchers in finding appropriate research literature or in conducting automated or semi-automated testing of data. In addition, data mining agents may present human researchers with a set of potential hypotheses that can be induced from the data sources. In particular, the kinds of resources available in the bioinformatics domain, with numerous databases and analysis tools independently administered in geographically distinct locations, lend themselves almost ideally to adoption of a multi-agent approach. Here, the environment is open and distributed with resources entering and leaving the system, there are large numbers of interactions between entities for various purposes, and the need for automation is substantial and pressing. Some early work in this direction, using agents for genome analysis, is demonstrated by the GeneWeaver project in the UK [7]. More substantial work is now underway on the use of agents as part of a large-scale eScience project on a Bioinformatics Grid testbed, also in the UK [8].

3.4 Electronic Business

To date agents have been used in the first stages of eCommerce, product and merchant discovery and brokering [9]. The next step will involve moving into real trading – negotiating deals and making purchases. This stage will involve considerable research and development, including generating new products and services such as market-specific agent shells, payment and contracting methods, risk assessment and coverage; quality and performance certification, security, trust, and individualisation. Researchers will need to look to different fields that have dealt with interaction problems, such as game theory, economics and sociology.

It can be argued that the real impact of electronic commerce will be on a dramatic change in the supply chain. If a consumer can contact directly the producer instead of a reseller it might produce an increase in efficiency of the overall supply chain. These changes in the supply chains will permit new markets to appear, old markets to change and the participation of new players. These observations raise some broader questions about eCommerce in general, and the speeding-up effects of agents in particular. Consumers who are excluded from the eCommerce loop may find their prices and choice become worse.

In the very near future a boom in agent-mediated auctions is expected. The auction is a long-established and well-understood trading mechanism, and available agent technology can support such agent-mediated auction houses. In the longer term, full-supply chain integration is the aim.
3.5 Simulation

An important category of applications of multi-agent systems is in simulation of natural or artificial societies. These applications include education and training systems, scenario exploration and policy systems, and entertainment systems, though we focus here on scenario exploration.

Social simulation is somewhat unusual in that it does not require many of the challenges listed earlier to be addressed in order for it to succeed in the timescales considered in this paper. Since simulations are by their nature closed (even though they may model open systems) they are almost immediately enabled. However, there are many open issues to be resolved before agent-based simulation models can be applied more widely to public policy domains. For example, there is as yet no general understanding of what constitutes good performance by a multi-agent system, except perhaps in some domains. There is no guarantee, for example, that an agent society in which different species of agents co-evolve in the course of their interactions with one another will progress in any sense; later generations of a species may be less fit than earlier generations of that same species when pitted against earlier generations of their competitor species. In such a case, at which timepoint should the simulation be terminated? Different termination points may lead to different assessments of system performance and different recommendations to policy makers. Indeed, the question of performance assessment of multi-agent systems is part of a larger, mostly open, question of decision support systems in general.

Despite the work that remains to be done, the phenomena of emergent and normative behaviour are increasingly important in cooperating systems. Dynamic adaptation of behaviour in distributed systems can be applied beneficially in a variety of fields. Examples are traffic management (policies, measures, patterns), water management, privacy (legal principles as norms), sustainable environment (policy modelling for understanding effects of policies for, e.g., urban planning, mixed transport), logistics (dynamic planning and control), health care (dynamic patient rostering and resource allocation), education, physical systems (water motion, physical laws), biological systems (genomics, neurology), economic models and business process automation. Agent-based social simulation can provide insight into the structure and effects of policies and norms and can assist in understanding and modifying interaction patterns where appropriate (and possible).

4 Towards the Next Generation through Agent Technology

The ideas and technologies discussed in this document are neither arbitrary nor abstract. They can be seen in IBM’s articulation of the concept of Autonomic Computing to identify self-healing technology akin to the human body and its management by the autonomic nervous system. They can be seen in HP’s Planetary Computing initiative to develop an adaptive large-scale computing infrastructure. They can be seen in the vision of both Philips Electronics and the European Commission in their work on developing a view of Ambient Intelligence for the near term and for the next 50 years. Agent technology is a central tool in the efforts to achieve these grand challenges. It already offers real benefits, but much more value lies in what it will do in the future.

Acknowledgements

This paper is a revised version of the executive summary of the AgentLink Roadmap for Agent-Based Computing that was prepared as part of the activities of AgentLink II, the European Network of Excellence for Agent-Based Computing (IST-1999-29003).

References


Open Directions in Agent-Oriented Software Engineering

Franco Zambonelli and Andrea Omicini

Agent-based computing is a promising approach for developing applications in complex domains. However, despite the great deal of research in the area, a number of challenges still need to be faced to make agent-based computing a widely accepted paradigm in software engineering practice, and to turn agent-oriented software abstractions into practical tools for facing the complexity of modern application areas. In this paper, after a short introduction to the key concepts of agent-based computing and to the state of the art in the area, we try to identify a few key open research directions.

Keywords: Agent-Oriented Software Engineering, Large-scale Multi-Agent Systems, Multi-Agent Systems.

Introduction

The power of agents and Multi-Agent Systems (MASs) to facilitate the complexity of a variety of today’s ICT scenarios is becoming increasingly recognised. For instance, several industrial experiences already testify the advantages of using agents and MASs in manufacturing processes, Web services, Web-based computational markets and distributed network management. In addition, several studies examine the possibility of exploiting agents and MASs as enabling technologies for a wide range of future scenarios, such as Pervasive Computing, Grid computing and the Semantic Web.

However, the emerging general understanding is that MASs are more than an effective technology and indeed represent a novel general-purpose paradigm for software development [16][6]. In fact, agent-based computing introduces novel abstractions and provides a new way of seeing software, different from traditional (i.e., object- and component-based) software engineering approaches. Specifically, the traditional way of conceiving software – in terms of functional entities interacting with each other in a client-server fashion – is substituted by a perspective in which software is modelled and designed in terms of autonomous software entities (agents), situated in an environment and interacting with it, and flexibly achieving their goals by coordinating with one another in terms of high-level protocols and languages.

The above features are well suited to tackle the complexity of software development in modern scenarios: (i) the autonomy of application components reflects the intrinsically decentralised nature of modern distributed systems, and can be considered as the natural extension of the notions of modularity and encapsulation to deal with the increased complexity of the scenarios. Also, it can help in modelling open systems whose components may be owned by different stakeholders [11]; (ii) the explicit presence of an environment in which agents are situated reflects the fact that software systems are increasingly tending to interact either with the physical world (e.g., to control some physical process) or with a computational world (e.g., Web resources); (iii) the flexible way in which agents operate and interact (both with each other and with the environment) is suited to the dynamic and unpredictable scenarios in which software is expected to operate.

Paralleling the increasing acceptance of agent-based computing as a novel software engineering paradigm, there has been a great deal of research related to identifying and defining suitable models and techniques aimed at supporting the development of complex software systems in terms of MASs [5]. Such research efforts, which can be roughly grouped under the term “Agent-Oriented Software Engineering” (AOSE) [6][2], are endlessly proposing a variety of new metaphors, formal modelling approaches, development methodologies and modelling techniques, specifically suited to the agent-oriented paradigm. Nevertheless, the research is still in its early stages, and several challenges need to be faced before AOSE can live up to its promises and become a widely accepted and a practical usable paradigm for the development of complex software systems.

Franco Zambonelli is professor of Computer Science at the University of Modena and Reggio Emilia, Italy, since 2001. He obtained the Laurea degree in Electronic Engineering in 1992, and the PhD in Computer Science in 1997, both from the University of Bologna, Italy. His current research interests include: distributed and pervasive computing, multiagent systems, agent-oriented software engineering. In these areas, he has published over 120 papers and co-edited 7 books. He is a member of IEEE, ACM, AIIA, and TABOO. <franco.zambonelli@unimore.it>

Andrea Omicini is professor of Computer Science and Engineering at the University of Bologna in Cesena, Italy, since 2001. He obtained the MSc in Electronic Engineering in 1991, and the PhD in Computer Science in 1995, both from the University of Bologna, Italy. His current research interests include: coordination models and infrastructures, multiagent models and technologies, agent-oriented software engineering, and declarative approaches and technologies. In these areas, he has published over 110 papers and co-edited 8 books and 5 special issues of international journals. He is a member of ACM, IEEE, AIIA, and GULP/ALP. <aomicini@deis.unibo.it>
2 Current Directions

The key implication in the adoption of an AOSE approach is that the design and development of software systems cannot rely on conceptual tools and methodologies which were conceived for a totally different (old) paradigm. It is still possible to develop a complex distributed system in terms of objects and client-server interactions, but such a choice appears odd and complicated when the system is a MAS or it can be assimilated to a MAS. Rather, a brand new set of conceptual and practical tools – specifically suited to the abstractions of agent-based computing – is needed to facilitate, promote, and support the development of MASs, and to fulfil the great general-purpose potential of agent-based computing.

Researchers in the area of agent-based computing have recognised the above needs, and a considerable research effort is now being focused on the above topics. It is not in the scope of this paper to survey all relevant work in the above areas of AOSE. A number of excellent and extensive articles have been written to this purpose [5], and we refer the interested reader to them. Still, a short summary of the current mainstream research directions is worth reporting.

- **Agent modelling:** Novel formal and practical approaches to component modelling are required, to deal with autonomy, pro-activity, and re-activity. A variety of agent architectures are being investigated, each of which is suitable to model different types of agents or specific aspects of an agent: purely reactive agents, logic agents, agents based on beliefs-desire and intentions. Overall, these researches have so far notably clarified the essence of agency and its different facets.

- **MAS Architectures:** As it is necessary to develop new ways of modelling the components of a MAS, so it is necessary to develop new ways of modelling a MAS in its whole. Detaching from traditional functional-oriented perspectives, a variety of approaches are being investigated to model MASs. In particular, approaches inspired by societal or organisational metaphors [15][7] are the subject of the majority of researches and are already showing the specific suitability of these metaphors in different application areas.

- **MAS Methodologies:** Traditional methodologies of software development, navigating engineers from analysis to design and development, must be tuned to match the abstractions of agent-oriented computing. To this end, a variety of novel methodologies to discipline and support the development process of a MAS have been defined in the past few years [5][15], clarifying the various sets of abstractions that arise during MAS development and the duties and responsibilities of software engineers in that process.

- **Notation Techniques:** Specific notation techniques to express the outcome of the various phases of a MAS development process need development, because traditional object- and component-oriented notation techniques cannot easily apply. In this context, the AUM (Agent Unified Modelling Language) proposal [1], extending the standard UML toward agent-oriented systems, is the subject of a great deal of research and is rapidly becoming a de facto standard.

- **MAS Infrastructures:** To support the development and execution of MASs, novel tools and novel software infrastructures are needed. In this context, various tools are being proposed to transform standard MAS specifications (i.e., AUML specifications) into actual agent code [12], and a variety of middleware infrastructures have been deployed to provide proper services to support the execution of distributed MASs [4].

Clearly, all the above works are contributing to increase the acceptance and the practical usability of the paradigm. Nevertheless, the number of challenging research problems to be solved and the number of potentially interesting research directions is much larger than may appear from the above list.

3 Open Directions

Without being exhaustive, we try in the following to sketch a few key research directions that we think deserve further work to promote the acceptance and the diffusion of agent-based software engineering.

3.1 Assessing the Advantages of Agents in Software Engineering

Despite the current enthusiasm, the potentials of the paradigm need to be supported by stronger quantitative arguments. Software engineers who have to spend money and man-months in the development of a complex software system will be hardly be convinced to adopt a new paradigm (and pay the overhead involved in such a process) simply because it is conceptually elegant and because researchers claims it is suitable to modern ICT scenarios [9]. They will want evidence of the fact that this will help them save money and resources. We emphasise that we are not referring here to the advantages that agents could bring to software systems, but to the advantages that agents could bring to the process of developing software systems.

It is our opinion that much work is needed in the direction of evaluating (in a quantitative more than in a qualitative way) the agent-based paradigm and the associated methodologies, to assess their actual advantages over existing paradigms in software analysis, design, and maintenance.

3.2 Evaluating Non-Standard Processes

The definition of agent-specific methodologies is definitely one of the most explored topics in AOSE, and a large number of AOSE methodologies has been proposed in the literature. However, what characterises most of the methodologies proposed so far is that they assume a very traditional cascade model (from analysis, to design, implementation, and maintenance). But are we sure that the traditional software process model has to apply to MAS too? How can the abstractions of agent-based computing possibly impact on the very way one should approach the building of a MAS? In a world of dynamic and complex software systems, do concepts such as requirements engineering, analysis, design, implementation, and maintenance still apply in the traditional way?

We, as yet have no answer to the above questions. Still, we think that scientists working in the area will really address themselves to this problem, and possibly produce novel sof-
ware process models more suited to agent-based computing and (hopefully) more effective than traditional ones.

3.3 Identifying Suitable Agent-Specific Notations

A new paradigm will be more swiftly adopted by software engineers if adoption involves minimal effort, i.e., by letting them exploit as much as possible of the knowledge they already have, and by minimising the need to acquire new knowledge and new ways of thinking. For this reason (and also because the same considerations apply to scientists), a large amount of research effort is being spent in the area of AOSE to exploit and extend traditional (e.g., object-oriented) notation and modelling techniques for use in the context of MASs.

In particular, as we have mentioned above, agent-oriented extensions to UML (AUML) are the current subject of a great deal of research [1]. These extensions, focusing on specific aspects of the agent-paradigm with a set of intuitive and largely familiar diagrams, will be of great use in facilitating acceptance of the paradigm [9]. However, despite the current enthusiasm, we are far from convinced that AUML is the ultimate answer. In our opinion, the complexity, dynamics, and situated nature of modern software systems cannot effectively deal with notions and modelling techniques originally conceived for static and non-situated software systems. A great challenge in the area of AOSE will be to identify brand new notations and modelling techniques, conceived from scratch to suit the specific characteristics of MASs.

3.4 Engineering Large Scale MASs

Most current research in agent-based software engineering (as well as in mainstream software engineering) deals with systems in which a limited number (e.g., from a few units to a hundred) of agents have to be defined to interact toward the achievement of a specific application goal. Thus, current approaches typically keep and enforce a strict control and understanding over each and every component of the system, where engineers can detail each agent they develop, the mechanisms underlying each inter-agent interaction, and each interaction of the agents with their environment.

However, software development is more and more having to deal with understanding and controlling the behaviour of huge software systems with a very large number of interacting agents (e.g., from several hundreds to billions, as in the case of agent marketplaces, P2P – Peer-to-Peer –, and Grid systems), possibly distributed over a decentralised network and immersed in a dynamic and uncontrollable operational environment. Here, due to the complex and decentralised nature of the systems subject of the engineering work, it is not possible to exert a strict control over each agent and each interaction nor to observe at a reasonable and manageable level of detail the individual components of the system. Rather, it is the collective behaviour of the system that matters, and macro-level approaches to software engineering are required. That is, approaches that can abstract from the fact that a global system is made up of possibly individually deployed sub-systems and agents and that, disregarding the behaviour of individual agents or individual sub-systems, focus on the behaviour of the system as a whole.

A large amount of experimental and simulation research, aimed at understanding what the behaviour of very large size (possibly world wide) MASs will be, is already available, but little is yet known about the appearance of the discipline of large-scale MASs engineering.

3.4.1 Sociology, Biology, and Beyond

The development of an engineering discipline for large-scale MASs cannot abstract itself from the study of the behaviour of complex large-scale systems that can be found in nature. Such studies have already been (and will continue to be) of great use for the building of specific classes of MASs (consider, e.g., ant-based routing systems). In addition, they will very likely be able to provide some general insights into how large-scale MASs work and how they can be effectively engineering controlled at the macro-level (i.e., without control on the behaviour of each and every component). Clearly, this is a necessary pre-condition of the definition of a discipline of large-scale MASs engineering. However, for these insights to be achieved in the near future, (i) a larger variety of phenomena will have to be explored and (ii) novel formal modelling approaches will have to be produced.

With regard to the first point, there are a number of interesting physical and biological phenomena that are currently underestimated in the area of AOSE but which have the potential to be effectively exploited in the building of robust and adaptive MASs. For instance, the emergence of regular spatial patterns observed in a variety of physical systems [13] could be possibly exploited to implement novel and very effective strategies for distributed coordination in large-scale MASs. As another example, the patterns of distribution of specimen populations, as deriving from their specific interaction mechanisms [14], could be of use toward the definition of adaptive MAS organisations and of effective strategies for dynamic division of labour.

With regard to the latter aspect, there is a need for a common modelling of the different types of natural/biological/physical phenomena that are being exploited in large-scale MASs. Its absence prevents comparison of heterogeneous approaches, as it is impossible to document the performed experiences for the sake of verifiability, reproducibility and reusability. The availability of a common modelling language would make it possible to build a catalogue of reusable patterns of global MAS behaviour. Last, but not least, devising a common modelling language would bring us very close to the identification of the universal laws of MASs (if any exist) and, consequently, of a general-purpose discipline of large-scale MASs engineering.

3.4.2 Identifying the Boundaries of a System

As most software systems – even if developed in isolation – are likely to be deployed as parts of larger open systems, the problem of identifying the boundaries between systems arises. In an open world where agents can get born and die at any time, where mobile agents will roam across network domains according to their own plans, and where the task to be accom-
plished by a software system can be delegated to external components (e.g., middleware services, mediators, brokers), the concept of a system boundary weakens, and the very notion of system blurs. In other words, it may be rather unclear what an engineer should consider as part of his ‘own’ system (and thus include it in its work) and what should instead be simply considered as something irrelevant to his work.

The common practice is to wrap any external entity into an agent that becomes part of the internal system, thus encapsulating any boundary effect into a set of well-identified wrapper agents. This does not solve the problem, it merely hides it inside some agents. Also, this practice is completely useless when, as often happens, most of the agents of a system have to interact with some parts of the external world. A more effective solution is to enforce all interactions in a MAS, both internal and external, to take place via some shared interaction infrastructure [8] (e.g., a blackboard or a tuple space [10][3]). In this case, the shared interaction infrastructure acts as a virtual space with well-defined boundaries and well-identifiable dynamic of interactions across boundaries. Unfortunately, this solution dramatically clashes with the common perspective of inter-agent interactions, in which agents are typically assumed to be able to directly talk with each other in a peer-to-peer way, without the mediation of any shared interaction space.

Possibly, a more general direction (abstracting from the actual presence of a shared infrastructure) could be that of identifying, and integrating in AOSE methodologies, usable guidelines for the identification of boundaries, and of electing boundary conditions (that is, the modelling of the characteristics and dynamics of the interactions across boundaries) to a primary abstraction in open MAS development [7]. Unfortunately, what these guidelines should be and how boundary conditions could be modelled we just do not know at the moment.

4 Conclusions

The area of AOSE is definitely at a very early stage. While an increasing number of research groups get involved in this topic to explore the implications of developing complex software systems according to the agent-oriented paradigm, a number of fascinating challenges are still open to investigation. As we have tried to overview in this paper, AOSE research cannot simply be directed to the definition of new agent-specific development methodology and to the adaptation of existing notations. Issues such as the definition of novel methodologies and tools, and the identification of appropriate engineering approaches to deal with large scale systems and with systems with no clear boundary are going to offer a wide field for future researchers.

References


Veriﬁcation and Validation Techniques for Multi-Agent Systems

Rubén Fuentes-Fernández, Jorge J. Gómez-Sanz, and Juan Pavón-Mestras

Software Engineering aims at producing software systems built correctly and whose functionality matches initial requirements. These features have motivated research in veriﬁcation and validation methods. Agent Oriented Software Engineering (AOSE) pursues producing Multi-Agent Systems (MAS) with similar quality measures as those systems produced by conventional software engineering. However, there are important conceptual differences between conventional and multi-agent systems that are motivating new methods for veriﬁcation and validation of this new kind of system. This paper gives an overview of the conceptual differences and presents some of these methods brieﬂy.

Keywords: Agent Oriented Software Engineering, MAS Survey, Multi-Agent Systems, Software Engineering, Software Systems.

1 Introduction

A good software product should satisfy customer requirements as well as showing correct behaviour, such as not hanging in the middle of an operation and not producing undesirable outputs. Customer requirements satisfaction and basic functionality can be checked with different types of tests (black box and white box) either run manually by beta testers or automatically by test batteries. However, verifying other properties, such as being deadlock-free or exercising fairness, is not that easy.

In general, the techniques allowing us to check that the application has the required functionality and that this functionality is correctly built are grouped together under the titles of veriﬁcation and validation. Validation tests satisfaction of customer requirements; veriﬁcation deals with correctness of the system.

The abstractions used in the modelling stage of development inﬂuence greatly the kind of veriﬁcation that can be applied [8]. The agent-oriented paradigm proposes modelling software systems with social and intentional abstractions; an agent and a Multi-Agent System (MAS). Conventional computations are understood within this as an interaction among different agents or MAS [11]. This conceptual framework enables the description of some aspects of the system and its environment more coherently, homogeneously and from a new level of abstraction.

These new concepts and properties, which arise from using this agent paradigm, imply constructing new veriﬁcation and validation approaches, different from conventional software engineering ones. In [14] there are some remarks about these differences: conventional step-wise reﬁnement techniques that do not support agent features; problems in ﬁnding concrete computational interpretations of some agent concepts (such as behaviours deﬁned with possible-worlds logic); excessive simpliﬁcation of the systems in order to reduce proof complexities. Other authors [13] criticise the traditional handling of social features in systems produced through conventional software engineering. These social features, which are key elements in the agent paradigm, are related to organisational,
cognitive, developmental, evolutionary and motivational concepts. However, many approaches try to work with these properties using techniques that were conceived for other purposes, without taking advantage of specific agent characteristics.

In the previous context, verification and validation of MAS have not just imported techniques from other paradigms, but have also created new approaches to solve this problem. The presentation of these specific techniques follows the structure in [4]: conventional, semi-formal, formal, and hybrid. The categorisation groups refer to the formalism of the specification language used and the associated methods. The last group of hybrid approaches, are derived from the combination of the previous ones, to increase usability and analysis power.

The presentation of these techniques makes use of a case study example, which considers the evolution of a traditional bookselling company and its university student customers using an e-business approach (Jual Møller Bokhandel A/S [3]). The system construction is to be based on the agent paradigm. Besides the system itself, in this chapter we are especially interested in the resulting specification of the system. This is to be examined for non conventional properties: if there are actors in the system with competitive goals; how different design decisions are evaluated according to agent features; or how different social organisations affect to the interaction among agents.

Each one of the four previous groups has a dedicated section in this paper, referencing the mentioned case study. At the end of the paper, readers will find a brief discussion about the evolution of verification and validation in the context of Agent Oriented Software Engineering (AOSE.)

2 Conventional Methodologies

This group embraces those methodologies that make their specifications in natural language, complemented with arbitrary diagrams or images. The processes in these methodologies tend as well to be described in natural language texts, whose accuracy depends on the skill of the developer. As a representative example of these techniques, we focus on the application of Ethnography to software development.

Ethnography approaches are used to verify and/or validate a set of design decisions using the real context in which they were taken. These analyses gather an initial model and handle their implications in the environment of the users. Subsequently a detailed report is created of the modified workflows with the new system that can be compared with the expected behaviour. Among the techniques used in these analyses are the observation of actors in their usual environment, document research, interviews and questionnaires.

As an example of the kind of descriptions that Ethnography handles in development tasks, readers can find the following example, adapted from [6]: this interaction took place between two users when problems emerged in using the computer system of a neuromagnetometer, which is a device to study the brain activity. In the transcript, symbol (– – –) refers to inaudible speech. Key notifications of the system software appearing on the computer screen are transcribed in italics.

[... ] 50 User 1: Data are correct. Let’s try. (– – –)
51 User 2: Okay. I try now (On the screen: HTTP Transmission failure)
52 User 2 (worriedly): There is still something in that network configuration — [...]

Despite the shortness of the excerpt, it can be seen that main drawbacks of Ethnography for verification come from its very same virtues as a tool for the study and description of organisations. Firstly, results of the studies are not presented in a format suitable for reuse in development. They are speech-oriented presentations that do not directly address developers’ needs and do not describe plainly the design issues. Secondly, decisions about the process to follow lie with the individual and are taken without the help of structured guidelines.

Because of their languages, informal methods are difficult to automate. One of the few available automation options is the use of techniques for natural language processing. The aim of these techniques is to obtain knowledge from textual sources and make some basic inferences [9]. Nevertheless, these techniques also have their own limitations. Those that scale better, such as probabilistic ones, need a large body of pre-tagged text for training. Besides, they have difficulties in analysing the discourse or grasping notions of meaning and relevance. Other techniques, such as those that use taxonomies, demand great human effort, but improve the understanding of the meaning.

As a conclusion of this overview of conventional techniques, we should underline that these approaches are the easiest to learn because they use, as their language primitives, common symbols of everyday communication. Despite this, the use of these languages does not guarantee correct understanding in the development team, comprising stakeholders, end-users, and developers. The considered abstractions can be unknown or have a different meaning for some members of the group. In the ethnographic example, some of the problems emerged due to the different awareness of the situation by users and developers.

Another drawback of these approaches is that their use has to be manual or, if some scalable automated process is considered, is necessarily vague because the inherent ambiguity of their textual sources.

3 Semi-formal Methodologies

Semi-formal methods use some kind of semi-formal notation, mainly diagrams, to describe the information about their artefacts. These notations have well-defined syntaxes with a number of predefined primitives, and allow the use of natural language only in a limited manner. The basic vocabulary usually comes with processes to elicit and transform the information and then describe it within the target notation. Their reasoning resources about that information are quite limited. Many times they are just manual processes or ad-hoc solutions specifically conceived for the property to be checked. Both the processes and the reasoning mechanisms for verification are mainly described in terms of natural language.

The first example in this group of methodologies is INGENIAS [9]. INGENIAS is based on the use of meta-
models that give the primitives and syntactic properties of models. To obtain the information for those models, INGENIAS describes a complete development lifecycle that includes rules to identify and transform design entities.

INGLENIAS is an example of semi-formal methodology with ad-hoc solutions to verify properties. Currently, its checks are described with natural language and programmed as components of its supporting tool – the INGENIAS Development Kit (IDK).

The verification of models for contradictions would imply programming a new component for the IDK. Developers would have to decide what a contradiction is and how they could describe it in Java. Then, that component would traverse the models, gathering goals and their dependencies and checking if they constitute a contradiction.

The second example in this section is CREWS-SAVRE [7], which is a tool automatically to generate exhaustive scenarios out of a domain taxonomy. Based on that taxonomy, CREWS-SAVRE provides a systematic guide to build scenarios covering alternative courses of action, exceptions and errors. This approach is an example of automation based on the use of knowledge repositories about certain aspects of development.

For our example, deciding with CREWS-SAVRE between several alternatives for design requires the development of a way to describe them in the taxonomy. After that, developers would study the guides for scenario generation in order to decide how to change them for the modified taxonomy. Finally, users would have to validate manually the new scenarios according to their own criteria.

To summarise this section, we can remark that semi-formal techniques have greater constraints than conventional ones in the expressive power of their languages. As advantages, these constraints allow them to introduce better guidance for processes and more automated techniques. In any case, if developers want to benefit from these advantages, they have to carry out in advance challenging work to obtain the required knowledge and build the required tools. Whatever the solutions are, these proposals always try to keep a high level of usability for their languages.

As the cases of the INGENIAS models or the taxonomy and guides of CREWS-SAVRE show, the approach with concrete solutions for every property eases the checking of isolated simple properties. However, advanced verifications or those involving multiple properties are difficult to verify because they imply a manual combination of checks. Besides, customers and developers are forced to operationalise their properties in a complex environment, where mistakes can be hard to detect.

4 Formal Methodologies

These techniques represent information with formalisms, i.e. non-ambiguous languages that are based on mathematical principles. Examples of these languages are Z (a set based representation), B method (state machines based representations) or logics. Each language is associated with specific verification and validation techniques.

In the agent domain, the first example deserving mention is DESIRE [2], a design and specification framework that permits building of agents based on the recursive composition of interconnected tasks. DESIRE describes agents and the MAS itself as networks of tasks organised in a hierarchy. The interaction and co-ordination among agents is specified as interchanges of pieces of information and control dependencies. The verification mechanisms base on temporal logics.

To model our problem in DESIRE, we would start with the analysis of tasks that users and the system perform, as well as their associated goals. Afterwards, properties to be verified would be represented in temporal logic: what is a conflict among goals or how to choose among design alternatives. Checking properties consists of demonstrating that these are satisfied in this concrete problem using the DESIRE representation of the system.

Another significant example is the work of IMPRESS [12]. It shows how to integrate different techniques into a common formal base. IMPRESS originates in solving several problems that affect the validation tests for specifications, such as not indicating the way to recover from an error situation or not guiding the acquisition of new knowledge, when the failures are originated in the incompleteness of the information.

To solve previous problems, conventional validation techniques were mixed with machine learning techniques. The idea was to formalise requirements with MSFOL (Many Sorted First Order Logic) and consider them as an imperfect theory. Users’ use cases would be examples and counter-examples that the theory should take into account and, if necessary, cause the adaptation of the theory. This evolutionary approach resulted in a theory improving itself by a machine learning method.

After this brief review of the examples, we can see that the advantage of these approaches over the previous ones is that they allow the drawing of more conclusions from specifications in an automated way [14]. The temporal logic of DESIRE allows proving complex properties of the system and the domain, or the execution and animation of specifications; IMPRESS facilitates the collaboration of users in the refinement of the specifications and provides a semi-automatic correction method for incorrect specifications.

These approaches have several drawbacks as well. Firstly, they assume a fixed architecture in the agents, e.g. a task based architecture in DESIRE. This approach produces some benefits, but it also implies providing more support through the process (to handle the correct building of the network of tasks, for instance) and adopting the task design philosophy to express agents and MAS, which is not trivial. Secondly, the specification needs a high level of detail in order to be used for the verification and validation techniques. The higher the level of detail, the more accurate it is, but it also takes more effort to produce the specification. Thirdly, formal notations do not ensure that their readers understand correctly the meaning of the results, as seen with IMPRESS. This happens because many of the existing formalisms require skills above those of an average developer.

5 Hybrid Methodologies

This kind of approach mixes some of the previous ones.
Usually two languages, a semi-formal and a formal one, are applied at the same time. The semi-formal language is used as a communication tool. The formal one is used when a fine-grained specification or a sophisticated computation over models is needed.

As example of this approach, we introduce Tropos [5]. It is an agent based software methodology where the requirements gathering stage determines what processes to apply in each development cycle stage. The concepts proposed by Tropos are applied in three kinds of representations: diagrams, partially formal annotations and formal specifications. The Tropos formal language, Formal Tropos, analyses the primitives of the graphic language used for diagram construction and adds them as temporal logic constructs. Therefore, a diagram-based specification can be translated to the formal language. Diagrams enable a quick visual modelling of the system with an intuitive vocabulary. On the other side, the translation of the diagrams to the formal language, complemented with additional specifications, offers the possibility of verifying the specifications with formal methods.

The specification of the case study with Tropos would be similar to what Sections 3 and 4 show. Tropos would start with a diagram modelling stage for the social properties previously remarked as actor goals and their dependencies. Final verification would require using Formal Tropos.

Another example of these approaches is IF (Intermediate Format) [1], which integrates different techniques of verification to give a better solution to each particular problem. IF uses an intermediate language for the interchange of specifications and results among different distributed system validation tools. In this sense, it does not support any concrete methodology nor does it supply guidelines for the checking process. It is a support toolbox.

As a conclusion, the hybrid approaches pursue maintaining the usability of the semi-formal approaches with a level of analysis close to formal when required. The main criticism is the nature of the properties to be checked. These come from the client requirements or the knowledge and experience of the development team. Due to this fact, these properties influence the specifications in order to make them consistent with software engineering practices and the needs identified by the client. There are no analysis elements that do not belong to the information already managed by the development team.

6 Conclusions

This survey on verification and validation of multi-agent systems has reviewed the underlying modelling language and techniques for these activities, and has considered the results obtained with these tools and the cost of their use, both to learn them and to specify with them.

As a general statement we can say that those techniques using formalisms have more powerful tools for analysis and are less ambiguous than others, but their related costs are also higher. To counter this disadvantage, the solution is to have mechanisms that improve their usability. This can be achieved in most cases through abstraction and encapsulation techniques. So, interfaces can be provided that allow the creation of the main part of the specifications without worrying about the details of the core formalism. The need for working with the original language only arises when the development needs fine-grained specifications or sophisticated reasoning mechanisms.

For property verification, formal approaches usually support within the same formalism the specification and the inference mechanisms to check properties. On the other hand, informal and semi-formal approaches are limited to manual verifications, maybe described in natural language. Some examples automatically check properties with tools, as it is the case with the INGENIAS Development Kit (IDK). The problem for these methods is that properties are already prescribed in the environment or, if they can be added or modified, the developer needs some facilities to be able to program new verifications.

Finally, we have realised that the reviewed techniques do not completely take into account some specific features of the agent paradigm. The verification of properties about agent organisation, their motivation or their learning capabilities only have ad-hoc solutions, generally coming from the adaptation of techniques initially conceived to check other kinds of properties. This makes it difficult to take full advantage of the potential that these properties show, for instance, in the social sciences.

Acknowledgements

This work has been funded by the Spanish Ministry of Science and Technology, under grant TIC2002-04516-C03-03.

References


Applying the Tropos Methodology for Analysing Web Services Requirements and Reasoning about Qualities of Services

Marco Aiello and Paolo Giorgini

The shift in software engineering from the design, implementation and management of isolated software elements towards a network of autonomous interoperable service is calling for a shift in the way software is designed. We propose the use of the agent-oriented methodology Tropos for the analysis of Web service requirements. We show how the Tropos methodology adapts to the case of Web services and in particular how it can be used to model quality of service requirements. We base the investigation on a representative case study in the retail industry.

Keywords: Agent-oriented Methodology, Software Engineering, Quality of Service, Web Services.

1 Introduction

The opportunities offered by the growth of the Internet in terms of networking infrastructure, open standards and reach of users, are focusing research and industrial interests on application areas such as electronic commerce, enterprise resource planning, supply-chain management, and peer-to-peer computing, to name the most prominent ones. This is deeply and irreversibly changing our views of software and, in particular, software engineering. Interoperability and scalability play a fundamental role in the development and management of software as nowadays a piece of software cannot be thought in isolation, but rather, as an element of a network of interacting software elements. Continuous evolution to meet changing and new requirements is becoming an essential feature of software. In addition, software must be robust and autonomous, capable of serving end users with a minimum of overheads and interference. Software is thus becoming more and more a service offered to a human user or to another software element rather than an isolated application running on a specific machine for a specific predefined requirement. This is the view of software as a ‘service’ well conceptualized by the service-oriented computing paradigm [16].

The most prominent example of the service-oriented computing paradigm is to be found on the Internet, where the set of standard interfaces for the interaction of software elements is well-known as Web services. Web services are a set of standardized interfaces for the description, discovery, invocation, composition and orchestration of independent loosely-coupled software elements residing on the Internet. As software is changing, one of the challenges is to find appropriate concepts, tools and techniques to design, engineer and manage software. Traditional software engineering methods may prove to be cumbersome or not capable of capturing the full potential of the service-oriented paradigm. In [6], UML is used to design business processes that manage the execution and interaction with various independent Web services. Aspect-oriented programming is investigated in [10] for designing Web service based electronic utilities, i.e., distributed applications. But all these approaches lack fundamental features of Web services, that is, the autonomy of services, the need to model services at a high level of abstraction in terms of what a Web services goal is rather than all its atomic functionalities, and the need for runtime support for changing execution environments.

Agent-oriented software development methodologies are gaining popularity over traditional software development...
approaches [11][7]. After all, agent-based architectures do provide for an open, evolving architecture that can change at runtime to exploit the services of new agents, or replace under-performing ones. In addition, software agents can, in principle, cope with unforeseen circumstances because their architecture includes goals along with a planning capability for meeting them.

Mostly, software is thought of in terms of its functional behaviour, i.e., what the software does. But this does not completely describe the software’s behaviour. Non-functional properties, such as the average execution time, are important elements to determine the usability and utility of a software product. With the term Quality of Service (QoS, for short), we refer to the non-functional properties of a software service. In the context of Web services, QoS is a critical task for a number of reasons: first, autonomous services depend on one another for their functioning and they need to be aware of the QoS of the collaborating services; second, services may compete with one another and a service requester may decide on a service based on its QoS properties; third, a service provider may offer the same function with differentiated QoS, for instance at different prices, and must therefore publicize the different qualities of the same function. There is no consensus on what the qualities are that fall in the QoS of a Web service. The traditional view inherited from the networking community places only performance and availability in the set of QoS, but other properties are also relevant such as accessibility, integrity, reliability, regulatory and security [14]. Some authors assume that any custom parameter that can be modelled as a non-functional property of a service may be considered as an element of the QoS [18, 19, 15]. In this paper, we consider a wide spectrum of QoS properties, such as performance, cost, reliability and security, and we introduce a framework that is flexible and open to any user-defined quality as QoS.

We propose the use of the agent-oriented methodology Tropos [3][1] for the analysis of Web service requirements. The goal is to model software deployed using Web services at a high abstraction level by using agent-oriented techniques. In this approach, we do not model every individual Web service as one agent, but rather model the whole set of interacting services as a multi-agent system, where different dependent strong and soft goals coexist. Thus, using Tropos we consider Web services not simply on the basis of their functional interfaces, but rather on how each service impacts shared goals of a software system, and which role each service has in such an entire system. These considerations are particularly useful to model complex systems through their life-cycle, that is, from the early requirements analysis down to the actual implementation. In particular, the methodology emphasizes early requirements analysis, the phase that precedes the prescriptive requirements

Figure 1: Actor Diagram for the Online Retail Store Example.
specification. In this paper we use this feature of Tropos to model quality of services of systems based on Web services as QoS are typically high level characteristics not described at the level of Web service individual interfaces.

The paper is structured as follows: Section 2 introduces the Tropos methodology, where concepts, modelling and analysis techniques are presented through a representative case study. In Section 3 we propose a goal analysis framework for qualitative and quantitative reasoning about QoS. Section 4 discusses how the framework relates to existing Web service standards. Concluding remarks are summarized in Section 5.

2 Requirements Analysis with Tropos

Tropos rests on the idea of using requirements modelling concepts to build a model of the system-to-be within its operational environment. This model is incrementally refined and extended, providing a common interface to the various software development activities. The model also serves as a basis for documentation and evolution of the software system.

Tropos is an agent-oriented methodology in which agent’s related notions (such as agent, goal, belief, task, and social dependency) [4] are used in the whole development software process, from early phases of the requirements analysis. Requirements analysis in Tropos consists of two phases: *Early Requirements* and *Late Requirements* analysis. Early requirements is concerned with understanding the organizational context within which the system-to-be will eventually function. Late requirements analysis, on the other hand, is concerned with a definition of the functional and non-functional requirements of the system-to-be.

Figure 2: Part of the Rationale Diagram for the Retailer System.
During early requirements analysis, the requirements engineer identifies the domain stakeholders and models them as social actors, who depend on one another for goals to be fulfilled, tasks to be performed, and resources to be furnished. Through these dependencies, one can answer why questions, besides what and how, regarding system functionality. Answers to why questions ultimately link system functionality to stakeholder needs, preferences and objectives. Actor diagrams and rationale diagrams are used in this phase.

Figure 1 shows the actor diagram for an online retail store example. This example is an extended and revised version of the example introduced in [12]. The diagrams present the principal stakeholders and their interests. The Customer actor has the goal to buy products and the softgoal to buy at lowest prices. It depends on the Retailer actor for having good services and on the Bank for using bank services. The Retailer actor has the softgoal of maximizing profit and depends on the Bank for the bank services, on the CreditAuthority to validate the customers' ability to pay, and on the Direct Supply Vendor to ship products to the customers. The Direct Supply Vendor depends on the Retailer for products offering and on the Transport Centre to ship goods.

Actor diagrams are extended during early requirements analysis by incrementally adding more specific actor dependencies which come out from a means-end analysis of each goal. This analysis is specified using rationale diagrams, which appear as a balloon within which goals of a specific actor are analysed and dependencies with other actors are established. Goals are decomposed into subgoals and positive/negative contributions of subgoals to goals are specified (see Figure 2 as an example.)

During late requirements analysis, the conceptual model developed during early requirements is extended to include the system-to-be as a new actor, along with dependencies between this actor and others in its environment. These dependencies define functional and non-functional requirements for the system-to-be. Actor diagrams and rationale diagrams are used also in this phase.

Figure 2 shows part of the rationale diagram for the retailer system. Basically, this analysis extends the goal analysis for the

---

1. Softgoals represent vaguely defined goals, with no clear-cut criteria for their fulfilment.
Retailer actor. The extension includes the analysis of the services (hexagons) that can be adopted in order to satisfy the Retailer System’s goals and how such services impact on the qualities of the system, namely the softgoals security, reliability, and performances. For the sake of simplicity, the diagram reports only some of the possible contribution links between services and qualities.

3 Reasoning about Qualities

The adoption of specific Web services can have different consequences on the qualities of the software system. In order to reason about these effects, we propose to adapt and use the Tropos goal analysis techniques presented in [8][17].

The analysis starts from the goal models developed in the late requirements phase, and in particular focuses on the models developed for the system we want to develop (the Retailer System in our example). The goal model consists of a set of nodes (goals and services/tasks) and relations over them, including the n-ary relations AND, OR and the binary relations (contribution links) + and −, + (−) relationship is used to represent that the satisfaction of a goal contributes to the satisfaction (denial) of another goal.

Figure 3 presents part of the goal model for the Retailer actor extended with the analysis concerning the adoption of the adopted services and their impact on the qualities of the system. The contribution links between services and privacy quality are both qualitative and quantitative. Qualitative links are used for qualitative reasoning, while quantitative links for quantitative reasoning.

3.1 Forward Reasoning.

An algorithm and its implementation for forward reasoning has been presented in [8]. Given a goal graph, the user assigns some initial values to some goals (typically leaf goals), then these values are forward propagated to all other goals according to the rules above described in [8]. The initial values can be qualitative (such as fully satisfied, partially satisfied, fully denied, and partially denied) or quantitative (probability for the satisfaction or the denial of a goal).

The user can then check the final values of the goals of interest (typically root goals), and see whether she/he is satisfied with these values. Basically, forward reasoning is used to observe the effects of the adoption of a set of Web services.

For the example in Figure 3, we might, for instance, be interested in finding the effects of adopting a set of services over the top goals increase return on investment and sell products as well as over the privacy softgoal Security. So for instance, suppose we decide to use the following services (i.e., we assign FS (Fully Satisfied) labels to them): eShop service, day definite (by DHL), and PayPal. With these services Privacy will be partially satisfied (PS), sell products will be fully satisfied (FS), and increase return on investment will be partially denied (PD). Of course, we could suppose that expand the markets and then increase return on investment will also be partially satisfied. In this case we have a conflict, that is, we have evidence both for the satisfaction and the denial of the goal.

Similarly, we can perform qualitative reasoning. Suppose to assign to eShop service, day definite (by DHL), and PayPal the value S=1 (i.e., probability of Satisfaction equal to 1), that is, we decide to adopt them. As consequence we obtain that Privacy will assume S= 0.4 and D=0.5 (i.e., probability of denial equal to 0.5) for denial, whereas sell products will assume S=1.0. This means we can satisfy our top goal but we cannot say much about Privacy (we have a conflict). Note that we have not considered the effects produced by the adopted services on the increase return on investment goal.

3.2 Backward Reasoning

An algorithm and its implementation of backward reasoning has been presented in [17,5]. In particular, the implemented tool solves the following two problems: (1) find an initial assignment of labels to leaf goals which satisfies a desired final status of root goals by upward value propagation, while respecting some given constraints; and (2) find a minimum cost assignment of labels to leaf goals which satisfies root goals. Basically, backward reasoning is used to find a possible set of services to be adopted in order to satisfy our top goals at the minimum cost.

In our example, we might be interested in finding a set of services (at the minimum cost) that satisfy our top goal sell products and Privacy softgoals. For instance, suppose we want to satisfy sell products and at least partially satisfy Privacy softgoal. The software gives no solution for the full satisfaction of softgoal Privacy, but it produces several solutions for its partial satisfaction. Fixing the same cost for all services, one of these solutions consists of the following services: same day (by DHL), PayPal, and internal email service.

Analogously for the quantitative reasoning, we might be interested in finding the minimal cost set of services that satisfy totally the top goal sell products (S=1.0) and partially the Privacy softgoals, Say S=0.7. The software produces a set of solutions that satisfy such a request, i.e., a set of services that, if adopted (S=1.0), produce the desired results over sell products and Privacy.

4 Quality-aware Web Services

The qualities of the retailer system we have presented so far are performance, reliability and security. These are custom QoS measures for software and, in particular, for services, but are not the only possible ones. Availability, integrity, and regulatory conformance are other QoS that may need to be modelled, furthermore, other parameters specific to the application at hand may need modelling. The framework proposed does not commit to any specific quality, but rather gives freedom of choice to the designer.

This freedom must be reflected at the service level, in other words, services must be able to describe their qualities and have shared vocabulary of service qualities. Standard service description languages such as WSDL (Web Services Description Language) lack the necessary constructs to address this issue. Two approaches are possible: on the one hand, one can extend WSDL with ports for the description of quality properties of the services (such as in [9]); on the other hand, one could
complement WSDL interfaces with ancillary documentation for the description of quality of service characteristics of the service.

In [15] a symmetric model based on constraint satisfaction techniques is used to verify QoS desires coming from the requester. In [19], an XML-based language (eXtensible Markup Language) used for negotiating QoS values among service requester and provider is presented. A semantic web approach in which services are searched based on quality of service attributes semantically tagged is presented in [18]. A predictive QoS model for workflows involving QoS properties is proposed in [2]. In addition, the industry has proposed a number of standards to this end: IBM Web Service Level Agreement (WSLA) and HP’s Web Service Management Language (WSML) are examples of languages used to describe quality metrics of services, [13]. What is missing is a framework to design composition and orchestrations of services with desired global QoS requirements and, dually, to analyse global QoS properties of Web service compositions.

The framework we propose is independent from the choice made on whether one extends WSDL or one uses an extra document for the description of service qualities such as WS-Policy. The only requirement is, naturally, that all the services implement the same infrastructure for service quality description and use a negotiated and agreed ontology for describing the qualities. Let us consider the Privacy quality of the Retailer System example by adapting WS-Policy to our framework. Suppose the PayPal service publishes the following policy:

```
01 <wsp:Policy xmlns:wsse="..."
   xmlns:wsp="...">
02 </wsp:Policy>
03 <wsp:ExactlyOne>
04 <wsse:SecurityToken wsp:Usage="wsp:Required" wsp:Preference="80">
05 <wsse:TokenType>wsse:SecureSocketLayerVersion2.0</wsse:TokenType>
06 </wsse:SecurityToken>
07 <wsse:SecurityToken wsp:Usage="wsp:Required" wsp:Preference="50">
08 <wsse:TokenType>wsse:SecureSocketLayerVersion3.1</wsse:TokenType>
09 </wsse:SecurityToken>
10 <wsse:SecurityToken wsp:Usage="wsp:Required" wsp:Preference="10">
11 <wsse:TokenType>wsse:SecureSocketLayerVersion3.0</wsse:TokenType>
12 </wsp:ExactlyOne>
13 </wsp:Policy>
```

In our framework, this policy would be interpreted as the fact that the PayPal is providing its services with three different qualities, the three SecurityTokens. At least one of these needs to be chosen, line 02. The different quality of service of these is represented by the wsp: Preference attribute on lines 03, 06, 09. This is interpreted as the fact that the first choice (line 04) gives a contribution to Privacy of 0.8, while the second (line 07) of 0.6 and the third (line 10) of 0.1.

5 Concluding Remarks

The shift in software engineering from the design, implementation and management of isolated software elements towards a network of autonomous interoperable service is motivating the investigation of new modelling and design techniques. We have proposed the use of the agent-oriented methodology Tropos for the analysis of Web service requirements. We have shown how the Tropos methodology adapts to the case of Web services and in particular how it can be used to model quality of service requirements. Forward reasoning and goal satisfaction have been proposed to design an architecture meeting given QoS requirements, but also to understand which QoS properties a system will have and how the various services influence the QoS parameters. We have based our investigation on a representative case study and have shown a wide range of non-functional properties to be captured by the framework we propose. Finally, we have shown how one can adapt existing Web service technologies to be included in the proposed framework.

A tool exists for the analysis of early requirements using Tropos. The extension of such a tool to deal with Web services and in particular with QoS requirements for Web services is under way.

A limitation of the proposed approach is that the contribution of the single service to a given element of the quality of services is independent from those of the other services, that is, the combined effect of different services to the same quality is not captured. Future investigation will be devoted to solving this issue by considering global interaction of quality features.

References


Developing a Multi-Agent System Using MaSE and JADE

Pedro Cuesta-Morales, Alma-Maria Gómez-Rodríguez, and Francisco J. Rodríguez-Martínez

This work examines the use of agent technology when dealing with distributed real world problems. By means of a case study, the system analysis and design, which was done using an agent-oriented methodology, is shown. The subsequent implementation with a specific development framework was mapped onto a Multi-Agent System (MAS). Particularly, MaSE (Multi-agent Systems Engineering) methodology and JADE (Java Agent DEvelopment Framework) were used for building a system which sets automatically a date and hour for a meeting among different users of the same organization.

Keywords: JADE, MaSE, Multi-Agent System, Negotiation.

1 Introduction

In recent years, agent-oriented development has become a new paradigm in Software Engineering [2]. The Agent concept constitutes a powerful abstraction tool when dealing with software development, making easier the construction of distributed, intelligent and robust systems [8].

Agents are usually defined by considering some characteristics that they must fulfil. These attributes include autonomy, reactivity, sociability and pro-activity, and also other issues like: rationality, mobility, etc. Generally speaking, the solution to complex problems cannot be approached using a single agent, but by means of a set of agents which interact in order to achieve the system goals [9][10].

A number of software applications have been built using an agent-oriented approach. Taking into account the experience provided, methodologies which guide the engineer in the development process and tools that facilitate the implementation are defined [6]. The main aim of this paper is to share the experience of authors in the development and design of a multi-agent system using a particular methodology and a specific development tool. Among all the methodologies currently available (Gaia, MaSE, Ingenias, Tropos, etc.) MaSE (Multi-agent Systems Engineering) was the one selected [3] mainly because it is domain-independent, it covers the whole development lifecycle, and it has a support tool: AgentTool [4]. In respect of implementation, JADE (Java Agent DEvelopment Framework) has been chosen [1]. The reasons taken into account to make this selection have been that because it complies with the requirements form FIPA (Foundation for Intelligent Physical Agents) [5], it is one of the more used platforms at this moment and it is thoroughly documented.

Agent technology is not suitable for any kind of problem, but it provides a good development method in situations like the following ones:

- When the knowledge is distributed.
- When negotiation, competition or cooperation capacity among several entities is needed.
- When the system must act autonomously, etc.

The problem selected incorporates several of the previously introduced characteristics. It is the development of an automatic meeting scheduler system, i.e. autonomous software, which does not need the continuous involvement of all users in order to fix a meeting. This problem also demands communication and negotiation among the entities which manage the users’ personal agenda.

The reminder of the paper is organized as follows. The second section describes the functional requirements of the system to be built. Next, the agent-oriented methodology used in the development is introduced and its application to the use...
case is shown. Sections five and six focus respectively on a
description of JADE and its utilization in the construction of
the meeting scheduler. Finally the conclusions achieved by the
experience are presented.

2 System Description

As it has been said before, the main aim of the system to
be constructed is to fix automatically meetings in accordance
with the personal agendas of users invited to such meetings. In
order to achieve this general goal, the system must provide
several functionalities; amongst others they are:

- **User management:** Users must register with the system and
can organize themselves in groups.
- **User’s personal agenda management:** User agenda will
reflect the available timetable and will include periodic and
non-periodic activities.
- **Meeting scheduling:** Any user may propose a meeting on a
particular date. According to it, all the users invited to the
meeting must revise their agenda and negotiate to decide a
suitable date and time.
- **Informing users:** Once a meeting has been fixed, it is neces-
sary to send users such information, and, in addition, to
update their agendas.

3 MaSE Methodology

Multi-agent Systems Engineering (MaSE) methodology
[3] is a complete methodology for developing heterogeneous
multi-agent systems. MaSE covers the complete lifecycle of
the system, from the analysis to the design utilizing a number
of graphically based models (uses UML – Unified Modelling
Language – components). The models are transformed into
diagrams in order to describe system agents, their communica-
tions, and the internal structure of each agent detailed in depth.

MaSE is supported by a software engineering tool called
AgentTool [4]. AgentTool allows the designer to formally spec-
cify all the MaSE’s models. It also supports automated design
transformations and automatic verification of inter-agent
communications.

The lifecycle of MaSE is iterative. It is assumed that the
analyst or designer moves between steps and phases freely so
that, with each successive step, additional detail is added and,
eventually, a complete and consistent system design is
obtained.

The purpose of the Analysis phase is to produce a set of roles
whose tasks describe what the system has to do to meet its over-
all requirements. The MaSE Analysis phase consists of three
steps: Capturing Goals, Applying Use Cases, and Refining Roles.

Figure 1: Role Model Diagram.
MaSE assumes that the development process starts with a requirements specification which includes the whole set of well-defined requisites. Using the initial requirements of the system, in the Capturing Goals step, the goals are identified and structured into a Goal hierarchy diagram. Next, the Applying Use Cases stage identifies the use cases and creates the sequence diagrams to help to define an initial set of roles. These use cases are, like in UML, a basic scenario about the desired behaviour of the system. Finally, the Refining Goals phase transforms the goals previously obtained into a set of roles. Together with roles a set of tasks are created; tasks define the role behaviour. This step implied the construction of a Role Model Diagram and several Concurrent Task Diagrams, each one specifying the role behaviour for each task, using a finite state automaton.

The main aim of the design stage is to define the overall system organization by transforming the roles and tasks introduced during analysis into agent types and conversations. The design stage is composed of four phases: Creating Agent Classes, Constructing Conversation, Assembling Agent Classes, and System Design.

The first stage maps roles to groups of agent classes and creates the Agent Classes Diagram. Next, in Constructing Conversation, all the conversations are detailed. Description of each conversation needs two Communication Classes diagrams: one diagram for the initiator agent and another for the responder. In the phase of Assembling Agent Classes, the architecture used for agents is decided and then all the internal agent components are defined. Finally, it is in System Design where the number, kind and location of each agent instance are shown in the Deployment Diagram.

4 Meeting Scheduler Development

4.1 Analysis

After taking into account the functional requirements, the system goals have been identified. These goals are included in a Goal Hierarchy Diagram. In this diagram the main goals were: manage users, maintain personal agenda, organize meetings and notify users. With an iterative process all the main goals were decomposed into subgoals. For instance, the main goal manage meetings has been divided into: fix meetings, cancel meetings, change meetings, get participants, get free dates and obtain meeting dates.

During Applying Use Cases phase the Use Cases of the system have been identified. In the system two main actors have been modelled (System Manager and User). All the use cases have been detailed in depth using Sequence Diagrams.

In the Role Model Diagram for the meeting scheduler system, six roles were identified by the developer (see Figure 1): System manager, Users manager, Users informer, Meeting scheduler, Agenda manager, and User.

To satisfy each goal a role develops certain tasks. In Figure 1 the association between meeting scheduler role and tasks is detailed: fix meeting, notify user, obtain suitable date and time. Finally, all interactions involving tasks (communication protocols) are shown in the same figure.

With these roles and tasks the designer defined the internal behaviour of each role in Concurrent Task Diagram Models.

4.2 Design

The final result of this stage was the Agent Classes Diagram (Figure 2) which shows the overall agent system classes and conversations among them. Four agents have been identified:
Matchmaker, UserInterface, AdministratorInterface and MeetingManager. The association between role and task is not one-to-one; for instance, the role meeting scheduler is included in UserInterface and MeetingManager classes. Figure 2 shows four agent classes, their associated roles and the corresponding conversations.

The final step of system design uses a Deployment Diagram for showing the system structure. For constructing the diagram it was taken into account that many instances of the same agent class can be running in the system at the same time. For instance, many users can be connected, each one in a different host. Besides, the meeting manager agent and the matchmaker can be running in two different servers. These latest agents must be always running for the system to be operative. A free version of this Deployment Diagram is shown in Figure 3.

5 JADE Development Framework

Java Agent DEvelopment Framework (JADE) is an agent-oriented tool, implemented in JAVA and FIPA-compliant. It is composed of an agent platform (execution environment) and a set of packages which provide the basic support for multi-agent systems construction.

JADE can be distributed among several computers and is customizable by means of a remote graphical interface. In addition, it provides the mandatory components (FIPA) for agents management:

- **AMS** (Agent Management System), which is responsible of providing the white page, agent lifecycle and agents directory services.
- **DF** (Directory Facilitator), that provides yellow page service
- **ACC** (Agent Communication Channel), which controls message passing.

The development environment incorporates a set of graphical tools which facilitate the platform management, providing support to agents debugging and execution. These tools are: **RMA** (Remote Monitoring Agent), **Dummy Agent**, **Sniffer Agent**, **Introspector Agent**, **GUI-DF**.

In order to create a JADE agent, it is necessary to instantiate a class which extends the Agent class. Each agent is implemented as a thread following the lifecycle proposed by FIPA. Codifying an agent is defining the task it must accomplish during its execution, each task is an instance of **Behaviours** class. In addition, each Agent class will have methods to add and delete behaviours; this action may be done at any time in its lifecycle, depending on what the programmer had decided.

Communication architecture offers a flexible and efficient message-passing mechanism; for each agent, the framework creates and manages a queue for private messages. The communication paradigm used is based on asynchronous message passing, which follows FIPA ACL standard. Each message is an object of **ACLMessage** class, and has methods for managing message parameters, for sending and receiving messages, for message filtering, etc.

Trying to facilitate the development of agents conversations, JADE offers a library where interaction protocols defined by FIPA (Query, Request, Contract-Net,...) are implemented. Interaction protocols define the order and type of the messages involved in a conversation. For each task, it is necessary to add a behaviour in the agent which initiates the conversation and other in each of the agents which will respond (AchieveREInitiator/AchieveRESPonsor) and to implement the suitable handlers to initiate or respond each protocol message.

Agents must share semantics if the communication is to be effective. Therefore, exchanged messages must have a content written in a particular language and must share the same ontology. JADE provides support for using content languages and ontologies. That is, the information is represented inside the agent as a JAVA object (easy to manipulate) and inside the message as a character sequence (easy to transfer). Working with both ways of information representation implies using an instance of **ontology** class and a codec for a content language (i.e. an instance of **Codec** class). Ontology validates the information from a semantic point of view, while the codec translates it to a character sequence, following the syntactic rules of the content language. Taking into account that building ontologies manually is tedious, JADE allows the utilization of other tools for automating its construction (for instance, Protégé for definition and Bean Generator for generating ontology classes).

6 System Implementation

In the system proposed in this work, roles identified during analysis have been related to agent classes in design, and with extensions of JADE Agent or GuiAgent classes in implementation. (GuiAgent class is an extension of Agent class which provides the agent with a graphical interface). So, **UserInterface Agent** class (which, as can be seen in Figure 2, incorporates User and AgendaManager roles) and the **AdministratorInterface Agent** (incorporating Administrator and Users-Manager roles) are GuiAgent derived classes. While **InformAgent** (with UserInformer role) and **MeetingManager Agent** (which implements AgendaManager and MeetingManager roles) are defined from Agent class, because they do not need a graphical interface.

Each task identified in role model and detailed in the concurrent task diagrams, has been used as basis for defining the details of each conversation. From the communication class diagrams of both the agent which initiates the conversation and the responder the interaction protocols that must be used were identified. For instance, when a meeting is to be fixed, the UserInterface Agent asks the Informer Agent to identify the users pertaining to a particular group. This communication is made by means of a FIPA-Query protocol. It is implemented by the AskUser/GroupsInitiator classes (used by UserInterface Agent) and AskUser/GroupsResponder (used by Informer Agent). Once this information is available, the UserInterface Agent makes a request for fixing a meeting to its MeetingManager Agent using a FIPA-Request. This is implemented adding a behaviour **ProposeMeetingInitiator** to UserInterface Agent and the corresponding **ProposeMeetingResponder** parameter in the MeetingManager Agent. This agent will communicate with the MeetingManager agents of all other users implied in the meeting, in order to negotiate a suitable date using the FIPA-ContractNet protocol. Next, all the users’ personal agendas are
updated automatically and the UserInterface agents are notified of these changes by means of a FIPA-Query protocol.

An important issue relating to the JADE platform and models is that some of the developed models introduce information which is not in fact necessary for implementation using the framework. For instance, communication class diagrams demand the achievement of a low abstraction level, specifying all the states the agent passes through during conversation. Meanwhile, conversations in JADE are defined at a higher level, because it adds patterns which implement the standard FIPA interaction protocols.

In the ontology development, the facilities of JADE relating Protégé integration using Bean Generator tool have been exploited. The basic elements of the defined ontology have been:

- Concepts: User, Group, Meeting, MeetingDate.
- Actions performed by agents: Meet, UpdateMeeting and UpdateUserGroups.
- Predicates: ConectedUser and SystemUsers.

7 Conclusions

This paper has shown the use of an agent-oriented approach in the development of complex distributed systems which require negotiation. The built system has fulfilled the aims and goals expected and has been used experimentally for fixing meetings among the members of a university centre.

During the development process, some difficulties were encountered when using the selected agent methodology, mainly due to the lack of practical use cases documentation; particularly cases which reflect the construction of similar systems to the one proposed in this work.

The JADE development tool provides suitable mechanisms for constructing the multi-agent system, making easier the programmer's task when implementing agents and communication among them. An issue that must be improved in the tool will be the development of specific debugger tools which take into account the agent model.

With respect to MaSE methodology and its integration with JADE development framework, some gaps, which must be taken into account in future developments, have been found. As it has been said before, details introduced by some models, for instance the communication class diagrams, can be avoided by attending to the facilities provided by the implementation tool.

Acknowledgements

This research is supported by the Spanish national projects TIC 2002-04516-C03-01 and TIC 2001-5108-E.
References


Engineering Multi-Agent Systems as Electronic Institutions

Carles Sierra, Juan A. Rodríguez-Aguilar, Pablo Noriega Blanco-Vigil, Josep-Lluís Arcos-Rosell, and Marc Esteve-Vivancos

As the complexity of real-world applications increases, particularly with the advent of the Internet, there is a need to incorporate organisational abstractions into computing systems to ease their design, development, and maintenance. Electronic institutions are at the heart of this approach. Electronic institutions provide a computational analogue of human organisations in which human and intelligent agents playing different organisational roles interact to accomplish individual and organisational goals. In this paper we introduce an integrated development environment that supports the engineering of a particular type of distributed system, namely multi-agent systems, as electronic institutions.

Keywords: Auctions, Electronic Institutions, Multi-Agent Systems, Software Engineering.

1 Introduction

As the complexity of real-world applications increases, particularly with the advent of the Internet, there is a need to incorporate organisational abstractions into computing systems to ease their design, development, and maintenance. Electronic institutions are at the heart of this approach. Electronic institutions provide a computational analogue of human organisations in which intelligent agents [8] playing different organisational roles interact to accomplish individual and organisational goals. In this scenario agent technology helps enterprises reduce their operational costs and speed-up the time to market by helping distributed business parties, represented by the agents, run smoother and in a more coordinated fashion. Electronic institutions appear as the glue that puts together self-interested business parties, coordinating, regulating, and auditing their collaborations.

But why electronic institutions? Research and development in agent-based systems has traditionally bargained for well-behaved agents immersed in reliable infrastructures in relatively simple domains. Such assumptions are not valid when considering open systems [6] whose components are unknown beforehand, can change over time, and can be self-interested human and software agents developed by different parties. Thus, open Multi-Agent Systems [8] (MAS) can be regarded as distributed systems where (possibly) large, varying populations of agents exhibiting different (possibly deviating, or even fraudulent) behaviours interact. Note that unlike distributed systems, cooperation among these agents is not fixed at design time but may emerge in real time.

Openness and self-interest without control may lead to unexpected, chaotic behaviours in MAS. The challenging issue is to avoid troubling execution dynamics, particularly in critical applications (e.g. open marketplaces, collaborative project management, virtual organisations, and supply network management). Therefore the design and development of open MAS appears as a highly complex task. It seems apparent, therefore,
for the need to introduce regulatory structures establishing what agents are permitted and forbidden to do. Notice that human societies have successfully dealt with regulation by deploying institutions. Thus, we advocate for the introduction of their electronic counterpart, namely electronic institutions (EIs from now on) [9], to shape the environment wherein agents interact by introducing sets of artificial constraints that articulate and thus help coordinate their interactions. Our actual experiences in the deployment of actual-world MAS as EIs ([12][2]) allow us to defend the validity of this approach. Note though, that as pointed in [4] and [11] we believe that engineers do need to be supported by well-founded tools. Hence the purpose of this paper is to introduce an integrated development environment for EIs that supports engineers in the principle design and development of MAS.

2 Electronic Institutions

According to [10], human interactions are guided by institutions, which represent the rules of the game in a society, including any (formal or informal) form of constraint that human beings devise to shape human interaction. Thus institutions are the framework within which human interaction takes place, defining what individuals are forbidden and permitted to do and under what conditions. Human organisations and individuals conform to the rules of institutions in order to receive legitimacy and support. Establishing a stable structure to human interactions appears as the raison d’être of institutions.

We defend the adoption of a mimetic strategy in order to cope with the complexity of engineering open MAS. Then, if we uphold that open MAS can be effectively designed and implemented as EI, what is an EI? In what follows we identify the core notions on which we found our current conception of electronic institution:

- **Agents and Roles:** Agents are the players in an electronic institution, interacting by the exchange of speech acts, or elocutions, whereas roles are defined as standardised patterns of behaviour. Any agent within an electronic institution is required to adopt some role(s). Recently the concept of a role is becoming increasingly considered by researchers in the agents’ community [3]. We differentiate between institutional and non-institutional (external) roles as well as institutional and non-institutional (external) agents. Whereas institutional roles are those enacted to achieve and guarantee institutional rules, non-institutional roles are those requested to conform to institutional rules.

- **Dialogical Framework:** Some aspects of an institution such as the objects of the world and the language employed for communicating are fixed, constituting the context or framework of interaction amongst agents. EIs establish the acceptable speech acts by defining the ontology and the common language for communication and knowledge representation which are bundled in with what we call a dialogical frame-2-work. By sharing a dialogical framework, we enable heterogeneous agents to exchange knowledge with other agents.

- **Scene:** Interactions between agents are articulated through agent group meetings, which we call scenes, with a well-defined communication protocol. We consider the protocol of a scene to be the specification of the possible dialogues the participating agents may have. Be aware, however, that the communication protocol defining the possible interac-

---

1. In the sense proposed by Searle [13], who postulates that utterances are not simply propositions that are true or false, but attempts on the part of the speaker at achieving some goal or intention.

2. Here we adhere to the definition in [1]: “An ontology for a body of knowledge concerning a particular task or domain describes a taxonomy of concepts for that task or domain that define the semantic interpretation of the knowledge”.

---

Juan A. Rodriguez-Aguilar is a researcher at the Artificial Intelligence Research Institute of the Spanish Scientific Research Council (IIIA-CSIC). He holds a BSc in Computer Science, a Masters degree in Artificial Intelligence and a doctorate in Artificial Intelligence. He worked as a researcher at the Centre for Coordination Science of the Sloan School of Management at the Massachusetts Institute of Technology. In 2000 he joined the research department of iSOCO SA where he worked as a research manager for two years, leading the research and development of iSOCO’s commercial procurement products. His research interests include multi-agent systems, electronic institutions, auctions and negotiation. Dr. Rodriguez-Aguilar has published over forty papers in several books, international journals and international conferences. He has been awarded the best Application prize in the “2003 AgentCities Worldwide Agent Technology Competition”. <jar@iiia.csic.es>

Josep-Lluís Arcos-Rosell is a Tenured Scientist of the Artificial Intelligence Institute of the Spanish Scientific Research Council (IIIA-CSIC). He received a MSc on Musical Creation and Sound Technology from Pompeu Fabra Institute, Barcelona (Spain), in 1996, and a PhD in Computer Science from the Universitat Politècnica de Catalunya, Barcelona (Spain), in 1997. He is the co-recipient of the Swets & Zeitlinger Distinguished Paper Award of the International Computer Music Association in 1997 and of the ICCBR’03 Best Paper Award of the International Conference on Case-Based Reasoning in 2003. He is presently working on Representation Languages for integrating Problem Solving, Case-Based Reasoning and Learning, on the integration of Software Agents with Learning Capabilities, and on Artificial Intelligence applications to Music. <arcos@iiia.csic.es>

Marc Esteva-Vivanco received the BSc in Computer Science and the PhD in Artificial Intelligence (2003) from the Universitat Politècnica de Catalunya, Barcelona (Spain). His PhD, done at the Artificial Intelligence Research Institute of the Spanish Scientific Research Council (IIIA-CSIC), was focused on the design and development of agent mediated electronic institutions. Now he is a Postdoctoral researcher at the University of Illinois at Urbana-Champaign. His current research interests include: multi-agent systems, agent mediated electronic institutions and agent mediated electronic commerce. He has published over fifteen papers in scientific journals and specialised conferences. He received the best prototype paper award at AAMAS 2002 and the second prize in the second edition of the award on New Applications for Internet given by the Telefónica Chair of the Universidad Politécnica de Madrid, Spain. <esteve@uiuc.edu>
tions within a scene is role-based instead of agent-based. In other words, a scene defines a role-based framework of interaction for agents.

- **Performative Structure**: Scenes can be connected, composing a workflow, in a so-called performative structure. The specification of a performative structure contains a description of how agents can legally move from scene to scene by defining both the pre-conditions to join and leave scenes. Satisfying such conditions will fundamentally depend on the roles allowed to be played by each agent and its acquired commitments through former dialogues.

- **Normative Rules**: Agent actions in the context of the institution have consequences, usually in the shape of compromises which impose obligations or restrictions on dialogic actions of the agents in the scenes where they are acting or will be acting in the future. The purpose of normative rules is to modify the behaviour of agents by imposing obligations or prohibitions. Institutional agents are committed to undertake the required actions so as to ensure that non-institutional agents abide by institutional rules.

The notions shown in Figure 1 picture the regulatory structure of an EI as a ‘workflow’ (performative structure) of multi-agent protocols (scenes) along with a collection of (normative) rules that can be triggered off by agents’ actions (speech acts). At run-time agents enter and leave an EI, join and leave scenes that are dynamically created and destroyed, moving among scenes, and act by performing speech acts.

The main goal of an EI is to enforce the specified norms to participating agents at run-time. With this aim all participating agents have their interactions mediated by the EI as shown in Figure 1. Unlike traditional approaches that allow agents to openly interact with their peers via a communication layer, our computational realisation of an EI must be regarded as a *social middleware* that sits between the external, participating agents, and the chosen communication layer validating (filtering in) or rejecting (filtering out) their actions.

3. **An IDE for Electronic Institutions**

IDE-eli, the Integrated Development Environment for Electronic Institutions, is a set of tools aimed at supporting the engineering of MAS as electronic institutions. Software agents appear as the key enabling technology behind the engineering of MAS as electronic institutions. Thus, electronic institutions encapsulate the coordination mechanisms that mediate the interactions amongst software agents representing different parties as depicted in figure 1. IDE-eli allows for engineering both electronic institutions and their participating software agents. Notably, IDE-eli moves away from machine-oriented views of programming toward organisational-inspired concepts that more closely reflect the way in which we understand distributed applications such as MAS. It supports a top-down engineering approach: firstly the organisation, and then secondly the individuals. IDE-eli is composed of:

- **ISLANDER**: A graphical tool that supports the specification of the rules and protocols in an electronic institution.
- **AMELI**: A software platform to run electronic institutions; electronic institutions specified with ISLANDER are run by AMELI.
- **aBUILDER**: An agent development tool.
- **SIMDEI**: A simulation tool to animate and analyse ISLANDER specifications prior to the deployment stage.

Figure 2 depicts the role of the IDE-eli tools in an electronic institution’s development cycle. The cycle is regarded as an iterative, refining process, and this is fully supported by the IDE-eli tools. In what follows we detail the different steps of

---

3. We understand commitments as obligations to do something or to bring about some state of affairs.

4. In Figure 1, JADE [7] is employed as the communication layer.
such development cycle along with the roles played by the IDE-eli tools.

- **Design**: Electronic institutions can be graphically specified with the aid of ISLANDER [3]. It allows for the definition of a common ontology, all of the interactions that agents may have, and the consequences of such interactions. Figure 3 shows the graphical user interface of ISLANDER. The result is a precise description of the kinds and order of messages that the agents in the MAS can exchange, along with the collection of norms that regulate their actions. ISLANDER eases the job of EI designers by supporting graphical specifications based on the formalisation of EIs offered in [4]. Note that the specification of an EI focuses on macro-level (societal) aspects, instead of on micro-level (internal) aspects of the agents.

- **Verification**: Once specified, an institution should be verified before opening it to external, participating agents. This step is twofold. Whilst the first verification stage focuses on static, structural properties of a specification, the second stage is concerned with the expected, dynamic behaviour of the EI. The static verification of EIs amounts to checking the structural correctness of the specifications. For instance, to check that protocols are correctly specified. This process is fully supported by the ISLANDER editor.

On the other hand, the dynamic verification of EIs is carried out via simulation. The process starts out with the definition of populations of agents with varying features capable of acting in the specified institution. In order to facilitate this task we have developed an agent builder capable of generating, from a specification, an agent skeleton depending on the roles and interactions in which the agent may participate. In order to completely define an agent, the generated skeleton must be filled out by agent designers with some decision making mechanisms. Once agents have been implemented, simulations of the EI can be run using the SIMDEI simulation tool developed over REPAST [14]. SIMDEI supports simulations of EIs with varying populations of agents to conduct what-if analysis. The institution designer is in charge of analysing the results of the simulations and returning to the design stage if they differ from the expected ones.

- **Deployment**: Once the institution specification is validated it can be deployed and opened for agent participation. Thus it is time for agent programmers to implement their participating agents. Notice that we do not impose restrictions on the type of agents that can participate in an EI. Agent designers can choose their own language and architecture. Nonetheless, we believe that it is important to support this intricate development process via the aBUILDER tool. aBUILDER facilitates the development of agents in JAVA in a pre-defined architecture. In the future it is planned to support further languages and architectures.

- **Deployment**: An electronic institution defines a normative environment that shapes agents’ interactions. Since agents may be heterogeneous and self-interested we can not expect that they behave according to the institutional rules. Therefore any EI is executed via AMELI [5], an infrastructure that mediates and facilitates agents’ interactions while enforcing the institutional rules. The implemented infrastructure is general purpose, as it can interpret any ISLANDER specification. Therefore it must be regarded as domain independent, and it can be used in the deployment of any specified institution without any extra coding. AMELI keeps the execution state and uses it, along with the EI specification, to validate the actions that agents attempt. Ergo the execution of an EI starts out by running AMELI after loading the specification. Thereafter external agents may enter the institution to interact with other agents through AMELI. An EI execution can be monitored thanks to monitoring tool that depicts all the events occurring at run-time, as shown in Figure 4.

In the next section we sketch an actual-world EI whose development has been supported by IDE-eli.

### Case Study: Agents in Auction Markets

In the Mediterranean regions fresh fish has traditionally been sold through downward bidding auctions operating in auction houses close to harbours. Fish is grouped into sets of boxes, called lots, and put at auction following the Dutch protocol: price is progressively and quickly lowered – 4 quotes per second – until a buyer submits a bid or the price descent reaches the withdraw price. The buyer submitting the bid can decide to buy the complete lot or just some boxes. In the later case, the remaining boxes are put back into auction at the next round. When the last box is sold, the auction is over.

Some fish markets are adapting their selling methods to new technologies and most auctions are nowadays automated by some auction system. The presence of human buyers at the auction houses, however, is still necessary. This imposes two
main barriers. First, it restricts the potential buyers to those present in the auction house. Second, it makes the participation in several auctions simultaneously costly, as companies have to send a representative to each one. The elimination of such limitations would be very profitable for both the buyers and sellers. Increasing the number of buyers makes the market more competitive, and thus increases the buying price to the benefit of sellers. It also permits the participation of buyers without intermediaries saving costs to the buyers.

Agent technologies may be used to eliminate these limitations. The Multi-Agent System for Fish Trading (MASFIT) [2] allows buyers to remotely and simultaneously participate in several wholesale fish auctions with the help of software agents, whilst still maintaining the traditional auctions. The participation of buyer agents in auctions is mediated by an EI. The MASFIT’s institution controls buyers’ access to the auctions, provides them with information, and collects their bids during the auctions. With this aim, the auction system running at the auction houses has been connected to the developed institution. MASFIT interconnects multiple auction houses, and therefore it gives structure to a federation of auction houses. Importantly, the MASFIT system guarantees that the buyer agents have access to the same information, and have the same bidding opportunities as human buyers physically present at the auction house. Furthermore, the system does not alter the current operation of the auction houses themselves.

As reported in [2], the IDE-eli tools played a key role in the design and development of the MASFIT system. The MASFIT electronic institution was specified using ISLANDER, and the agents in the institution have their interactions mediated by AMELI. It facilitates agent participation and communication, within the institution, while enforcing the institutional rules encoded in the specification. It also loads institution specifications as generated by ISLANDER. AMELI permits, among many others features, the distributed execution of agents. This is important as agents must be running at different places.

Finally, note that the market scenario created by MASFIT makes the participation in simultaneous auctions a complex decision-making task. Agents have to manage huge amounts of information – even uncertain information – and their reasoning and processing time must be short enough to react to changes. To support this complex design, MASFIT also includes tools to create, customise, manage and train software buying agents.

## Summary and Conclusions

In this paper we have introduced an integrated development environment for the engineering of multiagent systems as electronic institutions. Major benefits derive from employing the IDE-eli tools. Firstly, they help shorten the development cycle. The engineering of an electronic institution requires a low-cost implementation since only its participating agents
must be programmed. The inherent flexibility of ISLANDER in the design of coordination mechanisms favours an easy, ready maintenance: once changes are accommodated in a new specification, this is ready to be run by AMELI, and agents are ready to “plug and play”. Secondly, the IDE-eli simulation tools support what-if analysis of electronic institutions’ designs prior to their deployment, facilitating the location of unexpected behaviours that may jeopardise critical applications. Furthermore, the development of both external (non-institutional) and internal (institutional) agents for the specified institutions is supported by aBUILDER.

IDE-eli has proven to be highly valuable in the development of e-commerce applications such as the MASFIT system presented in Section 4. However, a wider range of application areas may be tackled with the aid of the IDE-eli tools. In general terms, the electronic institutions approach is deemed as appropriate in complex domains where multiple partners are involved, and a high degree of coordination and collaboration is required. Thus, electronic institutions to support workflow management, the monitoring and management of shop-floor automation, or supply network management issues look promising in the near future.

For more information and software downloads, the interested reader should refer to <http://e-institutions.iiia.csic.es>.

Acknowledgements
Marc Esteva-Vivancos holds the Fulbright/MECD postdoctoral scholarship FU2003-0569. This paper has been partially supported by project Web-i(2) (TIC-2003-08763-C02-01). The authors would like to thank Teresa Solchaga’s help in editing as well as the IIIA Technological Development Unit’s programmers for their valuable contribution to the development of the IDE-eli.

References

Figure 4: Electronic Institution Monitoring.


The Baghera Multiagent Learning Environment: An Educational Community of Artificial and Human Agents

Sylvie Pesty and Carine Webber

This article focuses on a multiagent learning environment named Baghera. Software agent technologies have been quite successfully applied to model and conceive educational environments. The main reasons for that are related to the fact that the multiagent approach applies very well to domains where distance and cooperation among different entities are crucial issues. In this paper, the Baghera platform is presented and its two-level multiagent architecture is described. In order to illustrate this two-level architecture, a learning environment in the domain of geometry proof is introduced. Although many challenges still remain, we conclude this article by discussing some results already achieved, as well as contributions that the multiagent approach can possibly bring to the use of computers in the education field.

Keywords: FIPA-ACL, Learning Environments, MultiAgent Systems, Pedagogical Agents, User Modelling.

1 Introduction

Since the very beginning of Artificial Intelligence (AI) and Education research, the design of educational systems has been based on the principle of teaching a universally valid reference knowledge in a standard way. Thus usual approaches of software engineering were employed in order to set up an educational system. As a consequence, a classical modular architecture composed of three main components (the domain knowledge, the pedagogical expertise and the learner model) was applied in most cases. However, educational systems based on such modular architectures have shown to be too rigid to deal with the quick evolution of knowledge and the diversity of human culture and rationality. Attempts to import tools from AI have been made having in mind also a generic model of teaching and/or learning and referring to a static and academic view of the knowledge to be taught.

On the other hand, educational research has been developing a large number of local models of learning and teaching like models aiming at explaining how to supervise the student’s way of acquiring knowledge. Taking this into account, the development of the Baghera platform has followed an approach where: 1) learner’s knowledge is considered as being composed of a set of ‘conceptions’ whose basic criteria of relevance is not their conformity to some knowledge of reference but their efficiency in specific spheres of practice; 2) the design of educational environments takes the collaboration between human and artificial agents as a foundational principle; 3) education is the result of an emergent complex process and should not be taken as the result of the action of one isolated strategy or the accomplished goal of one isolated agent.

In this context, multiagent methodology has recently appeared as an alternative to conceive AI-based educational systems able to deal with the new expectations of educational systems. Since researchers in the educational field have shown that, if we take into account human differences, it is not possible to find a general strategy of teaching, it seems rather appropriate to think that learning is an emergent result of rich and coherent interactions occurred over time [1]. The multiagent methodology can certainly bring several advantages to the development of educational applications since it deals well with applications where such crucial issues (co-operation among different entities and integration of different components of software) are found.

In this paper, the Baghera learning platform is presented. Baghera is composed of two MultiAgent Systems (MAS), which characterise two levels of decision and behaviour (higher and lower level). Higher level relates to global decision-making and lower level is related to local decision-making.

---

1. Why Baghera? Because the core of the system is a society of non-human agents whose interactions aim the education of a human learner. But unlike the famous Jungle Book, this time some human agents take part in the adventure....
making needed to determine system’s pedagogical behaviour, while lower level is responsible for diagnosing student conceptions, according to student’s actions within the environment interface. This paper is organised as follows: Section 2 briefly illustrates the Baghera platform; Section 3 describes the higher-level multiagent system; Section 4 introduces the conception model and presents the lower-level multiagent system; Section 5 focuses on the lower-level agents’ behaviour, which is based on emergence and voting theories; finally, Section 6 discusses some results already achieved, as well as contributions that the multiagent approach can bring to the use of computers in the education field.

2 The Baghera Platform

From the students and teachers’ perspective, Baghera is a Web-learning application in the domain of Geometry proof. Figures 1 and 2 show snapshots of the student’s problem-solving environment.

Figure 1 illustrates the student’s problem-solving interface. On this window students find tools to edit a proof (A and B) as a free text (C). The problem proposed is composed by a statement (D) and a dynamic figure (E) constructed using Cabri-Java [3]. The dynamic figure can be manipulated in some ways so that the student may verify its geometrical properties and new hypothesis about the problem may come out. Figure 2 illustrates the library of geometrical theorems and properties that students can consult and insert into their text. The proof can be submitted to ATINF (ATelier d’INFérence, an automatic theorem prover applied to the domain of geometry) whenever required by the student [4].

Teachers’ interfaces have extra functionalities, as partially shown on Figures 3 and 4.

Figure 3 shows the interface where teachers are able to handle new exercises. Interface in Figure 4 allows teachers to supervise a particular student’s work by having access to the problem and its solution.

The Baghera learning environment implements a two-level architecture, which is the focus of this paper. Each architectural level corresponds to a particular MAS. The first level (the higher one) is composed of cognitive agents providing the main functions of the educational system (as presented in Section 3); the second level (the lower one) is composed of a large number of reactive agents, which are responsible for diagnosing students’ knowledge (as presented in Section 4). Educational decisions taken on the higher level are based on emergent results coming from the lower level.

The current version of the platform was developed using JatLite [8], a package of Java programs aimed at creating software agent applications. Each agent was extended by an interaction module providing support for creating protocols and coordinating interactions and protocols execution [7]. The actions
that are communicated are based on the speech act theory in accordance with FIPA-ACL standards [6].

Having briefly presented the platform from the point of view of user’s functionalities, we proceed now to describe its architectural levels.

3 Higher Level MAS
The higher level MAS of our platform is composed of cognitive agents providing the main functions of the educational system. Since students and teachers have different goals and needs during learning activities, they interact with different kinds of agents. Persistent data are kept in students’ schoolbags and teachers’ electronic folders. They are personal repositories of data. Teachers have access rights to schoolbags of students belonging to their classes. Each student counts on three artificial agents:

- **Companion – Student’s Personal Interface Agent:** It is an agent with a wide range of goals, associated with the student’s interface. It mainly monitors the student’s actions, notifying other agents when needed and giving access to system resources. This agent controls access to the student’s schoolbag and provides the user with information about the whole learning environment.

- **Tutor Agents:** Tutor agents present the student with the most suitable problem/situation having into account educational goals and learning context. Furthermore, their didactical decisions are based on students’ conceptions. To accomplish their goals they are able to launch the lower level MAS whenever a diagnosis is needed (e.g. a student has completed an exercise, so the tutor has to decide what to propose as next activity) and, once diagnosis phase is over, they plan interactions with other agents and users.

- **Mediator Agent:** The aim of this agent is to choose an appropriate problem solver for the student’s solutions. As it is shown later for the case of geometry proof learning, this agent is connected to an automatic theorem prover, and is able to perform proof verification, propose alternative proofs and build counter-examples. Besides these functions, this agent implements techniques to analyse and present proofs.

Similarly, each teacher can count on the two following artificial agents:

- **Teacher’s Personal Interface Agent:** It is an agent associated with the teacher’s interface. This agent controls the access to the teacher’s electronic folder and provides the user with information about the whole learning environment. This agent mediates interface functions related to: communication with other human and artificial agents, edition of new activities for the students, distribution of such activities to students, and supervision of work done by students.

- **Assistant Agent:** An assistant agent is also a kind of personal agent whose goals include assisting the teacher with the creation and distribution of new activities, which are kept in the teacher’s electronic folder. This agent controls the access to the teacher’s electronic folder and, when demanded, it hands the activities out to students.

As in any open multiagent system, the number of agents increases or decreases depending on the number of users logged in. The general agent architecture was based on the Belief, Desire and Intention model [9]. Each agent has a knowledge base representing its own internal state, as well as other agents’ states. Agents communicate following protocols according to their plans, goals and actions to run.

4 Lower Level MAS
The lower level MAS of our platform is responsible for diagnosing student’s conceptions [11]. The problem of diagnosing students’ conceptions is one of the bottlenecks of research on learning environments. One reason for this problem is the large variety of students’ possible conceptions, either correct or not. A relevant synthesis is presented by Confrey, whose work deals with the paradigm of ‘misconceptions’ [5]. According to his view, if we attentively look for a sense to a wrong answer given by a student, we may discover that it is reasonable. The problem of dealing with student’s mistakes and misconceptions has also been deeply studied by Balacheff [1], who believes that, when analysing students’ behaviour, the existence of contradictory and incorrect mental structures from the viewpoint of an observer must be considered. Such (contradictory and incorrect) mental structures may however be seen as consistent when applied to a particular context (a class of problems or tasks). Following these important principles, Balacheff has proposed a formalism to represent conceptions as follows.

**Definition of a Conception.** Let C be a set of conceptions \( \{c_1, c_2, \ldots, c_n\} \). Each conception \( c_i \) is defined by a quadruple \((P_i, R_i, L_i, \Sigma_i)\) such that:

- \( P_i \) is a set of problems \( \{p_{i1}, p_{i2}, \ldots, p_{in}\} \) where the conception applies;
- \( R_i \) is a set of operators \( \{r_{i1}, r_{i2}, \ldots, r_{in}\} \) solving problems from \( P_i \);
- \( L_i \) is a grammar used to express members of \( P_i \) and \( R_i \).
– $\Sigma_i$ is a set of control structures $\{\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{in}\}$ which represent strategies to solve problems from $P_i$.

This formalism has been used in our framework to represent and diagnose students’ conceptions. A more detailed presentation of this model, followed by examples, may be found in [10] [11].

4.1 The Diagnosis Task: A Group-Decision Problem

Usually a diagnosis task is run by a powerful entity able to observe, make hypotheses and come up with a diagnosis describing the state of the process being observed. Diagnosis may however be a complex task when it comprehends the observation of multiple controls, sensors and variables. In this context, it is appropriate to conceive the diagnosis task as a group-decision problem, where agents (representing sensors, control and variables) collectively take decisions in order to characterise the state of the observed process. Thus diagnosis is not seen as a function of any particular agent, but as ‘emerging’ from the interactive behaviour of several autonomous processes carried out in a co-operative way in the environment.

According to these principles, we have conceived an approach to diagnosis based on the spatial voting theory, where agents share a n-dimensional issue space of possible diagnoses, self-organise themselves according to their proximity, and form coalitions in order to reach together a state characterising a particular diagnosis [11]. The winner coalition represents a state of student’s knowledge (normal, abnormal or faulty) that the majority considers to be the state of the observed process (in our case, conceptions held by the student). In our model coalitions are interpreted in terms of diagnoses according to their positions in the issue space. In this paper we follow a typical definition of coalition [12], described as follows: Considering a set of agents $A = \{a_1, a_2, \ldots, a_n\}$, a coalition is a non-empty sub-set of agents $G$ such that $G \subseteq A$.

4.2 The Theory of Spatial Voting Applied to Diagnosis

In the domain of multiagent systems, the voting theory has been used as a technique for reaching consensus in negotiation processes and group-decision making [12]. Candidates may represent a plan or a strategy for resource or task allocation. Voters may have interests to satisfy and strategies to vote. Analysing their individual preferences and goals, agents choose the best alternative and make a decision in a collective way.

The theory of spatial voting considers that (usually political) questions and voters’ preferences can be quantified. In the spatial voting model, an Euclidian vector space $R^I$ having $I$ dimensions is defined. A dimension originally represents a political issue to vote. Each voter or candidate is represented by a vector in the issue space $R^I$, corresponding to the ‘ideal point’ representing his/her opinions on each issue. Typical behaviour of voters consists of forming coalitions with other voters (close in space) in order to gain power. Similarly, candidates try to adapt their platforms (represented by their positions) in order to attract more voters. The coalition having the majority of voters rules. Competing coalitions may still adapt their platforms to gain greater support. In essence, our interest in spatial voting theory relies on the possibility of capturing group decision, as well as modelling the influence of an agent preference over the preferences of the rest of agents, in order to solve a problem of diagnosis.

In our context the task of diagnosis can be described as follows: Given a set of candidate diagnoses $C = \{c_1, \ldots, c_n\}$, a set of agents $A = \{a_1, \ldots, a_n\}$ and an environment (an Euclidian space having $i$ dimensions), the problem consists in assigning one or more (possibly concurrent) groups $G$ of agents $(GeA)$ representing the state observed and consequently the diagnosis ascribed by the society. The problem is considered solved when one or more coalitions are formed and they remain stable.

4.3 The Lower Level Agents Model

Agents are defined according to the following formalism: Consider $C$ as the set of conceptions $\{c_1, \ldots, c_n\}$ to be diagnosed. For a set of agents $A = \{a_1, \ldots, a_n\}$, let $E_i$ be a set of elements $\{e_{i1}, e_{i2}, \ldots, e_{in}\}$ from $(P, R, L, \Sigma)$. An agent $a_i$ encapsulates exclusively one element $e_i$ (as explained in Section 4). Let $Q_i$ be a set of acquaintances $\{q_{i1}, q_{i2}, \ldots, q_{in}\}$ of the agent $a_i$. Assume $C_i$ as the set of conceptions $\{c_{i1}, c_{i2}, \ldots, c_{ik}\}$ for which the agent $a_i$ votes. $C_i$ is a subset of $C$. Finally, consider $V_i$ as the set of votes $\{v_{i1}, v_{i2}, \ldots, v_{ik}\}$ given by an agent $ai$ to the $k$ conceptions belonging to $C_i$.

An agent $a_i$ is defined by: An identification $N_i$, an internal state $S_i \in \{active, inactive\}$, an element $e_i$ (representing a sensor, control or variable), a set of candidate diagnoses $C_i$, a voting vector $V_i$, a set of acquaintances $Q_i$, and a satisfaction function $f_i(e_i, Env)$. Each agent is in charge of an element $e_i$, whose values may vary according to changes in its physical environment or to signal an occurring problem, for instance.

When an agent locally perceives changes that may indicate a global abnormal behaviour, it becomes automatically involved in the current diagnosis task. In this situation, it is placed in the issue space according to its voting vector $V_i$. Candidate conceptions are represented by points in the space as well. Close agents share close ‘opinions’ and thus they belong to each other’s list of acquaintances $Q_i$. All agents present the same behaviour with regard to the environment and other agents. This behaviour consists in (1) spatially voting for ‘candidate’ conceptions, (2) identifying agents sharing candidates conceptions, (3) forming coalitions with them, and finally (4) merging coalitions in order to increase their utility.

When agents stop, the environment puts a mark on all the interactions that have taken place. This information makes it possible to observe the convergence of agents towards the formation of one or more stronger coalitions. At the level of agents, interactions are created (according to agents’ spatial proximity) and they generate, at the level of society, a process of coalition formation. The final organization of the society is interpreted as the result of a diagnosis task. It means that each coalition of agents is interpreted in terms of a diagnosis of conceptions held by a learner in the resolution of a problem.

When agents from the lower level MAS become stable, since no significant changes are perceived in agents’ states and voting preferences, diagnosis is considered to be over. At this moment, the tutor agent (from the higher-level MAS) receives
the diagnoses as an input to decide (based on educational theories) how to respond to the student. For instance, tutor agent may, if incorrect conceptions are diagnosed, propose problems where the wrong procedures of resolution previously employed failed. Besides interactions with the lower level MAS, a problem solver (ATINF), as well as the point of view of others tutor agents, are useful in providing extra inputs for tutor’s pedagogical decision-making process.

5 Conclusions

The Baghera platform is intended to constitute an educational community of artificial and human agents having different and complementary abilities. The results we have presented in this article represent the initial achievements towards this goal. Baghera has been evaluated by the partners of an European project, especially concerning student diagnosis (lower level MAS). So far results obtained from the computer-based diagnosis system have been positively evaluated by the researchers teams [2]. We recognize however that some challenges still remain, since educational theories are quite complex and finding their computational counterpart models is not straightforward.

Baghera belongs to a very complex class of software tools. From an implementation perspective, the MAS approach has helped developers to model and conceive an application where components, tools, educational theories and decision-making process should work together. In this context, multiple (and sometimes divergent) agents must co-exist and co-operate. Software agent technology has allowed the development of a highly dynamic educational environment where students may learn and interact with other human and artificial agents. Besides, new software agents may anytime join the society in order to suit evolving needs and expectations of students and teachers. Part of this integration relies on technological issues, where FIPA standards as well as the use of common ontologies contribute to accomplish this task. The other part depends on the mechanisms available to reach multiagent global consistency, even though locally agents may diverge (decision and game theories, among others). Finally, we believe that the multiagent approach has proven to be flexible enough to deal well with applications where crucial issues (distance, co-operation among different entities and integration of different components of software) are found.

References

Management of a Surveillance Camera System Using Software Agents

Jesús García-Herrero, Javier Carbó-Rubiera, and José M. Molina-López

Many applications based on distributed resources use the software agent paradigm. In this work, software agents are applied in a surveillance system based on video cameras to sense the environment. The coordination of cameras will enhance the global image obtained from the information provided by all the cameras. Software agents are embedded in each camera and control capture parameters. The coordination procedure is based on high level messages between agents and on the internal interpretation of the situation from each agent. We have applied data mining techniques to learn from real situations how to extract knowledge from the environment in order to detect conflictive situations and to improve the cooperation between cameras.

Keywords: Active Vision, Coordination, Machine Learning, Sensor Management, Software Agents.

1 Introduction

Acquisition and data processing systems from multiple sensors appear in radar-based surveillance systems [1] and in camera-based surveillance systems [7]. In both cases, the surveillance system is composed of a number of sensors to acquire data from each target in the environment (detections, plots, tracks) which are fused in the fusion centre [15], where there is usually a human operator responsible for the supervision of the whole surveillance area. In the design of these systems, two types of interdependent problems arise [6]. The first, data fusion, is related to how data from different sources can be combined in an optimal way [15]; the second, multi-sensor management, uses results obtained in the previous process and determines how to optimize the overall operation of the system by controlling the individual operation of each sensor [9].

This work is centred on the second problem, the coordinated management of different sensors [10]. Sensors forming part of the network can be defined as passive or active depending on their capacity for self-management – whether or not a sensor is able to manage its own resources. Active sensors allow sensor operation to be controlled in such a way that the tasks they perform depend on the surveillance system’s real environment. Functions of a generic sensor which can be managed are [15]: spatial management, management of operation type, time management, and data (communication) management. The management criterion is defined to optimize sensor operation for a certain application. The manager operation is based on quantified parameters such as: probability of detection, quality of tracks, target identification [10], etc. In this regard, two functions need to be implemented: a cuing function, to use detections from one sensor to redirect the attention of another sensor towards a specific target, and a hand-off function to transfer tasks from one sensor to another [1].

The main aim of this work is the development of a management system to coordinate a network of cameras deployed in a surveillance area. The development system is based on a high degree of sensor autonomy, that is, the final decision about the tasks to be executed by the sensor is taken within its own...
management system [16]. The coordination process must organize the flow of information in such a way that communication between the different local managers from each camera and the fusion centre optimizes the use of global resources. In this application, cameras’ time and space allocation (what zones are to be covered at what time) will be managed, along with data processing (what objective is pursued by each manager), and even the coordinated solving of data processing problems. The architecture supporting the coordination process will use Distributed Artificial Intelligence techniques [2], in particular the theory of multiagent systems. Each sensor node of the surveillance network can be considered as a software agent which collaborates with the rest of the agents to solve a given problem (coordinated management). In previous work by the same authors [9], multiagent systems were used to coordinate a network of sensors (in this case multifunction radar) for air defence, using negotiation techniques to determine the most appropriate distribution of tasks at any moment. This present paper extends the application of multiagent systems to the distributed management of active cameras, including the management problems of space-time allocation and data processing.

The decision making capacity of each agent is approached by means of systems based on fuzzy logic. In previous works related to surveillance systems, the authors developed fuzzy systems to determine the priority of tasks in air defence systems [10] and to design image segmentation techniques (with a single camera) in complex situations [3, 8]. In [10] a fuzzy system is developed to prioritize the set of tasks that the sensor must carry out based on measured data and collateral information, such as meteorological or terrain information. The election of fuzzy systems to represent uncertainty in the manager makes use of the robustness and capacity of generalization of these systems. Also in this work, the detection of conflict situations requiring cooperation between software agents is defined by means of knowledge extracted from the images captured by the system. Low level image analysis enables us to detect tracking problems that can require the use of several sensors to perform the desired surveillance [4,8].

2 Agent-based Camera Surveillance System

The research presented in [9] and [10] was focused on managing radar-based surveillance systems through the cooperation of several agents. The performance of that multiagent system prompted us to extend our research to active cameras-based surveillance systems. In our scenario, as in [9] and [10], the multiagent system is made up of two kinds of agents: videocamera agents where each agent controls a videocamera, and the fusion agent which is the central component responsible for the fusion task. All of them have their own decision making process that may conclude actions involving communication with other agents. The purpose of these communication actions is to meet overall goals cooperatively.

Videocamera agents are responsible for, among other things, the monitoring of mobile targets and the surveillance of pre-assigned zones. Meanwhile, the fusion agent should merge the information acquired by the videocameras agents in order to verify whether the overall goals are being met (every mobile target is monitored, and all zones are covered). Depending on the result of this evaluation, the fusion agent may redefine the tasks assigned to the videocamera agents.

Intelligence in artificial vision systems, such as our videocamera agents, arises from the cooperation of two main components with different functionalities: perception and execution of actions/communications. The perception component studies the images captured by the camera and the results of this analysis are considered by the execution component in order to determine what action to take. Both components run in asynchronous mode, with different activating time intervals. Therefore while perception is processing the images captured, the execution component of the videocamera agent may run enough basic reasoning cycles to decide what action to take in the specific situation facing it at that instant. Both components (perception and execution) apply different artificial intelligence techniques: while image processing plays a central role in perception, execution may rely on a reactive decision or on plan and goal based deliberative reasoning.

The authors of [7] propose a reactive approach with three interaction levels: the most basic level allows (reading/writing) access operations on a shared memory, the intermediate level enables interactions between agents with the same target to be precisely identified and located in time and space, while the most complex level of interactions is the set of communications intended to coordinate agents with different targets and re-assign targets to those agents.

However, in a similar problem, described in [13], the authors adopt an approach based on plan and goal based deliberation. In this research, cameras act according to dynamically variable surveillance priorities. Those priorities will depend on the state of the environment. In our scenario priorities depend only on targets, not on the immediate environment.

So, we assume that our videocameras have this ability, and they can survey several mobile targets (of a varying degree of importance) in a defined area (see Figure 1). We also assume that the areas assigned to the videocameras have an average degree of overlap. These cameras are placed in fixed locations, but they have the ability to turn form side-to-side and up-and-down (pan-tilt controls). This movement enables them to orient themselves to track mobile targets. From the image captured we can distinguish a central zone where targets can be viewed.
with sufficient quality, outside which the location and identification of objects is inadequate. Our perception component will let the execution component know whether or not a mobile target is inside this central zone.

Based on these assumptions, videocamera agents have to decide which mobile targets they are going to monitor, in order to prevent any mobile targets from going unmonitored and to achieve an efficient surveillance coverage of the area. The issue of coordination between cameras can be solved in a centralized way: a central entity with full knowledge of all the cameras’ input that can make decisions on behalf of all the cameras, as is suggested in [5]. However, a distributed solution may occasionally (due to scalability and fault-tolerance issues) be an interesting alternative. Distribution is obtained from a multi-agent system, in which each camera is represented and managed by an individual software agent. In addition to computational and data exchange constraints we should also stress the distributed nature of target identification with image processing. It is necessary to have a processor associated with each camera to analyse the images captured in order to identify and locate the targets included in that image. In this way, coordination is a natural solution to this kind of problems, as the authors of [7] consider for a similar problem.

Each agent knows only part of the information (partial knowledge) and has to take decisions within this constraint. A particular agent does not know what is happening outside the area that it is observing, neither does it know the orientation and monitoring intentions of the other cameras. Consequently, the quality of the decision cannot be optimal since optimal decisions can only be taken if we have optimal conditions: i.e. complete knowledge and no computational constraints. This is the case of [14] where each agent works in a completely autonomous way. Even with only partial knowledge, with coordination among agents, the quality of decisions can be increased to close to optimal.

3 Distributed Coordination between Videocamera Agents

In this point we will show the internal architecture of agents, and we will explain the process of coordination between videocamera agents solving problems by themselves (without the participation of the fusion agent), although we will also point out the conditions that would justify the mediation of the fusion agent. We will describe the types of message that a camera agent may send, and the responses to the messages received that will lead to cooperation. The messages received will take the form of requests for execution or information from other agents. The negotiation protocol used to determine each agent’s specific tasks uses the final goals and task priorities set out in [9][10].

3.1 Knowledge Levels of Agents

In this proposal the deliberative nature of software agents involves the description of such agents in terms of the classical structure [12] of beliefs, desires and intentions. Beliefs are the factual knowledge an agent has about itself and about the external world, which in our case are:

- The location of targets captured by the agent camera
- How well each specific target can be visualized (whether or not the target is in the central zone of optimal vision)
- The targets which the agent considers it has been assigned for monitoring
- The other videocamera agents present in the system
- Which other cameras are monitoring targets included in the image captured by the camera
- Which targets are being viewed by other cameras

The intentions of an agent are the actions that it expects to carry out. These can be of two types: turn the camera in a given direction or send messages. Camera turns allow the continuous monitoring of certain targets, while message exchanges permit coordination between cameras.

Each of the abstract desires pursued by an agent controlling a camera are linked to a plan (sequence of intentions) with which it expects to satisfy that desire. When a given condition is triggered, the abstract desire is instantiated in a specific goal, linked to an instance of the corresponding abstract plan. Therefore in our scenario, we will have the following desires:

1. To maintain the continuous monitoring of targets that are located in the optimal zone of vision of the image captured by the camera.
2. Start monitoring a new target detected in the image captured by the camera.
3. Redefine the assignment of a certain target to another agent camera because our camera cannot keep monitoring it in the future.
4. Cooperate with other cameras in the redefinition of assignments of targets, when any of these other cameras cannot maintain the monitoring that they were carrying out.
5. Explore the area that our agent is responsible for, in order to detect possible new targets.

3.2 Coordination between Agents

Coordination is strongly linked with the different desires that videocamera agent may try to fulfil, and depends on the information shared, or execution requests received, by other agents:

- **Monitor a recently detected target:** this desire is instantiated by the detection of a new target in the image captured by the camera. The agent lets the other agents covering an overlapping area know that it will monitor this newly detected target itself. Then it waits for a message from another agent stating that the target has been previously detected (and monitored). If there is no response, the agent assumes that its request for responsibility for the target has been granted.

- **Reassignment of targets:** the instantiation of this desire takes place when several targets are located at opposite extremes of the field of vision. If this is the case, the agent has to decide what target to keep monitored, and therefore which target to abandon. This decision will be taken according to the priority of the targets, the quality of vision of each target, the number of targets at each extreme of the field of vision and the availability of other cameras to monitor the abandoned targets. The reassignment process that will follow is described in the next desire.
Cooperation with other agents in the reassignment of targets: the instantiation of this desire is not triggered by changes in beliefs, nor from the analysis of the image captured. The instantiation is caused by the reception of a message from another agent:

- If the message reports a recently detected target, and that target was previously monitored by our agent, it will notify that circumstance to the sender agent. Otherwise, our agent cannot see the target and therefore will not answer.
- If the message reports the intention to abandon the monitoring of a given target, and that target is in the central zone of vision (the optimal one), our agent will communicate to every nearby agent its decision to accept responsibility for that target.
- If the message reports the intention to abandon the monitoring of a given target, and that target is not in the central zone of vision (the optimal one), it will wait for another message from another agent accepting responsibility for that target. But if this second message does not appear within a certain time limit, it will evaluate the possibility of accepting responsibility for monitoring that target.

In the case of two similar answers, the agent which prompted the reassignment desire due to the decision to abandon the target will decide on behalf of the other two conflicting agents. The fusion agent will then verify that at the end of the negotiation every target has an associated videocamera agent. In the event of there being an unmonitored target the fusion agent can force a re-opening of the negotiation protocol.

Explore the assigned area of surveillance: this desire is instantiated after a fixed period of time when the assigned area is not being completely searched for new targets. Although this desire is permanently active, when another desire is triggered the exploration goal is moved backwards until the monitoring conflict is solved. The coordination of these tasks with the other agents will follow a process of negotiation similar to that described above.

4 Application of Data Mining Techniques to Software Agents

In this section we present the application of Data Mining techniques to the agents [17][11], with the aim of using knowledge about the coordination situation with other agents. This situation knowledge is extracted in the target tracking process, which applies two interconnected phases to the sensor data: data association and estimation [1]. The first phase organizes the received data and performs a correspondence analysis on the targets represented in the system (tracks). The second phase considers the state of each objective (its kinematic location and parameters) from the data associated in the first phase. The problem of the association of data has a combinatorial complexity and has been widely studied for certain conditions of statistical models. When the data to process is images we have a problem with complex types of data, where each target can generate manifold detections distributed in a structure within the entire pixel matrix.

The detection of conflicting situations in the tracking process can be adjusted with an analysis of diverse situations in real scenes. In previous works [4,8] the authors have presented systems able to induce models for assigning detected pixel groups (blobs) to tracks (the targets being tracked). A fuzzy system has been proposed to take the appropriate decision in each situation to update the trajectory and shape of the tracked targets from the extracted regions in the images. The aim of this work has been to ensure a robust and flexible design able to take into account a variety of operational situations while having the computational efficiency required to ensure real time operability. Finally, the developed model not only describes an approach to the problem of the association, but also gives infor-
mation about the difficulty of tracking each specific target depending on the context. Thus, it can be inserted in the scheme presented in Section 3 to coordinate the camera agents in our distributed system.

By way of illustration, we present a conflictive situation in which there is a large target moving inside a loading area of Madrid-Barajas airport (boxed airplane in Figure 1). Its tracking region is divided into multiple disconnected parts, while small targets (vehicles) located behind are moving at the same time and get partially occluded. In the figure there are also four snapshots with the extracted regions of each image (indicated with discontinuous lines and circles for the centroids), which show the imperfections in the target segmentation. The tracks drawn up in the perception module of the camera agent are also printed, represented by continuous outlines for the contours and a vector speed. They have been computed after applying fuzzy logic based processing, able to generate a single stable track per real target.

Knowledge extraction is based on the analysis of the available pre-processed images, looking for interesting relations between input data and results. The aim is to classify the types of situations and their appropriate solution. The first task that must be considered in the approach of applying learning techniques is the formulation of the problem to be learned.

For the case of image processing based surveillance systems, several techniques have been tried, broadly split into static and dynamic learning techniques. Thus, a fuzzy system was proposed in [8] using concepts to describe the possible input situations, such as overlapping of regions describing blobs and tracks, conflicts due to proximity among targets, density of groups, etc. Then, a procedure of learning by means of a neuro-fuzzy system was applied in order to infer suitable association rules. In this case, a static set of representative situations was generated to construct a system able to give them a satisfactory answer and be able to generate appropriate behaviour in intermediate cases.

With regard to dynamic learning, the aim was to obtain accurate estimation results (position, speed, dimensions, stability, etc.) and feed them back into the system in order to improve processing parameters, thereby deviating from classic schemes of supervised learning from labelled static examples. For instance, for systems with hard data assignment, the combinatorial explosion can be avoided with heuristics allowing blob discarding in the enumeration of hypotheses to reduce compu-
tation time. Figure 2 below shows an example where initially there would be 81 hypotheses for assigning 4 conflicting regions to 2 tracks. After the direct assignment of blobs b1 and b4 to their closest track, Track2, the number of hypothesis is reduced to 9.

With this image-data association, in [4] the authors applied model-tree induction techniques to extract models relating the abovementioned input concepts to system behaviour after discarding each type of blob, thereby building a heuristic model which combines efficiency and the benefits of robustness and precision, significantly improving the data analysis based system.

5 Conclusions

In this paper, systems based on software agents have been applied to the management of a surveillance system using cameras as sensors. The use of software agents allows the design of a more robust and decentralized system, so that management is distributed among the different camera agents. The architecture of each agent and its level of reasoning has been presented, as well as the corresponding coordination rules. Finally, we have highlighted the capacity of software agents to learn from real situations and the complexity of the situation facing the agent, all of which is very useful information when coordinating videocamera agents.

Acknowledgements

Funded by the Spanish Government bodies CICYT (TIC2002-04491-C02-02) and CAM (07T/0034/2003 1).

References


An Agent-Based Architecture for Developing Internet-Based Applications

Juan M. Corchado-Rodríguez, Rosalía Laza-Fidalgo, and Luis F. Castillo-Ossa

This paper presents a practical application of an agent-based architecture which has been developed using the methodological framework defined by case-based reasoning systems. The deliberative agents developed within the framework of this research have been used to construct a multi-agent architecture used in an industrial application. The developed architecture is presented together with the results obtained.

Keywords: Agent-based Architecture, CBR-BDI Agents, Case-based Reasoning Systems, Industrial Application, Multi-Agent System.

1 Introduction

Deliberative agents are used in the development of web-based dynamic systems due to their capabilities such as autonomy, reactivity, pro-activity, social ability, reasoning, learning, and mobility, among others [12]. Several architectures have been proposed for building such agents, most of them based on the BDI (Beliefs, Desires and Intentions) model [10]. In this model, agents have mental attitudes of beliefs, desires and intentions. In addition, they have the capacity to decide what to do and how to do it according to their attitudes. Beliefs represent the agent’s information state – what the agent knows about itself and its environment; desires are its motivation state – what the agent is trying to achieve –, while intentions represent the agent’s deliberative states – intentions are sequences of actions – and can be identified as plans [8]. A BDI architecture has the advantage that it is intuitive, and it is relatively simple to identify the process of decision-making and how to perform it. Furthermore, the notions of belief, desires and intentions are easy to understand. However, its main drawback lies in the need to find a mechanism that permits its efficient implementation, as has been analysed in [2].

In order to overcome implementation problems, we propose the use of a Case-Based Reasoning (CBR) system for the development of deliberative agents. The proposed method facilitates the automation of their construction. Implementing agents in the form of CBR systems also facilitates learning and adaptation, and therefore a greater degree of autonomy than with a pure BDI architecture [7][6]. If a proper correspondence between the three mental attitudes of BDI agents and the information manipulated by a CBR system is established, an agent with beliefs, desires, intentions and a learning capacity will be obtained.

Section 2 discusses the relationships which can be established between CBR and BDI concepts. The e-business multi-agent architecture developed is presented in Section 3. Finally we present our conclusions in Section 4.
Constructing Deliberative Agents using a Case-based Reasoning System

Case-based reasoning (CBR) systems solve new problems by adapting solutions that have been used to solve similar problems in the past. This knowledge is stored in the memory (the case base) of the CBR system in the form of cases or problems. As shown in Figure 1, the CBR system performs a reasoning cycle that consists of four sequential phases: retrieve, reuse, revise, and retain [1]. Each of these activities can be automated, which means that the whole reasoning process can be automated to a certain extent [3]. Accordingly, it is possible for agents implemented using CBR systems to reason autonomously and therefore to adapt to environmental changes.

The proposal presented in this paper defines a direct mapping from the concept of an agent to the reasoning model. In the present model, intentions are cases, which have to be retrieved, reused, revised and retained. The structure of the CBR system has been designed around the concept of a case. A case is made up of three components: the problem, the solution, and the result obtained when the proposed solution is applied [2]. The problem defines the situation of the environment at a given moment. The solution is the set of states undergone by the environment as a consequence of the actions that have been carried out within it. And the result shows the situation of the environment once the problem has been solved. This can be expressed as follows:

Case: \(<\text{Problem}, \text{Solution}, \text{Result}>\) (< > sequence, [ ] optional)

Problem: initial_state
Solution: sequence of \(<\text{action}, \text{[intermediate_state]}>>\)
Result: final_state

A BDI agent is defined in terms of its beliefs, desires and intentions:

Belief: state
Desire: set of \(<\text{final_state}>\)
Intention: sequence of \(<\text{action}>\)

The relationship between CBR systems and BDI agents can be established by implementing cases as beliefs, intentions and desires which bring about the solution of the problem. The

---

Figure 1: UML Sequence Diagram Describing a CBR Life-cycle.

Figure 2: CBR-BDI Agent Integration Diagram.
obvious relationship between BDI agents and CBR systems can be seen in Figure 2. When the agent starts to solve a new problem, with the intention of achieving a goal, a new CBR reasoning cycle starts which makes it possible to obtain the solution. The retrieval, reuse and revision phases of the CBR system facilitate the construction of the agent plan. The agent's knowledge-base is the case-base of the CBR system which stores the cases, which are past beliefs, desires and intentions. The agents work in dynamic environments and their knowledge-base has to be adapted and updated continuously; the retain phase of the CBR system takes care of this aspect.

3 Design of CBR-BDI Agents

Given the architecture and definition of CBR-BDI agents, the development of a Multi-Agent System (MAS) can follow the process defined by any agent-oriented methodology which considers the identification of deliberative agents, their responsibilities and goals, their roles in the organization, and the specification of interactions and protocols. Here we concentrate on the design of deliberative agents capable of learning and adapting to new situations by using the architecture proposed in the previous section. To set up an agent using this architecture we need to identify an initial set of beliefs, desires and intentions and include them in the agent's case-base in the form of cases. Next, a number of metrics for the retrieval, reuse, revision and retain steps have to be defined, and rules that describe the Expert’s knowledge must be established, if available. Once the agent has been initialised, it starts the reasoning process and the four steps of the CBR system are run sequentially and continuously until its goal is achieved (or there is sufficient evidence of a failure situation).

The abovementioned architecture has been used to develop an information system for a construction company, D&B Constructions, that specializes in installing heating and air conditioning systems over a wide area of northwest Spain. They have a sales force which is constantly growing, which means that new salespeople are taken on board without much experience in many cases. Up until now the salespeople had to visit customers on demand, take note of their problems, and then contact an engineer or an experienced salesperson to estimate the job price and the personnel and material required to carry out an installation. The proposed multi-agent system supports salespeople, and in particular the inexperienced personnel, while they are visiting customers, by providing advice in drawing up offers, and especially in the estimation of the cost of new installations.

CBR-BDI agents have been implemented using JADE (Java Agent DEvelopment Framework) and JADE-LEAP. The specialised agents run on the company Intranet and the assistant agents run on a mobile device (mobile phone or PDA). Also, in a system of these characteristics, data security has to be taken into consideration. A Role-based Access Control with elements that allow the certification of operations has been implemented to ensure data security and protection of information (similar to the one used in [4] This security system protects the databases and the information stored in the system from external agents or unauthorised personnel.

When modelling a multi-agent system, one of the first aspects to consider is organization. It is not in the scope of this
paper to describe the different methodologies proposed for defining an organization, but so as to be able to understand the structure of the system, we describe its overall behaviour and the interactions between participating agents. Figure 3 shows the system organization together with the agents’ attributes and their functionality. The organization identifies the types of agents and resources in the system, how are they grouped, and the relationships between them. There are two groups of agents: those that run on the intranet (the planning agent, the Internet search agent, and the administration agent), and those that will normally run on remote Internet connected nodes (the assistant agents):

- The planning agent is the only CBR-BDI agent used in this system. The other agents in the system do not have a CBR-BDI architecture because they are responsible for carrying out tasks that do not require reasoning. The planning agent estimates the construction cost, and the personnel and material required to carry out a construction project. It also generates reports about customers (or potential customers) using the information stored in the company databases and that obtained by the Internet search agent. The planning agent generates working plans using its incorporated CBR system.

- The Internet search agent incorporates a web search engine that looks continuously for potential customers and information about them, new suppliers and products. This agent starts looking from a predetermined web address and searches for others using Web optimized natural language processing strategies based on a combination of the grammatical characterization of words and statistical decision techniques. This agent is monitored and guided by a marketing expert. We are also currently studying the possibility of implementing this agent as a CBR-BDI agent to achieve more autonomy and efficiency.

- Assistant agents (there can be as many agents as salespeople and experienced engineers) are interface agents that facilitate the communication between salespeople and the planning agent, and between engineers and the planning agent. They also hold summary information about the customers visited by the salespeople.

- Finally, the administration agent is an assistant agent providing interface to different administration functions, such as the setting up of a new assistant agent, access to the case-base, the inclusion of new cases in the case-base, the definition of security protocols, etc.

The organization also determines the overall goals of the system, which will be later decomposed into more concrete goals for each participating agent. In this example, the main goal of the agent organization is to Assess Salespeople. This goal can break down in others, for instance, Estimate Construction Costs and Report on Customers. Agents in the organization collaborate to accomplish these goals. For instance, for the goal Estimate Construction Costs, the assistant agent takes input data and presents the results to the salesperson, the planning agent calculates the cost estimate, and the Administration agent assists the expert to review the case base (see Figure 4).

There is one assistant agent for each salesperson. Before a salesperson visits a customer, the assistant agent provides a description of the customer by comparing some data provided by the salesperson (e.g., name, address, and activity) with previous queries using relaxed K-nearest neighbour algorithms [11]. This information is related to previous building work carried out for the customer, his financial status, comments about him noted by the firm’s personnel during previous relations with this customer, location information, and other potentially useful data. This information is especially valuable when an inexperienced salesperson starts a negotiation process. If the assistant agent cannot help the salesperson or if the salesperson demands more information, the assistant agent contacts the planning agent which searches for information about the customer in its case-base. This agent also interrogates the Internet search agent asking for information about customers. The Internet Search agent obtains information from predefined web sites, and this information is analysed and indexed using an Internet optimized natural language processing algorithm, as mentioned above [4]. Information about potential customers, new materials and suppliers is sent to the Administration.
Agent, which can be interrogated by any of the Construction Company managers, engineers or sales supervisors. They can then use this pruned information to target new business. The administration agent is therefore an interface agent that facilitates the interaction between the users (Company managers) and the rest of the elements of the system: agents, databases and even salespeople.

4 Conclusions

The system has been successfully evaluated. The planning agent has been fed with 2,320 cases related to the installation of heating systems, which were selected to cover a wide spectrum of possible installations that the company could carry out. The system has been interrogated on 2,458 occasions up until February 2004. In 97% of the system interrogations the estimate by the CBR-BDI agent differed by less that 5% from the one given by an expert salesperson, and in 2% of the enquiries, it differed by less than 10%. Only 1% of the time did the agent output differ by more than 10% from the expert estimate. These errors can be minimised during the review phase in which an experimented engineer reviews the agent proposal. Company experts have estimated that the use of this agent-based system has reduced the cost of installation sales by up to 23%, and the time to sale by up to 38%.

With regard to typical BDI agent architectures, CBR-BDI agents make it possible to build agents with learning capabilities, which makes them more flexible to adapt to changing environment. The developer can use BDI concepts in the modelling of agents, while at the same time obtaining an efficient mechanism to specify an agent’s behaviour with the ability to solve new problems. The results of the use of the multi-agent system described in this paper show how single agents can be developed with this technology, and how such agents can be successfully used to construct an efficient agent-based system for e-business.

Acknowledgements

This work has been funded by the Spanish Ministry of Science and Technology, TIC2003-07369-C02-02.

References


Object Technologies

Integration and Reuse Based on Web Technology and Aspect Object Model

Yauheni Veryha and Eckhard Kruse

To enable easy and effective communication between different information systems, one has to select the most suitable integration strategy. Reusing of applications without large changes in their design could be the best solution in practice. This paper presents an object-oriented integration framework that uses Web technology and Aspect Object Model to integrate applications based on different component models.

Keywords: Aspect Object Model, Information System, Integration Framework, Reuse, Web Technology.

1 Introduction

When software reuse and software integration are put into practice, they can solve a large number of software development problems. The idea of integration and reuse is to allow organisations to have appropriate and related information in a timely manner with minimum modifications in their various existing legacy systems. Application integration is the process of integrating multiple, independently developed applications that may use incompatible technology and need to remain independently managed. To manage their industrial and business processes, today most organizations are using packaged software, like SCADA (Supervisory Control and Data Acquisition) systems, Enterprise Resource Planning systems, Supply Chain Management systems, etc. Successful implementation of consistent, scalable, reliable and cost-effective solutions depends on the standards and the methodologies used for integration. When writing new applications, one can select the exact technologies and methodologies best suited to the enterprise. Component-based development is one of the most known approaches widely discussed and presented in different research works [4] [7] [11] [12] [14] [15]. It promotes the development of tightly coupled, well-defined highly reusable components. Microsoft designed its Component Object Model (COM) specifically for this purpose [9]. The CORBA (Common Object Request Broker Architecture) implemented in several different Object Request Broker (ORB) products and Enterprise JavaBeans (EJB) are other common solutions for application integration [11] [12]. However, none of these technologies are designed to work seamlessly with the others. A true application integration framework must recognize that applications can be written based on different standards, use different technologies and be implemented across distributed heterogeneous platforms.

Practical solutions of integration problems usually require significant development efforts to support the common integration standards. In practice, the main integration and reuse problems include the following:

• Integrated applications may run on different operating systems (various versions of Apple Macintosh, Novell Netware, Microsoft Windows, UNIX, Linux and others);
• Integrated applications may use different incompatible software technologies;
• Integrated applications may not support standard integration concepts.

Reality showed that different competing standards for interoperability like CORBA, COM, IEC 1346-1, etc. [8] [9] [12] [13] would in many cases cause high expenses on modifications of existing software systems to support those standards. Thus, other approaches for fast and non-expensive integration may be needed to overcome common integration and reuse problems.

Software integration often requires many design modifications of existing systems. The reuse of applications without large changes in...
their design could be the best solution in practice. For client/server applications, it is a relatively new strategy to integrate applications on the client machine to reduce integration work [3] [5]. In distributed control applications, it is also important to simultaneously present data from different servers. This requires providing a common interface to integrated applications and establishing a data exchange between them by getting easy access to application specific data structures. Another common scenario in client/server applications is that client applications have to be able to access different servers to get application specific structured data. In practice, client applications are usually designed in such a way that they are able to get structured data from only one server at a given moment. This means that users may start different client applications connected to different servers simultaneously, however, each client application will operate only with data from one server, as pre-defined. In order to solve this problem and provide access to data from different servers, one may have to fully redesign the client application. This may be too time-consuming and expensive in many practical cases.

A closer look at the current state of the Web technologies shows that there are no doubts that they give a big push to the whole software industry. The new .NET technology from Microsoft, Java Web technology from Sun, as well as common HTML (Hyper Text Markup Language), XML (eXtensible Markup Language) and Web Service initiatives, showed that Web technology has not only its value as a tool for e-Commerce and e-Business, but also as an effective tool for solving different integration and compatibility problems [1] [2] [6]. Here, we present an integration framework based on Web technology and the Aspect Object Model [10], which is a common integration concept developed by ABB (Europe’s second largest engineering company), to provide an effective and easy solution to some of the common problems that arise during implementation of reuse and integration concepts. The integration framework allows integrating Web clients of various applications using the Aspect Object Model. One of the important functionalities of Aspect Object Model is that one gets a user-friendly on-screen GUI (Graphical User Interface) that is configurable to present different information. In such a GUI, navigation from one Aspect System to another can be done in most cases with only one or two keystrokes. For example, in process control systems based on Aspect Object Model, users can select (without doing any programming work) a required combination of process graphics, alarm lists and trend diagrams to present them in his GUI directly or as shortcuts so that real-time access to updated and accurate information is always available. This simplifies the user’s tasks and makes user’s operations faster. Third-party applications are accessible with the convenient navigational concept of “Aspect Objects” in the same way. Aspect Object Model includes also a typical object-oriented feature like, inheritance from object type, which makes it much easier to engineer the system.

We will use Aspect Object Model as a starting point and describe the features of the model and how it is realised in a framework in Section 2; we then continue by describing the integration framework environment in Section 3, some implementation details in Section 4 and discussion in Section 5; finally, in Section 6 we make a conclusion.

2 Aspect Object Model for Application Integration

With enterprise customers putting more applications on the Web, vendors are scrambling to provide the infrastructure that will integrate different applications and let them work together more efficiently. That is what Aspect Object Model has in mind. Figure 1 shows a block diagram representing schematically a simplified architecture of the object-oriented framework and its relation and interaction with integrated applications based on the Aspect Object Model. The object-oriented framework defines and implements a conceptual Aspect Object Model. It is not the goal of this paper to provide the detailed description of the Aspect Object Model because it would require too much paper space, therefore, we re-direct those interested in the Aspect Object Model to the reference [10] and ABB’s Web site <http://www.abb.com>

Shortly, the Aspect Object Model defines three central concepts: objects, aspects and aspect systems (see Figure 1). Objects represent quite large entities of physical objects like pumps, power stations, motors, etc. In the computer environment, they can be represented, for example, as COM objects using Microsoft COM (Component Object Model) technology and structured according to the concept of the “Directory enabled applications” of Microsoft [9]. Objects usually contain very little information or implementation in them. Instead of attempting to permit an object itself to represent its entire behaviour, part of its behaviour is delegated to different aspects. Aspects are used as a link to the Aspect Systems (applications) that actually implement the object’s behaviour. For example, the following applications can be integrated in the object-oriented framework as aspect systems: CAD (Computer aided design) systems, Microsoft Office tools, MatLab, etc.

User management within the object-oriented framework is implemented based on Windows NT accounts. This means that every server-authenticated user will have a certain role within the object-oriented framework and certain permissions and rights regarding the objects, aspects and aspect systems integrated by the object-oriented framework (see Figure 1). The presented architecture is applicable both to the server and the client machines. This means that, in case of the client machine, the object-oriented framework will be a remote one (remote objects of the object-oriented framework will be established on the

![Figure 1: Block Diagram Representing Schematically Object-Oriented Framework and its Relation to Applications Based on the Aspect Object Model.](image-url)
It is important to mention that in the given approach, a prerequisite is that integrated applications do support Web clients or can be extended to support them. Therefore, integrated applications can be managed separately in 'thin client' or 'fat client' scenarios. ‘Thin clients’ use pure HTML or Server Pages with client script while ‘fat clients’ use HTML Web pages with different embedded client controls, like ActiveX, Java Applets, Windows Controls (assemblies) from .NET, etc.

3 Environment of Integration Framework

The integration framework based on Web technology and the Aspect Object Model has been developed to work in a client/server environment, in particular using the HTTP protocol between Web server and Web clients. The architecture of the computer environment for the integration framework is shown in Figure 2. The server and client systems may run similar or different computer operating systems able to communicate with each other using HTTP protocol. The server system is accessed by the client system for specific functions, such as execution of applications, storage and retrieval of data, access to user information, etc.

Figure 2 illustrates a computer environment in which one or more client machines 2 and one or more server machines 1 are in the network and communicate using the HTTP protocol. Thus, the communication is established using the Request/Response model. Server 1 in the computer network is a personal computer or server platform that includes an operating system, Windows XP based server operating system 9. The server includes an object-oriented integration framework 11 with objects 12, that is aimed to integrate all running server applications 6 and/or applications located on other servers (not shown in Figure 2) by interfacing those applications using common programming interfaces similar to Microsoft COM [9]. The access to the data sources, like database 10, is possible through the integrating object-oriented framework 11, when it is needed to keep consistency of the current user context. The processes of object-oriented integration framework 11 and other server applications 6 run under the currently authenticated user context obtained after successful user authentication against the server domain 8 and one of its accounts located in the groups 7 of the server domain. Typical client machines 2 are personal computers or workstations running Microsoft Windows 2000 based client operating system 3.

Following successful authentication of the client machine, server objects 12 from the object-oriented integration framework 11 are dynamically established on the client machine using remoting of server objects using Microsoft .NET Web services (other remoting technologies can be also used to implement the same approach). The remote proxy of the object-oriented integration framework 4 associated with the server-authenticated user becomes available on the client with appropriate client objects 5 which are remotely established. An important feature of the object-oriented integration framework is that applications running on the client and on the server can work both on the client and server without differentiating the type of objects-oriented framework used (remote or non-remote). This can be achieved by providing the same registration information of objects (and their programming interfaces) on the client and server machines. Thus, the remote proxy of object-oriented framework provides the same navigation and structuring functionality to the
Web clients as the object-oriented framework on the server side provides to the server applications (6 see Figure 2). This assures simplicity of the object-oriented framework deployment, easy structuring and interoperation between separate applications referenced by the object-oriented integration framework independent of the client or server integration level (see Section 5 for details.)

4 Implementation

The implementation details of the integration framework based on Web technology and Aspect Object Model are largely dependent on the used software technologies and operating systems. In our project we implemented it using Microsoft .NET technology [9] as it is shown in Figure 3.

The Microsoft .NET technology allows to easily establish a remote proxy (see Web Client in Figure 3) of the object-oriented integration framework on the Web client machines and to provide the same Application Programming Interface for integrated applications (see Web Clients of Applications in Figure 3). In the worst integration case (when no deep integration is possible), in order to comply with the reuse requirements suggesting minimum modifications of integrated applications, we simply integrated Web clients of the applications on the client side using URL references, like:

http://<server>/<page>?objectID=<ObjectID>&aspectID=<AspectID>&Params=<Params>

This integration is shown using dashed lines in Figure 3. This means that integrated applications will use their own communication model with their Web clients. The communication between applications will then be established on the client using only common Web browser features, like, ‘drag and drop’, frames in Web browsers, cookies, etc. An example of such ‘thin client’ of the integration framework using Microsoft .NET and Aspect Object Model in the Web browser is shown in Figure 4.

The window in the right bottom part of the screen (see Figure 4) provides the area for integrated applications (Aspect Systems). Graphical user interfaces of integrated applications can also be popped-up in the new separate windows by clicking on appropriate Aspects in the Aspect List.

5 Discussions

The most important feature of reuse is that it can be highly profitable. Thus, for companies that want to minimize their development costs in integration projects, it is one of the best practical approaches to concentrate on reuse and, therefore, on integration frameworks that allow this. The presented integration framework based on Web technologies and Aspect Object Model uses a set of standard schemes. These not only provide a runtime and design environment, but also a number of standard components and knowledge of how to build a common user interface.

The functionality of such integrated system is accessible from a single GUI that is configured to present information and provide inter-action in a context appropriate to all user disciplines. This eliminates wasted time spent searching through data or views that are not relevant to the individual job function in order to get to the desired information. The growing deployment of peripheral applications related to productivity improvement significantly increases the amount of data available within the process control systems. This may lead to the situation when personnel can be exposed to information overload. The proposed concept based of Web technology and Aspect Object Model provides a single user interface that dramatically increases individual productivity by saving time and effort for each user. The drawback of the our approach is that not all applications support ‘thin client’ scenario. Thus, it may require significant additional work to integrate such applications properly.

Generally, it is not very difficult to integrate modern, component-based applications supporting well-known integration standards as well as running on the same platform at which the integration framework is targeted. However, integrating monolithic applications with poor or not clearly defined application programming interfaces is not only difficult but sometimes it is highly questionable due to the very high cost of the integration work. In the latter case, some easy and flexible integration approaches are required. An integration framework based on Web technology and Aspect Object Model is one possible direction to integrate separate applications achieving maximum possible reuse. In this case, Web technology provides a common integration environment for applications and Aspect Object Model provides common application structuring and interoperability concept.

6 Conclusion

We have presented an integration framework implementation based on Web technology and Aspect Object Model to encourage the building of integrated solutions using application reuse. Such integration solutions find their wide use in automation control systems, utility automation systems and other technical solutions where large number of separate applications should be simultaneously used to exchange data and provide common graphical user interface. By using available concepts, like Aspect Object Model of ABB and Web technologies, we have shown an implementation approach and integration environment that allows integrating and reusing available applications in a consistent and structured way. The implementation of the Aspect Object Model makes it possible to reuse
whole applications, which originally may not have been designed for reuse, but support Web technologies.

In the future, we will provide more practical aspects related to integration costs of integration framework implementation in pilot projects.

References
News & Events

CEPIS News

CEPIS Experts Appointed to ENISA Management Board

The European Council of Ministers has appointed German Professor Kai Rannenberg and Slovenian Government Secretary Niko Schlamberger as Stakeholder Representative and Alternate for the Management Board of the new European Network Information Security Agency (ENISA, <http://www.enisa.eu.int>). ENISA was formed by the European Union “to develop a culture of network and information security for the benefit of the citizens, consumers, enterprises and public sector organizations of the European Union” in June 2003. The Management Board is made up of EU Commission Representatives (3), EU Member States Representatives (25) and Stakeholder Representatives (3).

Proposed by the Council of European Informatics Societies (CEPIS, <http://www.cepis.org>) as experts for network and information security Rannenberg and Schlamberger will represent European academic and professional community within ENISA’s Management Board. Both bring in specialized knowledge gained in research, education, training, and professional work.

Professor Dr Kai Rannenberg (40) is Professor for Business Informatics. He holds the T-Mobile chair for Mobile Commerce and Multilateral Security at Goethe University Frankfurt, Main, Germany, <http://www.whatismobile.de>, since 2002. His areas of work include mobile communications, security and privacy, as well as identity management, and he is coordinator of the EU FP6 Network of Excellence “Future of Identity in the Information Society” (FIDIS). Before he was with the System Security Group at Microsoft Research Cambridge, UK, focusing on “Personal Security Devices and Privacy Technologies”. From 1993 to 1999 Rannenberg coordinated the interdisciplinary “Kolleg Security in Communication Technology” that worked on multilateral security, especially protection for users and subscribers. The Kolleg was sponsored by Gottlieb Daimler and Karl Benz Foundation. Rannenberg is chair of the CEPIS Special Interest Network “Legal and Security Issues” and vice-chair of the Technical Committee 11 “Security and Protection in Information Systems” of the International Federation of Information Processing IFIP.

Niko Schlamberger (62) is Government Secretary at the Statistical Office of the Republic of Slovenia in charge of leading special projects inter alia the introduction of a classified data protection system. Earlier he gained professional experience in programming, application development, consulting, project management, and general management in IT industry, in business, and in government. Schlamberger was head of software development in a major Slovenian bank, IT consultant, assistant to general manager of the ex-federal clearing agency, and head of the Slovenian government information technology office. Schlamberger is president of the Slovenian Professional Informatics Society Informatika. Since August 2003 he is also elected Vice President of the International Federation of Information Processing IFIP.

For further information please feel free to contact <secretary@cepis.org>.

July 26, 2004

ECDL News

Italy: Agreement between AICA and Sun for Use of StarOffice 7 in Private ECDL Test Centers

With the new version of the application Alice, the distribution and evaluation of ECDL tests can be performed with Sun products; candidates will be able to select StarOffice 7 as their platform of choice. With this initiative Sun Microsystems extends its collaboration with the Italian CEPIS society AICA (Associazione Italiana per l’Informatica ed il Calcolo Automatico, <http://www.atica.it>), whereas all the private ECDL Test Centres in Italy will also have the option of becoming StarOffice Resellers.

It is worth to mention that after the 2003 agreement with the Italian Ministry for Education, University and Research, StarOffice is available free for all educational centres of any level in Italy, and can be distributed to students, teachers and staff personnel.


Spain: ECDL Adapted for the Open Office Environment

The Spanish CEPIS society ATI (Asociación de Técnicos de Informática, <http://www.ati.es>) and the Universidad Autónoma de Barcelona (UAB) have signed an agreement whereby UAB will provide European Computer Driving Licence (ECDL) certification adapted for the non-proprietary Open Office environment. This will specifically involve the design and development of the automated tests and their templates for the abovementioned environment.

This important initiative is another demonstration of ATI’s commitment to the development of free software in Spain, a commitment which is evidenced by a number of initiatives, among which we would highlight the association’s participation in the campaign against software patents and the creation of ProInnova, an interest group in favour of the freedom of innovation in Informatics promoted by ATI and the Spanish association Hispalinux.

For further information contact <secretregen@ati.es> or visit <http://ecdl.ati.es>.

June 3, 2004

EUCIP News

Norway: Major Norwegian Learning Providers Come up with The Goods

EUCIP<sup>5</sup> accredited Learning Providers NITH and TISIP in Norway have produced courseware for EUCIP’s Core Level programme.

The first web based course from NITH (introduced in February 2004), for the Build module is delivered in English, over the Internet with tutor support by email. Courses for the two remaining modules of Plan and Operate will follow very soon.

NITH also has the capacity to run EUCIP exams from its accredited examination centres in Oslo, Bergen and Stavanger. To find out more, please visit <http://www.nith.no/eucip>.

TISIP is producing books for each of the EUCIP Core level modules of Plan, Build and Operate. The books are aimed at the Norwegian market and their planned release date is the beginning of the academic year 2004-2005. The books can be used as “stand-alone”s to supplement existing knowledge, or as part of the web based courses.

For more information, click on <http://www.tisip.no/eucip>.

July 1, 2004.

QofIS 2004 (Fifth International Workshop on Quality of Future Internet Services)

This workshop will take place in Barcelona, Spain, from Sep. 29 to Oct. 1 2004, organised by Universitat Politècnica de Barcelona.

The purpose of QofIS 2004 is to present and discuss design and implementation techniques for providing quality of service in the Internet. The impact of emerging terminals, mobility and embedded systems will create a new environment where networks will be ambient. New challenges are open by this new space where networks of interest ranging from personal networks to large-scale application networks will need to be designed and often integrated. Protocol mechanisms for supporting quality of service at the different layers of the networks will need to be assessed and eventually redesigned in such environments. The focus of this year’s workshop is on the provisioning of Quality of Service in the Emerging Networking Panorama, assessed by results of experiments carried out in simulation platforms and test-beds, and given the progressive irruption of optical technologies.

This workshop, at its fifth international event, is organized under the umbrella of the E-NEXT Network of Excellence on “Emerging Networking Experiments and Technologies”, which started earlier in 2004.

Along with presentations of papers and keynote speeches, QofIS 2004 will offer two one day workshops on October 1, namely the First International Workshop on QoS Routing (WQoSR’04) and the Fourth International Workshop on Internet Charging and QoS Technology (ICQT’04)


62 UPGRADE Vol. V, No. 4, August 2004 © CEPIS
Large-Scale Software Development in a Multinational Software Corporation

Jacek Czerniak and Wojciech Spiewak

© Pro Dialog 2004

This paper shows the process of code management. The study was based on the software integrity inspection process performed in R&D departments of the company Lucent Technologies. The authors present processes and principles that rule software code changes and quality control of performed changes.

Keywords: Software Engineering, Code Management, Software Inspection, Software Development Process, Software Quality

1 Introduction

Starting our discussion on code management in a corporation, we provide a number of notes concerning the background of the topic. The Lucent Technologies is a world-renowned company manufacturing telecommunication equipment, including the 5ESS phone switches, named 7R/E in their newest version. Used all around the world, the 5ESS switches are also the telecommunication network base of Polish providers i.e. TPSA, Netia, or El-Net.

The development of the software for the 5ESS switches takes place at Research and Development Departments of Lucent Technologies, which are spread over several countries, including USA, Poland, China and India. The 5ESS, in itself, is a multi-processor system, consisting of over 200 processors. The operating system that manages it is one of Unix releases originating from Bell Labs – formerly a part of AT&T. The software of 5ESS and 7R/E switches has been divided into 50 sub-systems. It is constantly being developed by numerous programmers (from about 3000 in 1998 and about 1500 at the moment). To provide a full picture, it is necessary to add that the entire software of the switch includes over 10 million lines of code [1]. From this prospect, it is easy to understand why it is so important for a company to have an efficient code management system.

Experience that has been gained by Lucent Technologies is even more interesting, taking into account that none of Polish software companies employs several thousand developers who, what is more, are spread over distant countries. Furthermore, the methods applied by LT and the quality achieved thanks to them have been confirmed by the certificates ISO9001 and TL9000.

2 Code Change Management

Access to code is provided by a separate group of servers storing the source code [2]. Each programmer has his or her own account on each server that is needed for his or her work. Code management for 5ESS and 7R/E switches is realised on two levels:

- ECMS (Extended Change Management System) [3] – the layer that initiates changes within the framework of a project,
- SCCS (Source Code Control System) [8] – the layer responsible for the control of the stored files’ versions.

Each project feature realised by a given developer is unequivocally identified by a unique IMR number. This acronym stands for Initial Modification Request. As the IMR number is associated with the project feature, it is easy to presume that we need more detailed identification of changes made in specific files of the code. That is why, on the basis of the IMR number, one generates numbers identifying specific changes, as needed. These are so called MR (modification request) numbers. It is a common programming practice to use one MR number to apply...
changes in one file. If such a solution seems slightly complicated, one should bear in mind that the database may be accessed 24 hours a day by several thousand programmers. Thus each of the code integrity fluctuations may be dangerous for the functioning of the system.

Once the changes are introduced to the file based on the MR number, the SCCS layer stores up-to-date information on added, changed, or deleted code lines. Such a collection of changes is called a delta [4][5]. For each delta, the ECMS layer stores a record of changes that includes:
- date and time of change,
- string of characters that identifies the developer,
- list of MR numbers, introducing changes.

Figure 1 shows the change introduction process using MR numbers. This process has already been partially described above, but the schematic presentation gives a complete and precise view.

Creating IMR Number for a Given Project

This phase has already been described quite well. Thus, summing up, it consists in generating a new number that initiates a change for a feature (IMR number). All the changes connected with a specific feature might thus be unequivocally identified as belonging to a given project. Thanks to those numbers, gathering code changes that are to be delivered to a specific customer is quite easy and clear.

Making Changes in Files

As has already been mentioned, the entire software that ensures proper operation of the 5ESS and 7R/E switch is divided into about 50 modules [1] which control the different aspects of its functioning. Among others, they include such subsystems as:
- Call Processing Subsystem,
- Charging Subsystem,
- Traffic Measurement Subsystem.

Thus, the developer sets environmental variables specific for the module, the files of which will be changed by him or her.

Creating MR Number Associated with IMR Number in order to Perform Changes to the Module

A developer who already has the IMR number associated with his or her feature has to generate a list of MR numbers specific for the current module by use of which the changes in files are going to be made. Those numbers allow for unequivocal identification of the change as belonging to a specific project feature and as applied by the engineer. Thanks to this, none of the changes in the code is anonymous, which also efficiently protects the system against sabotage.

Collecting Files for Editing

The files are unavailable to other users while they are being edited. Whereas other developers who would like to make changes to the same file can see by the ECMS layer interface, who is editing the file at the given moment. Thus editing the file for too long exposes the developer to phone calls and emails from annoyed co-workers. For testing purposes, it is also possible to create local copies of files, which are to be modified. However, it is intended exclusively for initial testing of a concept by the programmer. In the official base of the code, the changes are made only by collecting the files for editing using MR number.

Transferring Files Including Unofficial Changes to the Base

The changes are unofficial as long as they are not officially added to the code. They are written as so called deltas in the SCCS layer. Each user can view the code using the ECMS interface and find out information whether a given code segment is included in the official file and, if not, then what its status is. It is very important because certain changes may overlap and therefore such information are crucial for the developer.

Compilations and Code Testing

This is the phase of preliminary testing, performed by the programmer. Here he or she tests the entire feature or its separate part developed by him or her. The phase of testing the whole project, which consists of several or a dozen or so features carried out by different engineers, takes place later on.

Code Inspection

This phase of software generation is very interesting and requires more discussion. This process is directly responsible for maintenance of code quality, creation of engineers’ good programming habits as well as for standardisation of the form in which the entire code is kept. The details of the code inspection
process will be discussed in the next paragraph.

**Transferring Changes into the Official Files**

The actual transfer of changes for testing by the Feature Tester takes place in this phase. This is the person in charge of inspecting the entire project by testing the changes made by all the project participants. In fact, we deal here with a complete inspection of the project. Without a successful outcome of this phase, it is impossible to integrate the code with the official version.

**Integration of Changes with the Official Code**

Only at this phase the changes made to the code by the programmers actually become part of the official code.

Summing up, it has to add that all the phases mentioned above are executed very strictly. It is easy to understand this fact, bearing in mind that several thousand programmers are working simultaneously on several million lines of code. From this perspective one can see why it is so important for a company to have and to maintain quality confirmed by certificates mentioned at the beginning of this paper.

### 3 Code Inspection Process

**3.1 Introduction to the Code Inspection**

As mentioned in the previous paragraph, each change made to the code goes through called Inspection Process [6]. The author of the particular change is responsible for the organisation of such consulting event. Thus he or she collects the team consisting of the moderator and 2–4 inspectors, depending on the complexity level of the project feature which he or she deals with. The moderator is usually an experienced engineer who has been specially trained for leading code inspection processes.

Generally, the Code Inspection Process has three phases [7]:

- preparation,
- collection,
- repair.

The first one consists in an individual analysis of the code made by the inspectors searching for errors. Then, during the meeting of the inspectors, the moderator and the author, they exchange comments and propose possible changes. Finally, the author makes corrections suggested by the inspectors. Figure 2 presents schematically the phases described above.

**3.2 Details of the Code Inspection**

To have a complete understanding of the code inspection process, one has to present detailed study of it. Below (see Figure 3), there is a set of phases [7] through which the changes made by the developer must go.

**Creating MR Numbers**

Creating MR numbers, always when some change is to be done. As explained in the previous paragraph, MR numbers generate special software tools on the basis of IMR numbers. Thus, one can always identify the changes as associated with specific project or its independent part.

**Making Changes to the Code Using MR Numbers**

Changes are made to the code using MR numbers. This process has been described in detail in the previous section. It is only worth repeating that without MR number no changes to the code are possible.

**Preparation of the Package Intended for Inspectors**

The author of the changes prepares an inspection package intended for the inspectors. It is usually prepared in printed form and includes the code developed or changed by the author. Analysing those materials, inspectors may also learn how the code before the changes looked like. The package usually includes a short note that explains the type of changes that were made.

**Choosing Moderator and Inspectors**

The number of inspectors depends on the number of lines inserted by the author. The smallest team of inspectors consists of:

- moderator,
- three inspectors,
- secretary,
- author of the changes.

However one of the inspectors may also be the moderator, whereas the author usually also acts as the secretary. The moderator is usually an experienced engineer who has been specially trained for leading code inspection meetings.

**Arranging the Meeting of the Moderator and Inspectors**

The meeting of the moderator and inspectors is arranged by the author. In exceptional cases, it may be held on-line, by sending an e-mail with comments. This is called a Desk Inspection. However, the recommended form in the company is a face-to-face meeting, which is called a Meeting Inspection.

**Preparation Period**

Inspectors review the code independently in order to find potential faults. The time needed for this activity is then noted in the inspection protocol. There are certain standards binding in the corporation regarding the minimum time that should be spent for an inspection of a given quantity of code lines.

**Meeting of Inspectors and the Author**

As has already been mentioned, such a meeting may have two forms, i.e. Desk or Meeting. The entire meeting is chaired by the moderator. At the beginning the author briefly presents the problem which prompted the changes. Then, the participants exchange comments, suggestions and questions. Sometimes inspectors suggest alternative solutions and methods to those used by the author. During this meeting, the author collects all the suggestions and comments.

**The Author Makes Changes, if Needed**

The author is obliged to check the changes suggested by the inspectors. He or she pro-
ceeds in the same way as while making initial changes, using MR numbers in the way that has already been described.

**Moderator Checks Author’s Changes**
Moderator is obliged to check if the author took into account all the inspectors’ suggestions stated during the meeting.

**Moderator Approves Entire Changes Made by the Author**
Moderator fills in a special protocol of the meeting, including the approval of all the changes made by the author. To the official software version may be introduced only the changes made using MR numbers, for which in the base exists a record from the code inspection process, which has been approved a certified moderator.

Only after passing through all those steps, the code moves to the final testing phase. This is so called the Feature Test. After successfully passing those tests, the code will finally be added to the official version of the software and delivered to the customer. The same approach is applied in the project documentation.

### 3.3 Gains and Losses Resulting from the Code Inspection

#### Code Inspection Costs
There are two types of costs resulting from the introduction of code inspection. The first are of single nature and they may be treated as preliminary costs:
- implementation of procedures,
- training of moderators and inspectors.

The second category of costs are fixed costs, added to every development. The permanent cost are estimated as follows:
- app. 6% of development time devoted to Inspection,
- usually 10–15% of the development budget.

#### Gains Resulting from the Code Inspection
Earlier detection of defects saves time and money:
- many defects are found before testing, so testing is much quicker,
- total development time is reduced, even if inspection time is included, so inspection improves productivity.

Earlier detection of defects contributes to a better product:
- cleaner design,
- better documentation,
- better code,
- fewer defects,
- defects found earlier result in fewer deadline surprises,
- defects that remain will be easier to fix when found.

Inspectors learn from the inspection process:
- they learn from good code and documentation practice of others,
- they learn from mistakes of others.

In summary, here are some statistics in favour of inspection obtained from software development management tools:
- defects found by inspection: 57.7%,
- cost of finding defects by inspection: 1.58 work hours,
- cost of finding defects without inspection: 8.47 work hours.

### 4 Conclusions
As one can see from the above study, the principles applied during software development at Lucent Technologies are very strict. This process is constantly being improved and dynamically changed by scientists from Bell Laboratories and the University of Maryland. The data used for research on the effectiveness of the software development process are taken from numerous statistical tools used every day by developers. As can be seen, this multinational corporation has worked out its own work method and implements it in all departments, whether they are located in the USA, Europe or Asia. Thanks to the application of the same principles, it is easier to get a final product of high quality and handle migration of employees within the corporation, because apart from the procedures also the language is unified. The official language in the company is English, which significantly facilitates internal communication.

All those principles together are an interesting material for software development analyses. Code inspection process studies are performed not only at Lucent Technologies. For example, IBM company, another giant of the IT market, has worked out interesting material while developing similar methods. Both companies carry out experiments in order to determine an optimal number inspectors in relation to the quantity of inspected code lines. Recently, IT companies have also tested teams working in parallel who check the same code segment. In Poland, with a few exceptions, there are primarily small or medium size IT companies. It also seems to be interesting to analyse whether (and how) the principles presented above can be implemented in Polish companies. Which parts of the processes discussed in this article may be
applied in small and medium size companies, and which should only be introduced to big ones? It would also be interesting to determine, basing on the studies above, practical rules designed to assure integrity and high quality of the software developed by typical Polish IT companies.

References


Informatics Profession

How will we computer professionals earn a living? (And why don’t you teach for free?)

Ricardo Galli-Granada

© Novática 2004

This paper was first published, in its original Spanish version, under the title "¿De qué viviremos los informáticos? (Y tú, ¿por qué no enseñas gratis?)", by Novática (issue no. 168, March–April 2004, <http://www.ati.es/novatica>). Novática, a founding member of UPGRADE on behalf of CEPIS, is the bimonthly journal and magazine of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática, <http://www.ati.es>.)

In this article the author, a university lecturer in Computer Science, takes an unorthodox look at the computer profession and answers the question posed by an opponent of free software / open source “Would you be willing to adopt the same approach to your teaching and teach for free?”

Keywords: Computer Profession, Education, Free Software, Open Source, Proprietary Software.

It’s always the same; no sooner do I explain the advantages of development and business model of free software / open source than someone tries to counter my argument by saying: “You only say that because you’re a university lecturer and you have no idea of business... If you think free software is such a good idea, why don’t you teach for free?” [13].

One of the things that seem to be very hard for computer professionals to accept is that there is a good living to be had by selling services, and that we software developers really need to take on board the fact that we are working in a service market and need to focus our efforts towards that market if we want to aspire to the benefits that other professions reap. This requires us to bring our practices in line with those of other professions.

How the other half lives

Whenever I chat with computer professionals, when the conversation turns to the quality of life of lawyers, doctors, accountants, financial advisors, architects, etc. compared with IT developers, they all say, without hesitation, that it is far better than theirs, both in terms of money and social prestige.

“They really live well. They can charge good money for their work, based on set rates which no one tries to beat down. Totally unlike how computer professionals are treated...”

But what do all these professions have in common? All of them are based on information which is public and accessible at a very low cost. The laws and jurisprudence used by lawyers are public or easy to access; there are even firms that make money by gathering and collating information in different ways.

Doctors simply apply their knowledge of medical science and current protocols which are, of course, fairly widely known and accessible to nearly everyone (pharmaceutical sales reps also form an incredibly effective knowledge dissemination network).

As for accountants and financial advisors much the same applies. Their work is based on an exhaustive knowledge of laws, rules, regulations and the state of the markets, all of which information is also available to anyone who is interested in finding it.

However people continue to hire lawyers, accountants, and architects, and go to the doctor at the first sign of flu. Why? Because those people are experts in their respective fields, they have studied them for a considerable length of time and/or they have a fair amount of experience in them and know how to apply that knowledge in order to solve problems.

In spite of having access to hundreds of Google’s and databases of medicine, law or stock market reports in real time, we will go on using these professionals...

... and why should the computer profession be any different?

Why is it that an architect charges by the hours he puts in and what he most wants is for his buildings to be visited and admired by as many people as possible? Is he mad? Doesn’t that mean that everyone could copy him and take away his source of income? Wouldn’t it be better if he charged a commission for every visitor or copy made of his designs?

If laws are public and we all need to be aware of them, why should we have to spend a fortune on lawyers? It doesn’t make sense. Shouldn’t lawyers work for practically nothing? Or shouldn’t they keep the laws secret to prevent people from muscling in on their territory and forcing their fees down? Imagine, if you will, the following headline: “The bar association asks for public access to laws [1] and to legal information [2] to be restrict-
ed to prevent unlicensed practitioners and unfair competition from people who defend themselves in court thereby causing lawyers to lose money”.

It sounds totally absurd. But it could be argued that it’s a special case as the laws are drawn up by States and paid for by all of their citizens.

So let’s look at another example: “The medical association asks for access to medical formulas and protocols [3] to be restricted because otherwise the medical profession will die out due to low salaries and a lack of patients”. Yes, this would be totally ridiculous and unacceptable in any western society.

In this case it could also be argued that doctors can save lives, or lose them, and therefore it’s not the same situation.

While it can also be argued that computers fly airplanes, manage medical histories, conduct biological and genetic analyses, CAT and ultrasound scans, etc., I’ll assume for the

Ricardo Galli-Granada has been a lecturer at the Universidad de las Islas Baleares, Spain, since 1992. He holds a BSc in Computer Engineering and is a Doctor of Computer Science (his doctoral thesis was entitled “Data Consistency Methods for Collaborative 3D Editing”). He has worked in the field of computer graphics and CSCW (Computer Supported Collaborative Work). He has published more than 40 articles in national and international conferences, journals and chapters of books and he has also taken part in and/or led several European R&D projects. He was the co-founder of the service provider Internet Atlas-IAP S.L. and has directed and coordinated the development of digital editions of 6 newspapers in the Balearics and several e-commerce projects. He is currently collaborating on a number of free software development projects. <gallir@uib.es>
moment that the example is not entirely satisfactory and I’ll try to find another one.

How about this one?: “The Spanish Association of Architects asks for the abolition of the obligation to publicly exhibit plans and scale models of public works. They argue that not only does it hinder innovation, but it also endangers the livelihood of architects (and engineers) by encouraging unlicensed practitioners and unfair competition. The association has the support of Frank Gehry, the designer of Bilbao’s Guggenheim Museum and inventor of a new architecture based on curves and space-age metals. Gehry not only supports the association’s request but also asks for access to the entrance to the museum and the surrounding area to be restricted to prevent the indiscriminate plagiarism of his innovative designs by other architects with less training and no ability to innovate. The fact that the museum is open to all and sundry has caused me to lose money. If projects of this nature become commonplace, the architect’s profession will be in serious trouble. I would say that the whole industry, right down to construction civil engineering, is also at risk’, said Frank Gehry about the work he carried out for the Guggenheim museum”.

All right, so the above example is silly and unbelievable. How about this one then?: “The Spanish Association of Computer Engineers asks for web pages’ HTML code to no longer be made accessible to the general public. They argue that not only does it hinder innovation but it also endangers the profession of computer professionals by encouraging unlicensed practitioners and unfair competition. The association has the support of Tim Berners-Lee, the designer of web architecture and the HTML and HTTP standards. Berners-Lee does not only support the association’s request but he also asks for access to web page source code to be restricted to prevent the indiscriminate copying of his innovative designs by other computer professionals with less training and no ability to innovate. The fact that the system is so widely used has caused me to lose money. If systems of this kind become commonplace, the computing profession will be in serious trouble. I would say that the whole IT industry is also at risk’, said Tim Berners-Lee about the work he has done for W3C”.

Does this seem likely? Does it seem reasonable?

Personally, I think that the argument is not unlike the one used by many computer professionals who disagree with the idea of free software. Despite the fact that precisely the standards and the freedom of access to HTML source code – and the possibility of copy-and-pasting – have been decisive factors, not only in the growth of the Internet, but also in the standardization of quality levels throughout the website development industry.

Do you see any difference between the web designs and developments of a Silicon Valley company and the same product from a company in Sa Pobla, Mallorca? Would that have happened if HTML code had not been accessible?

Lecturers’ salaries

Moving on to the matter of my salary as a university lecturer, what do a lecturer, an architect and a doctor have in common? The answer is that all the knowledge we lecturers impart (i.e. copy) is public and accessible, as is all science. We merely copy it and impart it (sometimes, admittedly not often, our students actually understand us). ‘That is all we do, but we don’t demand commissions on our students’ future earnings as a result of the knowledge we have just imparted, nor do we impose restrictions on how that knowledge is used, copied or passed on. It becomes the property of each individual student.

So, why do we lecturers get paid? Simply because we sell a service, just like a doctor or a lawyer. There are people who need to study in order to meet certain requirements in order to obtain a professional qualification, and perhaps it’s easier to go to classes and listen to a lecturer and go to laboratories (or do it on the Internet) than to do it privately at home on a self-study basis. You may now be thinking why do we all have to pay university lecturers? But there are also private university and private university lecturers, and sometimes they earn more than the public university lecturers do. And the consumers or customers sometimes pay small fortunes to take engineering, bachelor or masters degrees at private universities.

In short, there is a market and demand, and these give rise to businesses (perhaps public universities are playing their role of ensuring universal access and quality merely by demanding research work from their students.)

There are tens of thousands of tutorials in Word [4] or Excel [5], but there are more and more private centres offering introductory courses to those packages (a question: would those teachers’ lives be very different if the market demanded OpenOffice [6] rather than Microsoft Office?).

In other words, there is a demand for teachers and that’s why we exist. Only because of the demand teachers do not exist by divine grace – it has been that way for hundreds of years, in spite of the enormous changes there have been in science and education). If the market ceased to demand education, there would be no teachers or they would have to emigrate to other countries where they were needed. As happens in many poor or third world countries. Or as has happened to me, personally: I had to emigrate from Argentina to be able to be a doctor of computer science and a university lecturer (the free market is no guarantee that we will be able to pursue our chosen profession).

There were those who said that the arrival of books, with all that meant in terms of the popularization of education, would cause sages (what we would nowadays call experts or scientists) and the whole education system (for the privileged few) to disappear, but in fact exactly the opposite happened. The dissemination of knowledge, culture and science has not only increased the demand for lecturers, teachers and scientists, but it has also improved their social status.

Never before has there been such a demand for a strong and competitive education system, at least in the leading countries (which not uncoincidentally have the best education and scientific systems...).

But IT is a special case...

So we can agree that the work of architects, doctors... even teachers, and even most scientists who only invent or discover something of little significance in their entire working lives (even those who work in private laboratories) is based on public knowledge and that only some of them, relatively very few, make small innovative contributions to industry as a whole.

In spite of this they are well paid and on average enjoy a fairly good quality of life, but it would appear that computer professionals think that the same business model (as represented by free software) applied to IT is unsustainable.

Most computer professionals complain about the number of hours they work, about the concentration that programming requires (it has been scientifically proven; programming requires a great deal of concentration and is very tiring work), about the pressure they are put under to meet deadlines. Or about how difficult it is for self-employed professionals to find customers who will pay the prices that they ask and are able to recognise the quality and reliability that they provide.

Apart from my academic experience [7], I have also dabbled in the business world [8]: I have been co-founder and partner of a tech company [9], and I have to say that yes, the IT market is a very tough one; I’ve been there. But why the difference with other professions? Are we worse people? Are we more stupid? Are they cleverer?
I prefer to think that we belong to a field of engineering which has still to reach full maturity and that we have not yet been able to fully formalize and rationalize the whole process of software production and maintenance.

Question: won’t we have a problem with the traditional development model?
The immaturity of software engineering may have contributed to the lack of maturity of the IT industry. You need to remember that the proprietary software package market (and the sale of operating systems as a separate product) is barely 25 years old. Prior to that software was not a product you could buy as a normal consumer product.

In my opinion that way the software industry is being developed creates some serious problems:

• As software is highly recyclable, replicable and transferable, to make proprietary software marketable as a physical product artificial restrictions are created, such as intellectual property law or software patents that penalise copying without express authorisation.

• In proprietary software we invent the wheel over and over again and it doesn’t always come out round since it can rarely be compared with competitors’ products.

• There is a serious contradiction in that, as they are secret, a large proportion of proprietary programmes may be in violation of the very intellectual property laws or patents that allow them to be marketed.

• As it is very easy to change computing standards, at least much easier than it is for a power supplier to change from 220V and 50Hz, or for Canal+ to change to PAL, it is much easier to create captive markets (customer lock-in) and thereby either do away with entry barriers or raise them even higher.

• The intrinsic properties of infinite replicability (zero marginal cost) and software transfer make it easy for monopolies and non-competitive markets to be established, ones with non competitive pricing, that is, prices that bear no relation to the production costs of the software (you only have to look at Microsoft’s earnings, despite participating in markets which generate huge losses, such as the XBox).

• Monopolies take advantage of the above-mentioned problems and of the network effect [10] generated by the sheer number of users their systems have, which allows them to define their own standards (or the commoditization of protocols [11]) which in turn extends the life of the monopoly.

• As the major markets are controlled by the monopolies and due to the replicability of software, very few programmers are actually employed. You only have to consider that Microsoft, one of the largest companies, with over 90% monopolies covering two very important areas, more than 400 million Windows licences sold, which is where the biggest profits come from, with 200 millionaires among its staff... employ a mere 10,000 programmers, approximately 0.4% of all the programmers currently employed in the USA (some 2.5 million in total) and 0.1% of the estimated total number of programmers worldwide.

• If it were not for free software, or open code programmes, only the smallest minority of technicians would have access to the knowledge of the basic techniques used in computing technology (operating systems, databases, text processors, viewers, calculating systems,...).

• As the tech industry is mainly centred around very few companies which concentrate all their programmers on American campuses, the geographical concentration is also very high and consequently the regionalization of profits in these new industries is practically non-existent. Basic developments can only be made at the top of the technological pyramid (SQL, personalization of ERPs and CRMs, spreadsheets, web pages, applications based on third party middleware, etc.) which makes regional SMEs think more than twice before embarking on RDI projects.

• As there is no motivation for companies to invest in research or risk money on innovation, the technological gap with the countries in which the major IT industries are concentrated (principally the USA and some Scandinavian countries) is growing wider by the day.

• Unless there is investment in research and innovation, the demand for specialised engineers or trained researchers is practically non-existent. This is what causes computer professionals to complain about their low salaries; complaints from the business community that universities teach complicated subjects which there is no call for, such as database engines (sic [12]!); academic failure, the high dropout rate of students of computer sciences, the low demand for university qualified engineers, the zero demand for trained research staff...

How do computer professionals survive in such a hostile market?

How can they:

• By carrying out relatively simple and high level developments that rely entirely on outside technologies and which cannot be adapted (or even revised or studied) from the technologies supported by home grown developments.

• By ‘knocking on doors’ to find customers.

• By continually dropping your prices in order to convince those customers to come on board.

• By trying to stop any more or less well trained computer professional from jumping ship to another company which will pay him/her 200 euros more every month.

• Sick to the stomach when you have managed to persuade someone to pay you 6,000 euros for developing a complete management programme, only to discover that some consultancy firm with an impressive English sounding name has billed the same for just twenty hours of work involving the drafting of a report advising the use of SAP or Oracle, which was done by someone on work experience.

• Finally, when you have found a good customer who wants you to install 100 computers, 50% of the software budget goes to a couple of American companies whose sole contribution has been to send you a box of CDs, a plastic card which says “Certified Engineer” and four manuals in PDF describing the programming and basic set up of the system.

• And for the coup de grace, the freelance designer of the web pages you made for the company as a freebie sends you a bill for practically half of what’s left of the money. Meanwhile, computer professionals earning a pittance are praying that things won’t get any worse.

But we shouldn’t lose sight of the basics...

... how do the customers fare? Well, even worse,

• with systems which fail for want of patches or maintenance,

• a constant renewal of licences, either software of hardware to keep systems up to date,

• tied to a supplier which prevents them from changing even when there are better technical and economic offers on the market,

• total ignorance of what they are being sold or what is being installed, which prevents them from ensuring that they have technological independence and are able to recover data to migrate to rival systems,

• complete renewal of systems every so many years due to the disappearance of the company that developed them or maintained the previous technology,

• etc. etc.

All the abovementioned reasons are why the prestige and ‘social status’ enjoyed by computer professionals is currently at their lowest ebb.
So, what now?
The scenario is depressing, but free software / open source may provide the solution to many of these problems. Especially if we accept that our profession is just that, a (liberal?) service profession, just like lawyers or architects.

And maybe it’s better to dream of being a comfortable middle class professional, like the dozens of lawyers or doctors that we all know, than dream of being the one and only Bill Gates that we see on the telly.

In another article I intend to go into some more issues, such as the reason behind the inevitability of the existence of free software, the other, non-monetary motivations of many FS developers, etc.... (but this article is hefty enough for now :-).  

Translation by Steve Turpin

Note: Further information on the subject can be found, in Spanish, at <http://bulma.net/body.phtml?nIdNoticia=1961>.

References (all of them in Spanish)