Software Engineering for Large-Scale Multi-Agent Systems – SELMAS 2003: Workshop Report

Alessandro Garcia[1], José Sardinha[1], Carlos Lucena[1], Jaelson Castro[2], Júlio Leite[1], Ruy Milidiú[1], Alexander Romanovsky[3], Martin Griss[4], Rogério de Lemos[5], Anna Perini[6]

[1]Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil
[2]Federal University of Pernambuco (UFPE), Brazil
[3]University of Newcastle Upon Tyne, UK
[4]University of California - Santa Cruz, USA
[5]University of Kent, UK
[6]ITC-irst, Italy

Abstract

This paper is intended to sum up the results of the Second International Workshop on *Software Engineering for Large-Scale Multi-Agent Systems* (SELMAS'03) held in Portland, Oregon, USA, May 3-4, 2003, as part of the *International Conference on Software Engineering* (ICSE'03). The main purpose of this workshop was to share and pool the collective experience of people, both academics and practitioners, who are actively working on software engineering for large-scale multi-agent systems. The call for papers elicited some 26 submissions, of which 19 papers were accepted for presentation. A selected set of the workshop papers and invited papers are to appear in the 2nd edition of the book *Software Engineering for Large-Scale Multi-Agent Systems* (LNCS, Springer, 2003). The workshop consisted of an opening presentation, several paper presentations organized into five sessions, and three panels. During the workshop we informally reviewed ongoing and previous work and debated a number of important issues. The SELMAS'03 Web page, including the papers and the electronic version of this report, can be found at <www.teccomm.les.inf.puc-rio.br/selmas2003>. We begin by presenting an overview of our goals and the workshop structure, and then focus on the workshop technical program.

1. Introduction

Advances in networking technology have revitalized the investigation of agent technology as a promising paradigm for engineering complex distributed software systems. Nowadays, the agent technology has been applied in a wide range of application domains, including e-commerce, human-computer interfaces, telecommunications and concurrent engineering. In general, software agents are viewed as complex objects with an attitude. Like objects, agents provide a specific set of services for their users. In fact, objects and agents exhibit several points of similarity. However, the development of multi-agent systems (MASs) poses challenges for software engineers since software agents are inherently different abstractions than objects.

A single software agent is driven by goals, knowledge and a number of behavioral properties, such as autonomy, adaptation, interaction, collaboration, learning and mobility. While these features introduce additional complexity to the different phases of the software process, there are many techniques for dealing with individual agents or systems built using only few agents. Unfortunately, existing software engineering approaches still are unable to cope with the complexity of large MASs. This is so because software engineering for MASs is still in its infancy. MAS features are now being applied to the development of large industrial software systems. Such systems involve hundreds, perhaps thousands of agents and there is a pressing need for software engineering techniques that allow their complexity to be effectively managed. Rigorous methods also are required to guide the process of MAS development. Without appropriate development techniques and methods, such systems will not be sufficiently dependable, trustable, extensible and easy to comprehend nor will their components be reusable.

The complexity associated with large MASs is not straightforward and involves numerous facets and dimensions. When a huge number of agents interact over heterogeneous environments, various phenomena occur which are not as to easy to explain as the behavior of few agents in a closed environment. As the multiple software agents become highly collaborative, new problems emerge. It makes their coordination and management more difficult and increases the probability of manifestation of heterogeneous exceptional situations, security holes, privacy violations, unexpected global effects and so on. Moreover, since users and software engineers delegate more autonomy to their MASs, and put more trust in their results, new concerns arise in real-life applications. Commercial success for MAS applications will require scalable solutions based on software engineering approaches in order to enable reuse and effective deployment. However, many existing agent-oriented solutions are far from ideal; in practice, the systems are often built in an ad-hoc manner and are error-prone, not scalable, not dynamic and are not generally applicable to large-scale MAS.

The above considerations motivated the organization of the SELMAS'03 workshop. The main goals of this workshop were:

1. to discuss the interplay between agents and objects from a software engineering viewpoint,
2. to understand those issues in the agent technology that complicate or improve the production of large-scale distributed systems, and
3. to provide a comprehensive overview of software engineering techniques that may successfully be applied

to deal with the complexity associated with realistic multi-agent software.

Other particular interests of the workshop were to collect experience reports regarding empirical studies, identify best practices for MAS development and to establish a research agenda for those researchers interested in multi-agent software engineering. The workshop brought together researchers interested in pushing the frontier in this important and burgeoning area, and practitioners who have experience with MAS development that can help guide their research. The workshop consisted of an opening presentation, three panels and five paper sessions, organized around some of the key themes that emerged from the position papers. The paper sessions were introduced by brief presentations and continued with general discussion.

2. Workshop Proceedings and Program Committee

The papers were collected in the ICSE workshop proceedings [1], and through the SELMAS'03 Web site. The Program Committee (PC) was composed of the following members:

Alessandro Garcia (PUC-Rio – Brazil) - **Chair**
Alexander Romanovsky (University of Newcastle - UK) - **Chair**
Anand Tripathi (University of Minnesota - USA)
Andrea Omicini (University of Bologna - Italy)
Andrés Díaz Pace (UNCPBA University - Argentina)
Anna Perini (University of Trento - Italy)
Awais Rashid (Lancaster University - UK)
Brian Henderson-Sellers (University of Technology-Australia)
Bruno Schulze (LNCC - Brazil)
Carlos A. I. Fernandez (Politechnical University of Madrid-Spain)
Carlos Lucena (PUC-Rio – Brazil) - **Chair**
Catholijn Jonker (Vrije Univ. Amsterdam - The Netherlands)
Cecilia Rubira (UNICAMP - Brazil)
Dan Marinescu (Central Florida University - USA)
Donald Cowan (University of Waterloo – Canada) - **Chair**
Eric Yu (University of Toronto - Canada)
Evandro de Barros Costa (UFAL - Brazil)
Federico Bergenti (University of Parma - Italy)
Franco Zambonelli (Univ. of Modena and Reggio Emilia - Italy)
Gerd Wagner (Eindhonven Univ. Technology – The Netherlands)
Gerhard Weiss (Technical University of Munich - Germany)
Gustavo Rossi (Universidad Nacional de La Plata - Argentina)
Jaelson Castro (UFPE – Brazil) - **Chair**
Jean-Pierre Briot (Laboratoire d'Informatique Paris 6-France)
John Debenham (University of Technology - Australia)
José Alberto R. P. Sardinha (PUC-Rio – Brazil) - **Chair**
José Carlos Maldonado (USP S. Carlos - Brazil)
Juergen Lind (AgentLab - Germany)
Julio Leite (PUC-Rio - Brazil)
Katia Sycara (Carnegie Mellon University - USA)
Liz Kendall (Monash University - Australia)
Luiz Cysneiros (York University - Canada)
Marco Mamei (University of Modena & Reggio Emilia - Italy)
Marcus Fontoura (IBM Almaden Research Center - USA)
Marie-Pierre Gervais(Laboratoire d'Informatique Paris - France)
Marie-Pierre Gleizes (Universite Paul Sabatier - Toulouse)
Markus Endler (PUC-Rio - Brazil)
Martin Fredriksson (Blekinge Institute of Technology - Sweden)
Martin Griss (University of California, Santa Cruz - USA)
Michael Huhns (University of South Carolina - USA)
Michael Stal (Siemens - Germany)
Michael Weiss (Carleton University - Canada)
Olivier Gutknecht (LIRMM - France)
Paolo Giorgini (University of Trento - Italy)
Paulo Alencar (University of Waterloo – Canada) - **Chair**
Robert Kessler (University of Utah - USA)
Rogério de Lemos (University of Kent - UK)
Ruy Milidiu (PUC-Rio - Brazil)
Tom Holvoet (Katholieke Universiteit Leuven - Belgium)
Tom Maibaum (King's College London - UK)
Van Parunak (Altarum Institute - USA)
Walt Truszkowski (NASA - USA)
Yannis Labrou (Fujitsu Laboratories of America, Inc.)

3. Workshop Organization and Structure

The organization was under the responsibility of the organizing chairs José Sardinha, Alessandro Garcia, Carlos Lucena (PUC-Rio), Jaelson Castro (UFPE, Brazil), Alexander Romanovsky (University of Newcastle, UK), Paulo Alencar and Donald Cowan (University of Waterloo, Canada) and with the assistance of the PC. Two full days were allocated for the workshop (May 3-4, 2003). There were about 45 participants who contributed, largely with position papers, which were reviewed and revised before the workshop. We received some 26 submissions from different countries. We selected 19 papers for presentation in the workshop. Each paper was reviewed by at least four members of the PC or additional reviewers; the final selection was made by the workshop organizers based on the evaluation forms. The presented papers were chosen because they offered different or novel perspectives on the workshop topics and because they had a high potential for generating issues that would stimulate the discussions. An additional description of the selection process, as well as all the participants' position papers, can be found at the SELMAS Web site.

The meeting provided a forum for the exchange of ideas on case studies and diverse approaches to the development of MASs. In preparation for the workshop, participants were requested to read all other submissions and asked to prepare a clear position statement and questions that were likely to stimulate discussion. Moreover, each presenter tried to identify open questions that could provide the basis for further research in the coming years. The talks were common to all participants, providing a sense of thematic unity by addressing different important topics in MAS engineering theory and practice. The quality of the presentations at SELMAS'03 was high and triggered a highly interesting discussion between workshop participants – whom we sincerely want to thank for their active participation and the level of their contributions to the debate. Interactions between the participants were lively and the discussion sessions often ran overtime.Furthermore, workshop participants discussed the benefits of future collaborations during the lunch and coffee breaks.

The workshop was structured into the following parts:

- An opening presentation by Carlos Lucena was the starting point and introduction for the morning and the afternoon sessions. He reported on the meeting's topics and goals and the workshop organization process (see section 4).

- Five technical sessions provided the framework to present theoretical and practical issues concerning MAS engineering. The first session addressed the interplay between agents and objects from a software engineering viewpoint. The second session was dedicated to presenting development methods for engineering large-scale MASs. The third session introduced interesting examples of large-scale MASs. The fourth session was about dependability and QoS aspects in the context of MAS development. Some frameworks for MAS development and their specifics were presented in the fifth technical session. At the end of each presentation, time was reserved for discussion. To maximize time, we appointed a chair for each session to coordinate the discussions. The most important topics of each session are briefly summarized in section 6.
- Three interesting panels addressed important workshop topics. The panelists answered questions from the audience and discussed with each other. Unfortunately, there was too little time to resolve many open issues. More information about the panels and the topics discussed is given below.

Following this successful workshop a number of workshop papers have been selected for extension and publication in a forthcoming special LNCS volume. Moreover, we will publish some invited papers in this special volume. It is also hoped that it will be possible to hold a third edition of the workshop as part of the ICSE 2004.

4. Opening Presentation: Setting the Stage

SELMAS'03 began with a kick-off presentation by Carlos Lucena. Lucena established a brief overview and the motivation for the workshop. Lucena also explained the selection process for the LNCS volume. The foils of this opening presentation are available at the SELMAS Web site.

5. Workshop Presentations

As we explained above, 19 papers were accepted for presentation. Unfortunately, two of the speakers were not able to travel to Portland. In this sense, we had actually 17 paper presentations during the workshop. In the first day, there were seven presentations in the morning and three presentations after the lunch break. On the second day, two speakers presented their work in the morning and there were five more presentations in the afternoon. Each speaker had 20 minutes per presentation, followed by 10 minutes for discussion. The papers and their authors were as follows. Summaries of these presentations are presented in the following section of this workshop report.

- *Extending UML to Modeling and Design of Multi-Agent Systems,* Krishna Kavi, David Kung, Hitesh Bhambhani, Gaurav Pancholi, Marie Kanikarla.
- *Agents and Objects: An Empirical Study on the Design and Implementation of MASs,* Alessandro Garcia, Claudio Sant'Anna, Christina Chavez, Viviane Silva, Carlos Lucena, Arndt von Staa.
- *An OO Framework for Building Intelligence and Learning properties in Software Agents,* José Alberto Sardinha, Ruy Milidiú, Carlos Lucena, Patrick Paranhos.
- *An Agent-Based Requirements Engineering Framework for Complex Socio-Technical Systems,* Paolo Donzelli, Paolo Bresciani.
- *TROPOS-T: Extending the Tropos Methodology to Include Requirements,* Jaelson Castro, Rosa Pinto, Andrea Castor, John Mylopoulos.
- *Lexicon Based Ontology Construction,* Karin Breitman, Julio Cesar Leite.
- *On Security Requirements Analysis for Multi-Agent Systems,* Paolo Bresciani, Paolo Giorgini, Haralambos Mouratidis.
- *From Static to Dynamic and back: Three Approaches for Role Composition,* Elke Steegmans, Kurt Schelfthout, Tom Holvoet.
- *InQuality: A Multi-Agent System for Enterprise Quality Management*, Andres Diaz Pace, Marcelo Campo, Alvaro Soria, Mario Zito.
- *A Multi-Agent System for Analyzing Massive Scientific Data,* Joel Reed, Thomas Potok, Mark Elmore.
- *On Monitoring and Steering in Large-Scale Multi-Agent Systems,* Takahiro Murata, Naftaly Minsky.
- *A Proposition for Exception Handling in Multi-Agent Systems,* Frédéric Souchon, Christelle Urtado, Sylvain Vauttier, Christophe Dony.
- *Object-Oriented Modeling Approaches to Agent-Based Cross-Organizational Workflow Systems,* M. Brian Blake, Hassan Gomaa.
- *A Pragmatic Agent Architecture for Layered Component Reuse using Subsystems,* Steven Fonseca, Martin Griss.
- *A Multi-Agent Platform for Reconfiguration, Adaptation and Evolution of a System at Architectural Level,* Amar Ramdane-Cherif, Samir Benarif, Nicole Levy.
- *Farm: A Scalable Environment for Multi-Agent Development and Evaluation,* Bryan Horling, Roger Mailler, Victor Lesser.
- *Software Engineering Challenges for Mutable Agent Systems,* Ladislau Boloni, Majid Khan, Xin Bai, Guoqiang Wang, Yongchang Ji, Dan Marinescu.

6. The Sessions

As mentioned above, there were five sessions of presentations and discussions. Each of the sessions was organized according to common themes in the position papers. The session summaries as produced by the respective session chair are presented below.

Session 1: Interplay between Agents and Objects
Chair: *Prof. Anna Perini (ITC-irst, Italy)*

Three papers have been presented in the first session (25 minutes for presentation, plus five minutes for questions).

- Hitesh Bhambhani presented the first paper. The authors have proposed an UML-based framework for modeling, analysis and construction of agent-based systems, which refer to the Belief Desire Intention (BDI) paradigm. Questions concerned: environment dynamicity issues; how objects (which are inherently reactive) can be used to build proactive agents; open system issues, such as those related to the possibility of PDAbased system to enter into the environment of the MAS.

- Alessandro Garcia presented the second paper. He presented an empirical study that compares the maintenance and reuse support provided by abstractions associated with two OO techniques for MAS development, i.e. aspect-oriented development and pattern-oriented development. Questions concerned: the relationship between aspect and pattern instantiations (what if aspects are considered as patterns at the code level?); contribution of the work respect to the definition of a new methodology (or a standard); the effectiveness of the used metric and possible results on performance measurements.
- José Sardinha presented the third paper. Sardinha presented an extension of OO frameworks that has been previously developed for designing MAS. The extension (called MAS-RL) allows implementation of learning properties in software agents by exploiting Machine Learning (ML) techniques called "Reinforcement Learning". An application of the framework to Trading Agent Competition was discussed. Questions concerned: the meaning of the term services in the framework; the scalability of GAIA (i.e. of the original framework); the relationship between agents and ML components (are they wrapper?); the extensibility of the framework respect to other ML techniques and the possibility of the developer of a new MAS-RL application to select those to be used.

Session 2: Development Methods for Large-Scale MASs
Chair: *Prof. Julio Leite (University of Toronto, Canada)*

There were five presentations during this session:

- An Agent-Based Requirements Engineering Framework for Complex Socio-Technical Systems
- TROPOS-T: Extending the Tropos Methodology to Include Requirements Traceability
- Lexicon Based Ontology Construction
- On Security Requirements Analysis for Multi-Agent Systems
- From Static to Dynamic and back: Three Approaches for Role Composition

The first presentation given by Paolo Bresciani dealt with the issue of modeling early requirements and how this modeling is used to drive an agent based requirements engineering process. Since the strategy is based on the i* framework, there was an emphasis on the distinction of soft and hard goals. Questioned about the difference between these types of goals, Paolo explained that the softgoals are the ones related to the aspects that are not functional.

Jaelson Castro was responsible for the second presentation. He emphasized the importance of dealing with early requirements and gave an overview of the Tropos approach to the construction of agent based systems. The papers emphasized the importance of traceability in the process, and detailed a meta-conceptual model to deal with traceability. The meta concepts are stored in traceability matrices. Jaelson was questioned about the level of granularity necessary for this type of traceability. He noted that this could be adjusted in the model.

Julio Leite presented the idea that elicitation of ontologies can be performed by a previous construction of the lexicon of the application language. Since ontologies will be necessary by a series of web services, a method was presented on how to transform lexicons onto ontologies, with these ontologies described in the Oiled editor. Asked about the distinction between connotation and denotation, the two different types of entries of the lexicon, he explained the difference and stressed that the connotation provides contextual information. Regarding quality aspects, a question was posed on how to guarantee that the entries are correctly described; he mentioned that the lexicon has a well-defined validation and verification process.

Paolo Bresciani presented a paper dealing with security requirements. He presented an extension to Tropos in order to explicitly label security softgoals in order to make it more visible and ready to be treated as a first class concept in the Tropos requirements models. In the networks of goals the idea of labeling with numbers the relations among security softgoals and other requirements in order to be able to estimate the issues related to complexity and criticality of the security requirements was presented. Questions were posed regarding the issues of executable models, like statecharts, for instance. Paolo argued that the proposal did not required execution but proved a very early analysis of the networks of goals in terms of security issues.

Kurt Schelfthout presented the last paper of the Session. He described design issues in defining agents as compositions of roles. He also presented three approaches for composing multiple roles and described a basic taxonomy for role composition. The approaches were illustrated using a simple example.

Session 3: Large-Scale Multi-Agent Systems
Chair: *Prof. Ruy Milidiú (PUC-Rio, Brazil)*

This session had two presentations:

- InQuality: A Multi-Agent System for Enterprise Quality Management
- A Multi-Agent System for Analyzing Massive Scientific Data

InQuality is a framework to help building XML document driven applications that support Enterprise Quality Management. The proposed framework provides workflow management features in order to operate with concepts such as processes, documents, roles, activities, assignments and resources. Through an extensive case study the authors discussed the gains on separation of concerns that results from the MAS approach.

Using the Oak Ridge Mobile Agent Community Framework, the authors developed a large-scale distributed MAS. The system goal was to help validate massive datasets generated when simulating complex physical phenomena. An experiment within the Terascale Supernova Initiative was used to illustrate the MAS advantages. Upon request, 800 agents work together to produce a visual representation of the dataset that enables the user to validate the simulation results.

Session 4: Dependability and QoS in MASs
Chair: *Prof. Alexander Romanovsky(University of Newcastle, UK)*

The second day of the workshop started with a session on Dependability and QoS in Large-Scale MASs. There were two presentations during this session:

- On Monitoring and Steering in Large-Scale Multi-Agent Systems
- A Proposition for Exception Handling in Multi-Agent Systems

The first talk of the session was delivered by T. Murata. N. Minsky, the second co-author of this work, attended the workshop and helped the attendees in clarification of a number of issues raised. The focus of this work is on proposing a novel paradigm supporting and enforcing coordination and control in large and open MASs. The idea is to use the Law-Governed Interaction mechanism to express specific policies related to monitoring and steering. A working example of a department store employing a number of buyers and a manager was used throughout the talk to demonstrate the main ideas of the approach proposed.

The second talk of the session on was delivered by F. Souchon. The talk started with an outline of the main requirements for the exception handling mechanisms suitable for developing complex MASs. Next, the speaker introduced the novel mechanism, in which, depending on the application needs, exception handlers can be associated with a service, an agent and/or a role. A simple set of rules describing how exceptions can be propagated and what the exception contexts for each possible exception are was demonstrated using a travel agency example. Some experimental work on implementing this mechanism in the MadKit multi-agent system was presented. The approach allows complex systems consisting of a number of cooperating agents to be developed with incorporating exception handling capability in a disciplined and regular fashion.

Session 5: Development Frameworks for Large MASs
Chair: *Prof. Hassan Gomaa (George Mason University, USA)*

In this session, architectures and frameworks for MAS development together with lessons learned were presented. The intention was to help software engineering researchers get a feeling for the critical issues to consider in the construction of MAS frameworks. This session had five presentations:

- Object-Oriented Modeling Approaches to Agent-Based Cross-Organizational Workflow Systems
- A Pragmatic Agent Architecture for Layered Component Reuse using Subsystems
- A Multi-Agent Platform for Reconfiguration, Adaptation and Evolution of a System at Architectural Level
- Farm: A Scalable Environment for Multi-Agent Development and Evaluation
- Software Engineering Challenges for Mutable Agent Systems

Hassan Gomaa presented a large-scale agent-based architecture to support a distributed services environment. In addition, he introduced a software engineering approach towards the programming, configuration and operational control of the agents that manage processes in a cross-organizational workflow environment. Martin Griss reported on his experience in the development of an MAS framework. He argued that the proposed architecture is more flexible and extensible than current MAS frameworks provide. This architecture includes a component decomposition framework and infrastructure that guides application developers to decompose agent behavior in reusable ways. At the highest level, agents are constructed from reusable subsystems. Subsystems interact by an event-based software bus that acts as the central nervous system of an agent.

In her talk, Nicole Levy presented the conception and implementation of a platform for reconfiguration, adaptation and evolution of MASs. This platform can evaluate an MAS architecture with respect to some quality attributes to improve its structural and behavioral properties. Roger Mailler introduced the Farm environment for simulating large-scale MASs. This environment uses a component-based architecture to support the researcher to easily modify and augment the simulation. Moreover, it allows distributing the various pieces to spread the computational load and improve running time. Mailler described technical details of the proposed environment along with discussion of the rationale behind the design.

In the last talk, Ladislau Boloni addressed mutability in MASs. Mutation was a term used to indicate controllable and well-specified changes of a program at runtime. He presented the Bond system, a FIPA compliant agent framework with support for mutability. He proposed a set of extensions to the Gaia methodology to handle certain important classes of mutable systems.

7. The Panels

During the workshop three panels were organized to discuss important topics of MAS engineering. The panelists answered questions from the audience and discussed with each other. On the first day there was only one panel that discussed the development methodologies for large-scale multi-agent systems while on the second day there were two panels: the first discussed the dependability and QoS in large-scale multi-agent systems; and the second panel discussed frameworks and architectures for large multi-agent systems. The panel summaries, produced by their respective moderators, are presented below.

Panel 1: Development Methodologies for Large-Scale MASs
Moderator*: Jaelson Castro*
Panelists*: Naftaly Minsky, Carlos Lucena, Paolo Bresciani, Anna Perini.*

This was the closing panel of the first day of workshop. The growth of interest in software agents and multi-agent systems has recently led to the development of new methodologies based on agent concepts. These methodologies propose different approaches in using agent concepts and techniques at various stages during the software development lifecycle.

The moderator, Jaelson Castro, in order to stimulate the discussion, produced a list of some modeling languages and methodologies that emerged in the recent years, such as Gaia, AAII, AOR, MaSE, Message/UML, AUML, OPEN/Agent, Tropos, PASSI and Prometheus. He then emphasized that the goal of this panel was to identify, analyze, and illustrate the commonalties and distinctions across different methodologies.

The first panelist, Paolo Bresciani, focused on the issue of Expressiveness and Usability in Requirements Engineering Methodologies for Multi Agent Systems. He claimed that these are the basic questions that may help us to understand if and why agent and goal-based Requirements Engineering are a promising evolution in Software Engineering. As an example he mentioned that Tropos allows for a very expressive requirements language with the potential of capturing various aspects. However, he argued that the Requirement Engineering process in Tropos was quite complex because it had to deal with many diverse elements, compromising its usability. On the other hand, he pointed out that a simpler approach, such as REF, could be easier to be adopted by practitioners, at the expense of expressiveness. To conclude he noticed that there are no simple answers, it all depends on the domain, the application and stakeholders. The solution could be the fusion of REF and Tropos (and other methodologies) in order to take advantage from both.

The next panelist, Naftaly Minsky, focused on coordination and protocols. He began defining what he called an "open" community of agents (or MAS):

- community whose members are heterogeneous---which may be designed and maintained independently of each other, and possibly written indifferent languages; and
- community whose membership may change dynamically.

He argued that for such a community to operate harmoniously and safely, its design needs to start with a collection of global constraints, or "law", designed to govern the interaction between its various member agents. This, in some analogy to the manner in which vehicular traffic is governed by the traffic laws, and to the manner in which countries are governed by their constitutions.

He also described some of the desired properties of such laws, and how such properties are satisfied by the Law-Governed Interaction (LGI)mechanism. He claimed that there is a complementarily between such laws of a MAS, and the more detail specification of its member agents, via a methodology like Tropos.

The next panelist, Anna Perini, discussed problems that need to be faced when going from system requirement analysis to system architecting adopting an agent-oriented approach. In her presentation she discussed her experience with the Tropos methodology for the development of distributed systems such as web-based system and peer-to-peer systems for knowledge management.

She explained that systems requirements analysis is driven by the questions of the following type: What are the goals of the users? Are there alternative means to achieve them or are there goals/plans that contribute to their achievement? What are the reasons for choosing one alternative over the others? Are some of them dependent on the system-to-be? How does the system accomplish (execute) these goals (plans) for the user? Again, are there alternative ways to do this? How are them evaluated.

Anna Perini also noticed that when we move to system design we are faced by different questions, such as : How can we identify system components which ensure an appropriate level of cohesion? How can we reduce coupling among components? How can we evaluate different architectural options analyzing non-functional requirements? Is there any architectural styles that we can exploit in an effective way? How can we exploit OO/AO development environment, development skills, …?

Carlos Lucena described the ongoing work at PUC-Rio, where they have been performing research and development efforts for achieving new software engineering technologies in the context of multi-agent systems, including:

- High-Level Aspect-Oriented Design Language for MAS Development
- Metric Suite for Aspect-Oriented Development (Empirical Studies)
- Viewpoints and Goal-Driven Requirements
- The TAO Meta-model for MAS's

Work is also under way at PUC-Rio to present a new method for integrating agents into object-oriented software engineering from an early stage of design. The proposed approach encourages the separate handling of MAS concerns, and provides a disciplined scheme for their composition. The proposal explores the benefits of aspect-oriented software development for the incorporation of agents into object-oriented systems. Achieving the benefits of separation of concerns is not a trivial task and demands the use of appropriate principles from an early stage of design.

An empirical study was also conducted ad PUC-Rio to compare the maintenance and reuse support provided by abstractions associated with two OO techniques for MAS development: aspect-oriented development and pattern-oriented development. The preliminary results indicated that abstractions from the aspect-oriented approach allowed the construction of a MAS with improved separation of MAS concerns.

Another interesting work reported by Carlos Lucena was the coupling of the agent technology with the viewpoint concept. This approach intends to specify a multi-agent system using integrated views, enabling a better specification of the system's features. These views are based on a classification of the structural and dynamic aspects of a multi-agent system. Each view is modeled and documented using a notation (ANote) language for visualizing, specifying, constructing, and documenting the artifacts of a multi-agent system specification. ANote provides a set of diagrams, each one modeling a different view of a multi-agent system.

Carlos Lucena also explained that the goal of TAO (Taming Agents and Objects conceptual framework) was to provide the foundations for agent and object-based software engineering. The work included the definition of an ontology to support essential concepts, or abstractions, for developing MASs. The benefit of having a conceptual framework is to provide support for developing new methodologies, methods and languages based on the essential concepts defined and related in the framework. Each concept is viewed as candidate abstraction modeling languages, methodologies and support environments to be applied in different phases of the MAS development.

Panel 2: Dependability and QoS in Large-Scale MASs
Moderator: *Rogerio de Lemos*
Panelists: *Tom Maibaum, Arndt von Staa, Paolo Bresciani, Alexander Romanovsky.*

In order to encourage discussion, moderator Rogério de Lemos initiated the panel by defining key terms associated with its topic. Computing systems, in general, can be characterized by five fundamental properties: functionality, usability, performance, cost and dependability. While quality of services (QoS) refers to the non-functional properties of systems, dependability, in particular, refers to the ability of a computer system to deliver service that can justifiably be trusted.

Since the theme of SELMAS 2003 was the interplay between the notions of agents and objects, the key question to be asked, from a software engineering perspective, was how could an agent-based approach improve the dependability and the QoS of software systems? Between the four basic means to achieve dependability, fault tolerance may be the most relevant one from the perspective of building agent-based systems since it deals with the continuous provision of services despite the presence of faults. The basis for designing and building any fault tolerant system is the specification of the failure assumptions of its components. However, if we consider that autonomy is a key property of agent systems, it is not clear how the failure assumptions can be associated with agents since their full behavior is not known beforehand. Even if we assume that autonomy can be restricted and the worst class of failures can be associated with agents, it still is not clear whether a feasible implementation can be found without making strong assumptions about the communication between agents.

Agents' ability to adapt to environmental changes also may introduce some challenges. This is so because the uncertainties associated with the outcome of their learning process might lead to unpredictability, which might impair the ability to evaluate the dependability attributes of multi-agent systems. Moreover, if autonomy and adaptability are essential features of these systems, what type of stability criteria should be associated with these systems for avoiding divergent behavior among agents? Another property of an agent is its mental state, and if the notion of this "mental state" is more anthropomorphic than the more conventional notion of a "state," then can this mental state be observed by other system entities? If this state cannot be observed then it cannot be controlled, making the provision of fault tolerance difficult.

Based on the issues raised above, the question to be asked is what then could be the role of agents in large-scale systems? There are several ways in which agents could be employed: first, as basic building blocks, like functions, objects and components; second, as means for integrating legacy systems; and finally, in a lesser role, as a means for the provision of specific services, such as an additional system layer to collect information, inform third parties and control the allocation of resources. On the other hand, instead of restricting their applicability, another alternative would be to impose restrictions on the features offered by agents depending on the role taken by them upon building systems. Likewise, for the sake of predictability, inheritance is discarded when designing real-time systems using object-oriented technology, or a safe subset of Ada (SparkAda) is employed when building safety-critical systems.

The moderator concluded by raising three questions designed to provide the basis for the panel: from the software engineering viewpoint, how do MASs' features (e.g. autonomy, self-adaptation, intelligence, mobility, emergent behavior...) make dependability and QoS easier or more difficult than in object-oriented systems? To what extent can conventional and object-oriented dependability and QoS techniques be used in the context of the engineering of MASs? What are the challenges facing the promotion of dependability and QoS in development of MASs?

Also within the realm of dependability, Paolo Bresciani discussed security in the context of agent-oriented software engineering. He started his talk by emphasizing that security analysis should be considered early in the lifecycle, instead of after the design of the system. In order to show the feasibility of this claim, Bresciani presented an approach for dealing with security requirements using the Tropos agent-oriented software engineering methodology. The security process in Tropos consisted of analyzing the security needs of the stakeholders and the system in terms of security constraints imposed on the stakeholders and the system, identifying secure entities that guarantee the satisfaction of the security constraints and assigning capabilities to the system to help towards the satisfaction of the secure entities. Bresciani concluded that it was not easy to consider security issues in a simple way at the requirements level, hence the reason why they are usually considered at the late stages of development.

In examining the agent literature, Tom Maibaum has found two essential new concerns. One, a qualitative difference with software engineering concerns in the past — namely, agents combine a large number of underlying technologies and, thus, are very complex artifacts. The second is a real difference in relation to past software engineering concerns, i.e., the use of introspection in the implementation of an agent. The use of anthropomorphisms in agent technology may be dangerous because it might lead to an illusion of understanding of concepts that does not actually exist. For example, Maibaum reiterated that he had never found a proper definition of the concept of autonomy, at least one that can be used for the design and analysis of agents. Thus, for him the most interesting research challenge in relation to agent technology is the management of the complexity of multiple technologies referred to above. The use of coordination technologies might be an aid to deal with this complexity.

In his talk, Alexander Romanovsky considered the issue of fault tolerance and exception handling in large-scale MASs. He started by stressing that due to types of faults in MASs what is actually needed is software fault tolerance at the application level rather than software based mechanisms for tolerating hardware faults (ACID transactions, replications, atomic broadcasts, etc.). Moreover, instead of recovery techniques based on rollback, alternative solutions based on forward error recovery should be sought. One of these solutions is exception handling, and the challenge is to develop novel exception handling techniques suitable for large-scale MASs. There are several advantages in using exception handling as a means to achieve fault tolerance: separation of code and flow in terms of normal and abnormal behavior, and the recursive structuring by multiple level exception handling as a way to limit the scope of recovery. However, new problems arise due to the characteristics of MASs. Agents are autonomous; thus, they cannot report exceptions to a higher level. Agents are interactive; thus, they need the concept of scope or exception context. Agents are mobile; thus, requiring special exception handling techniques since agents can leave the location and move to another location, or the execution environment and the resources available can change on the fly. The communication between agents is asynchronous, thus requiring the decoupling of producers and consumers, and anonymous communication using event-based interactions. In order to deal with some of these limitations, several solutions are possible. Exception handling contexts should include all the agents involved in a particular exception or its respective handling, and this could be implemented by defining all the cooperating agents in a particular location or dynamically by mutual agreement. In their implementations agents should include additional information about the sort of services they provide and assumptions about their environment and other agents; this additional information could also be store in some sort of registry.

Arndt von Staa focused his talk on agents and dependability and how trust can be obtained when using agents for building critical systems. In his initial statement he said that the 40-plus years he has dealt with computers has taught him to be skeptical. To support this statement he enumerated several cases in which either computers or software were held responsible for system failures. In order to motivate the usage of agents in critical applications, von Staa described a hypothetical scenario in which individual airplanes were agents in an MAS. The collection of airplanes forms a society of agents in which airplanes and obstacles enter and leave the space of interest of the society or of a specific airplane. In this society, each airplane would be able to sense the proximity of obstacles and other airplanes, to determine the possibility of collision in the next n time units and to take evasive action to avoid collision. The question raised by von Staa was whether anyone in audience would allow themselves to fly in such system. In the second part of his talk, von Staa's raised several issues related to the validation of MASs. For example, can we prove the correctness of MASs? More specifically, do we really know how to specify them or are we able to obtain adequate models? In terms of the development process of MASs, he questioned whether we correctly understand all agent interface problems? Can we trust separately (incrementally) developed agents and their integration? His final query was how one could test MASs? He concluded his talk by stating that we must still be very humble when proposing agent-based solutions and that high risk applications should not be based on MASs, at least for the time being.

Panel 3: Frameworks and Architectures for Large MASs: Issues & Challenges
Moderator: *Martin Griss*
Panelists: *Martin Griss, David Kung, Roger Mailler, Ladislau Boloni*

This was the workshop's closing panel, charged with looking at the technical issues and challenges related to building and operating large-scale multi-agent systems. The primary question was to what degree could traditional object-oriented software engineering techniques be used to address the peculiar problems of large-scale multi-agent systems. The panelists focused on similarities and differences between objects and agents and how these differences would affect several aspects of software engineering: modeling, design, programming and reuse.

Introduction. The panel started from several questions:

- What are the problems with the use of OO abstractions for the construction of frameworks for MAS development?
- How can OO techniques (design patterns, frameworks, aspects, etc.) be used to develop MASs? How should they be augmented (e.g. AUML?)
- What are the key challenges (reuse, maintenance, scalability, interoperability, trust, privacy...) for MAS frameworks?
- How much should MAS frameworks provide support for MAS features (e.g. autonomy, self-adaptation, intelligence, mobility, protocols, emergent behavior...)?

The panelists and audience discussed a subset of the issues.

Software Engineering for Large MASs. Software engineering is the systematic application of models, methods, processes, metrics, and tools to create software "economically." A coherent Agent-Oriented Software Engineering (AOSE) approach will address: development, support, management, evolution, and scale. For large-scale MASs, we could choose to apply traditional software engineering techniques to MASs ("An MAS is just a distributed OO system"), or invent a new software engineering technology for MASs ("An MAS is all about goals and beliefs"). We could even derive new software engineering techniques for non-MASs from MAS engineering experience (e.g, WSOSE for web services).

When we say "large," we could mean number of (active) agents, number of different kind of agents, number of developers, number of hosts or variety of platform types; or all of these. When we say MAS, we are stressing the specific issues that characterize agents' Autonomy, Emergent behavior, Distributed AI, Goals, Roles and Mobility and ACL communication to make the problem more complex.

Agent-Oriented Modeling and Design Language. Development of large, complex agent-based systems requires an engineering process. An engineering process requires a modeling approach and a design methodology. A modeling approach is associated with a view of the world and a modeling language that can be used to document the perception of a given world or an application (in the given view), resulting in a model.

The modeling concepts and constructs affect the resulting modeling in terms of abstraction, understandability, ambiguity, complexity and other attributes. For example, the object-oriented paradigm views the world as consisting of objects that encapsulate states and behaviors and interact with each other through message passing.

However, agents are not (just) objects. It is commonly recognized that there are essential differences between agents and objects. Agents are characterized as autonomous, goal-oriented, situation-aware and proactive as well as reactive. Objects, on the other hand, are passive, responsibility/functionality oriented and, at the most, reactive. Since agents and objects are so different, the views that we use to perceive the world to build OO systems and agent based systems must not be the same.

Thus the modeling language that we use to document the perception of the world for OO systems and agent systems must provide modeling concepts and constructs appropriate for agent systems. Of course, one could use an OO modeling language as an agent modeling language. This is analogous to using an ER model as the modeling language for OO systems. One could even use ER diagrams and its extensions as the modeling language for agents. But this choice is even further from the ideal because ER is not even appropriate as an OO modeling language. The modeling language could be either Booch diagrams, OMT, UML or any other OO modeling languages. AUML is one such attempt to extend UML with agent-specific capabilities; for example, modifying the interaction diagrams to better document protocol alternatives.

Agent-oriented programming. Agent-oriented programming techniques should naturally rely on object-oriented programming as their foundation. Moreover, we can learn a lot from the object-oriented field in making complex concepts accessible and easy to use. The ability to participate in standard interactions (negotiations, auctions and so on) should be made just as easy as the use of abstract data types in contemporary object-oriented libraries. Agent-oriented patterns would also further the field by automating and standardizing the development and deployment of multi-agent systems. Mobility and adaptivity features are desirable; but much work needs to be done towards adequate methodologies.

Agents and Reuse. A key approach to improving quality and reducing development time for large scale MAS is systematic reuse involving event-based frameworks, flexible components and generators. Reuse will bring other benefits, too, including improving interoperability, more features and ensuring certain properties for the entire system, such as improved dependability and security and other cross-cutting aspects.

The agent-oriented reuse process includes the domain-oriented development of reusable elements, using domain analysis to identify features, structure commonality and variability needed for a range of MASs. This is then expressed as a “Kit,” using a balance of architecture, framework, components, agent-construction languages and generative tools. The new framework and components need to leverage standards based on FIPA, JAS, web-services, semantic web, JMS and JXTA. Then, in order to develop a new MAS, one simply selects, develops or generates components, which are then customized and integrated into a complete system

The reusable elements will include multiple abstraction levels and granularity, such as (A) UML models, use cases, role models, architecture and design patterns, protocols in the form of interaction diagrams, state machines, or Petri nets. Components and templates in Java or other languages, combined in an event bus architecture that will permit subsystems and components to register dynamically and support flexible communication.

Conclusion. Engineering a large-scale multi-agent system is a complex task that involves designing a system that has well defined properties while allowing agents within the system to exhibit learning, autonomy, mobility, etc. For the most part, these goals are contradictory. If we allow individual agents to do whatever they want, the system will likely stray from its original design. If, on the other hand, agent behavior rules are too restrictive, we are left with simply another engineered, distributed system. What is clear is that OOP methodologies and paradigms, although appropriate for designing intra-agent components, fall short of allowing designers to build rules for inter-agent behavior. Particularly, OOP is incapable of addressing trade-offs between system-level guarantees and allowable uncertainty in individual agent behavior.

8. Conclusions

The particular focus of this second meeting was on the role of agents and MASs in software engineering. Altogether, the workshop was a very large success due to the quality of the submitted papers, the level of participation of the audience and the profile of the panelists. SELMAS’03 achieved its goal to provide a forum for interactive discussions on the research issues of software engineering for large-scale multi-agent systems. The speakers presented items for a research agenda during several of the talks.

SELMAS’03 put researchers from software engineering together to discuss the multi-faceted issues that emerge in using MASs to engineer complex, distributed systems. Given the level of the contributions, we are confident that the workshop was useful to the multi-agent software engineering community, by providing many original and heterogeneous views on such an interdisciplinary topic as well as several attempts to pull everything together. It is our hope that SELMAS’03 provided the agent community with a forum where novel ideas and results can be shared by crossing the boundaries of the many research and application areas that meet in the agent field.

Like SELMAS, other important, related workshops have been organized to discuss research and practice on multi-agent software engineering (such as AOIS and AOSE workshops [3, 4]). As is evident from these meetings and this workshop report, work on agent-based software engineering remains to be done. There are a number of ways to learn more about current work and get involved, including:

- Visiting the workshop web site for details of ongoing work.
- Reading the position papers from the SELMAS’02 book [2] and the SELMAS’03 proceedings [1] for background information.
- Contacting any of the organizers and authors of the SELMAS papers for more information.

Finally, a high-quality set of workshop and invited papers is going to appear in the second edition of the book *Software Engineering for Large-Scale Multi-Agent Systems* (LNCS, Springer, 2004). In addition, this volume will include some invited papers. The SELMAS’04 workshop is planned for the next year at ICSE 2004. We look forward to an excellent program also in the next year.

Acknowledgements

The organizers would like to thank all those who contributed with submissions to the workshop and the program committee members who invested many hours reviewing such submissions. In addition, we thank the session chairs and panelists for the fine work in coordinating the sessions and promoting an interesting panel, respectively. Finally, we would sincerely like to thank again the

SELMAS’03 participants for their active involvement in the meeting and the level of their contributions to the debate.

References

[1] Sardinha, J., A. Garcia, C. Lucena, J. Castro, A. Romanovsky, P. Alencar, D. Cowan (Eds.). Proceedings of the 2nd Workshop on Software Engineering for Large-Scale Multi-Agent Systems. *International Conference on Software Engineering (ICSE 2003)*, Portland, USA, May 2003.

[2] Garcia, A., C. Lucena, J. Castro, F. Zambonelli, A. Omicini (Eds.) (2003): Software Engineering for Large-Scale Multi-Agent Systems. *Lecture Notes in Computer Science*, Springer, Vol. 2603, April 2003.

[3] The International Workshop series in Agent-Oriented Information Systems, http://www.aois.org/

[4] The International Workshop series in Agent-Oriented Software Engineering, http://www.csc.liv.ac.uk/~mjw/aose/