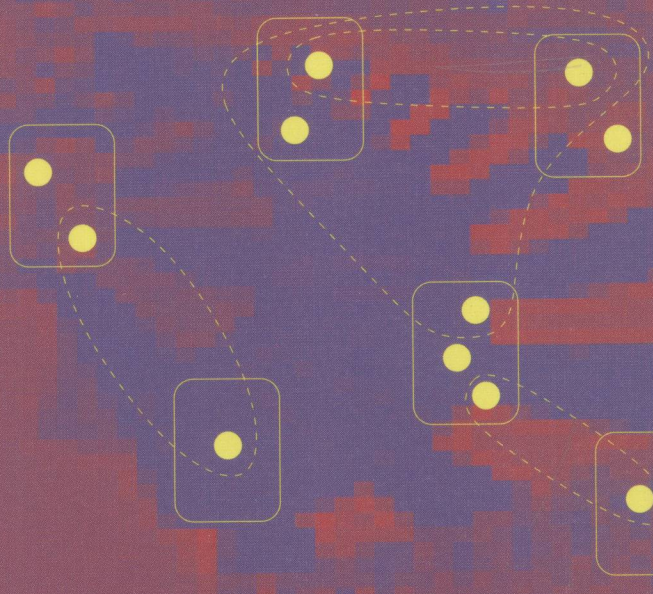
State-of-the-Art Survey

LNCS 4408

Ricardo Choren Alessandro Garcia
Holger Giese Ho-fung Leung
Carlos Lucena Alexander Romanovsky (Eds.)

Software Engineering for Multi-Agent Systems V

Research Issues and Practical Applications



Springer

Ricardo Choren Alessandro Garcia
Holger Giese Ho-fung Leung
Carlos Lucena Alexander Romanovsky (Eds.)

Software Engineering for Multi-Agent Systems V

Research Issues and Practical Applications

Springer

Volume Editors

Ricardo Choren
PUC-Rio, Rio de Janeiro, Brazil
E-mail: choren@les.inf.puc-rio.br

Alessandro Garcia
Lancaster University
United Kingdom
E-mail: garciaa@comp.lancs.ac.uk

Holger Giese
University of Paderborn
D-33098 Paderborn, Germany
E-mail: hg@uni-paderborn.de

Ho-fung Leung
The Chinese University of Hong Kong
Hong Kong, China
E-mail: lhf@cse.cuhk.edu.hk

Carlos Lucena
PUC-Rio, Rio de Janeiro, Brazil
E-mail: lucena@inf.puc-rio.br

Alexander Romanovsky
University of Newcastle
Newcatle upon Tyne, UK
E-mail: Alexander.Romanovsky@newcastle.ac.uk

Library of Congress Control Number: 2007931241

CR Subject Classification (1998): D.2, I.2.11, C.2.4, D.1.3, H.3.5

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-73130-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-73130-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12078462 06/3180 5 4 3 2 1 0

Preface

Software is present in every aspect of our lives, pushing us inevitably towards a world of distributed computing systems. Agent concepts hold great promise for responding to the new realities of large-scale distributed systems. Multi-agent systems (MASs) and their underlying theories provide a more natural support for ensuring important agent properties, such as autonomy, environment heterogeneity, organization and openness. Nevertheless, a software agent is an inherently more complex abstraction, posing new challenges to software engineering. Without adequate development techniques and methods, MASs will not be sufficiently dependable, thus making their wide adoption by the industry more difficult.

The dependability of a computing system is its ability to deliver a service that can be justifiably trusted. It is a singular time for dependable distributed systems, since the traditional models we use to express the relationships between a computational process and its environment are changing from the standard deterministic types into ones that are more distributed and dynamic. This served as a guiding principle for planning the Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2006) workshop, starting with selecting the theme, "building dependable multi-agent systems." It acknowledges our belief in the increasingly vital role dependability plays as an essential element of MAS development.

SELMAS 2006 was the fifth edition of the workshop, organized in association with the 28th International Conference on Software Engineering (ICSE), held in Shanghai, China, in May 2006. After each workshop edition, it was decided to extend its scope, and to invite several of the workshop participants to write chapters for books based on their original position papers, as well as other leading researchers in the area to prepare additional chapters. Thus, this volume is the fifth in the *Software Engineering for Multi-Agent Systems LNCS* series.

In planning this volume, we sought to achieve both continuity and innovation. The papers selected for this volume present advances in software engineering approaches to develop dependable high-quality MASs. In addition, the power of agent-based software engineering is illustrated using actual real-world applications. These papers describe experiences and techniques associated with large MASs in a wide variety of problem domains.

This book brings together a collection of 12 papers addressing a wide range of issues in software engineering for MASs, reflecting the importance of agent properties in today's software systems. The papers in this book describe recent developments in specific issues and practical experience. At the end of each chapter, the reader will find a list of interesting references for further reading. The papers are grouped into five categories: Faulty Tolerance, Exception Handling and Diagnosis, Security and Trust, Verification and Validation, and Early Development Phases and Software Reuse. We believe that this carefully prepared volume will be of particular value to all readers interested in these key topics, describing the most recent developments in the field of software engineering for MASs.

The main target readers for this book are researchers and practitioners who want to keep up with the progress of software engineering in MASs, individuals keen to understand the interplay between agents and objects in software development, and those interested in experimental results from MAS applications. Software engineers involved with particular aspects of MASs as part of their work may find it interesting to learn about using software engineering approaches in building real systems. A number of chapters in the book discuss the development of MASs from requirements and architecture specifications to implementation.

We are confident that this book will be of considerable use to the software engineering community by providing many original and distinct views on such an important interdisciplinary topic, and by contributing to a better understanding and cross-fertilization among individuals in this research area.

Our thanks go to all our authors, whose work made this volume possible. Many of them also helped during the reviewing process. We would also like to express our gratitude to the members of the Evaluation Committee who were generous with their time and effort when reviewing the submitted papers. In conclusion, we extend once more our words of gratitude to all who contributed to making the SELMAS workshop series a reality. We hope that all of us will feel that we contributed in some way to helping improve the research on and the practice of software engineering for MASs in our society.

February 2007

Ricardo Choren
Alessandro Garcia
Holger Giese
Ho-fung Leung
Carlos Lucena
Alexander Romanovsky

Foreword

Although agent-based systems originated in the artificial intelligence community, they have become, over the past decade, an important topic for software engineering research. The reason for this is quite simple; the agent paradigm is extremely useful, if not essential, for solving many problems in software construction in the modern world of highly distributed, service-oriented, telecommunications and Internet-based systems. In many senses, there is nothing all that new about agents. After all, learning, goal-based behavior, planning and so on have been the subject of study for decades. Basic definitions of agents always include the concept of autonomy, defined (usually by example) as the ability to decide whether to accept a communication or not. But this ability is inherent in all software. Just look at operating systems or any reactive system! The idea of an open system also predates agents, e.g., actor systems, the Internet, etc. What is different about agents is the novel combination of these ingredients 'in one package' and the degree to which characteristics such as autonomy and being open are driving forces in the construction of these systems.

In this sense, it is quite natural to ask questions about agent system construction from the point of view of software engineering. As with any piece of software, we would expect that the software will be properly engineered, and not developed according to the paradigm described by the old joke about AI: How does an AI programmer develop software? He begins with the empty program and debugs until it works!

We know a lot about properly engineering software systems, even if this knowledge is not always deployed, but is there anything new that needs to be added in order to adapt the existing techniques and tools to support the construction of agent systems? As an example, Jean-Pierre Briot notes in the foreword of the 2004 SELMAS volume that the FIPA Agent Communication Language standard is an extension of the middleware idea inherent in CORBA to support the kind of communication requirements of business systems. There has been much discussion in the literature of platforms for agent-based systems, such as JADE, Grasshopper, JACK, Zeus, etc. In my view, these are middleware proposals, analogous to CORBA. Of course, they are more structured than CORBA and support more multidimensional interaction. But they are still middleware concepts. (Unfortunately, they are often referred to as 'architectures', leading to confusion when discussing *software architectures*. See below.) The past volumes of SELMAS and the current volume address issues in software engineering that investigate how standard ideas in software engineering apply to the construction of agent-based systems and the extent to which they have to be adapted.

What seems remarkable to me is the robustness of existing software engineering techniques with respect to this change in domain of application. One only has to peruse the section titles in the present volumes to see this historical resonance: 'Fault Tolerance', 'Exception Handling and Diagnosis', 'Security and Trust', 'Verification and Validation', 'Early Development Phases and Software Reuse'. Of course, the papers might thus be uninteresting to the wider community if the change in domain, to multi-agent systems, did not require substantial work in adapting and extending these

techniques. This is certainly what made the papers in the volume of great interest to me.

In my own area of interest, software architecture, I noted above a confusion that has crept into the agent literature. There is much discussion about architecture, but it seems to relate to the internal architecture of the middleware component of relevant platforms. There is very little discussion of software architecture, per se, in relation to the application itself, other than at the gross level of components. One of my own students is doing 'archaeology' on multi-agent system designs in the literature to determine what the software architecture of these designs might be. Initial investigation would seem to indicate that most such applications have an implicit software architecture and it is a standard one from the software architecture literature. Layered architectures and blackboard architectures are common. Against our expectations, it is hard to spot any new software architectures emerging from the agent world. This is extremely surprising and might be a fruitful topic for further investigation and discussion at a future instance of SELMAS!

Tom Maibaum
McMaster University

Organization

Evaluation Committee

Natasha Alechina
Mercedes Amor
Carole Bernon
Rafael Bordini
Jean-Pierre Bnot
Giacomo Cab~i
Grui a Catalin-Roman
Mehdi Dastani
Mark Greaves
Zahia Guessoum
Giancarlo Guizzardi
Alexei Iliasov
Christine Julien
Rogerio de Lemos
Michael Luck
Viviana Mascardi
Haralabos Mouratidis
Andrea Omicini
Juan Pav6n
Gustavo Rossi
John Shepherdson
Viviane Silva
Danny Wcyns

Additional Reviewers

Juan Botia
Davide Grossi
Yuanfang Li

Lecture Notes in Computer Science 4408

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Lecture Notes in Computer Science

For information about Vols. 1–4525

please contact your bookseller or Springer

Vol. 4613: F.P. Preparata, Q. Fang (Eds.), Frontiers in Algorithmics. XI, 348 pages. 2007.

Vol. 4612: I. Miguel, W. Ruml (Eds.), Abstraction, Reformulation, and Approximation. XI, 418 pages. 2007. (Sublibrary LNAI).

Vol. 4611: J. Indulska, J. Ma, L.T. Yang, T. Ungerer, J. Cao (Eds.), Ubiquitous Intelligence and Computing. XXIII, 1257 pages. 2007.

Vol. 4610: B. Xiao, L.T. Yang, J. Ma, C. Muller-Schloer, Y. Hua (Eds.), Autonomic and Trusted Computing. XVIII, 571 pages. 2007.

Vol. 4609: E. Ernst (Ed.), ECOOP 2007 — Object-Oriented Programming. XIII, 625 pages. 2007.

Vol. 4608: H.W. Schmidt, I. Crnkovic, G.T. Heineman, J.A. Stafford (Eds.), Component-Based Software Engineering. XII, 283 pages. 2007.

Vol. 4607: L. Baresi, P. Fraternali, G.-J. Houben (Eds.), Web Engineering. XVI, 576 pages. 2007.

Vol. 4606: A. Pras, M. van Sinderen (Eds.), Dependable and Adaptable Networks and Services. XIV, 149 pages. 2007.

Vol. 4605: D. Papadias, D. Zhang, G. Kollios (Eds.), Advances in Spatial and Temporal Databases. X, 479 pages. 2007.

Vol. 4604: U. Priss, S. Polovina, R. Hill (Eds.), Conceptual Structures: Knowledge Architectures for Smart Applications. XII, 514 pages. 2007. (Sublibrary LNAI).

Vol. 4603: F. Pfenning (Ed.), Automated Deduction – CADE-21. XII, 522 pages. 2007. (Sublibrary LNAI).

Vol. 4602: S. Barker, G.-J. Ahn (Eds.), Data and Applications Security XXI. X, 291 pages. 2007.

Vol. 4600: H. Comon-Lundh, C. Kirchner, H. Kirchner, Rewriting, Computation and Proof. XVI, 273 pages. 2007.

Vol. 4599: S. Vassiliadis, M. Berekovic, T.D. Hämäläinen (Eds.), Embedded Computer Systems: Architectures, Modeling, and Simulation. XVIII, 466 pages. 2007.

Vol. 4598: G. Lin (Ed.), Computing and Combinatorics. XII, 570 pages. 2007.

Vol. 4597: P. Perner (Ed.), Advances in Data Mining. XI, 353 pages. 2007. (Sublibrary LNAI).

Vol. 4596: L. Arge, C. Cachin, T. Jurdziński, A. Tarlecki (Eds.), Automata, Languages and Programming. XVII, 953 pages. 2007.

Vol. 4595: D. Bošnački, S. Edelkamp (Eds.), Model Checking Software. X, 285 pages. 2007.

Vol. 4594: R. Bellazzi, A. Abu-Hanna, J. Hunter (Eds.), Artificial Intelligence in Medicine. XVI, 509 pages. 2007. (Sublibrary LNAI).

Vol. 4592: Z. Kedad, N. Lammari, E. Métais, F. Meziane, Y. Rezgui (Eds.), Natural Language Processing and Information Systems. XIV, 442 pages. 2007.

Vol. 4591: J. Davies, J. Gibbons (Eds.), Integrated Formal Methods. IX, 660 pages. 2007.

Vol. 4590: W. Damm, H. Hermanns (Eds.), Computer Aided Verification. XV, 562 pages. 2007.

Vol. 4589: J. Münch, P. Abrahamsson (Eds.), Product-Focused Software Process Improvement. XII, 414 pages. 2007.

Vol. 4588: T. Harju, J. Karhumäki, A. Lepistö (Eds.), Developments in Language Theory. XI, 423 pages. 2007.

Vol. 4587: R. Cooper, J. Kennedy (Eds.), Data Management. XIII, 259 pages. 2007.

Vol. 4586: J. Pieprzyk, H. Ghodosi, E. Dawson (Eds.), Information Security and Privacy. XIV, 476 pages. 2007.

Vol. 4585: M. Kryszkiewicz, J.F. Peters, H. Rybinski, A. Skowron (Eds.), Rough Sets and Intelligent Systems Paradigms. XIX, 836 pages. 2007. (Sublibrary LNAI).

Vol. 4584: N. Karssemeijer, B. Lelieveldt (Eds.), Information Processing in Medical Imaging. XX, 777 pages. 2007.

Vol. 4583: S.R. Della Rocca (Ed.), Typed Lambda Calculi and Applications. X, 397 pages. 2007.

Vol. 4582: J. Lopez, P. Samarati, J.L. Ferrer (Eds.), Public Key Infrastructure. XI, 375 pages. 2007.

Vol. 4581: A. Petrenko, M. Veanes, J. Tretmans, W. Grieskamp (Eds.), Testing of Software and Communicating Systems. XII, 379 pages. 2007.

Vol. 4580: B. Ma, K. Zhang (Eds.), Combinatorial Pattern Matching. XII, 366 pages. 2007.

Vol. 4579: B. M. Hämmerli, R. Sommer (Eds.), Detection of Intrusions and Malware, and Vulnerability Assessment. X, 251 pages. 2007.

Vol. 4578: F. Masulli, S. Mitra, G. Pasi (Eds.), Applications of Fuzzy Sets Theory. XVIII, 693 pages. 2007. (Sublibrary LNAI).

Vol. 4577: N. Sebe, Y. Liu, Y.-t. Zhuang (Eds.), Multimedia Content Analysis and Mining. XIII, 513 pages. 2007.

Vol. 4576: D. Leivant, R. de Queiroz (Eds.), Logic, Language, Information and Computation. X, 363 pages. 2007.

Vol. 4575: T. Takagi, T. Okamoto, E. Okamoto, T. Okamoto (Eds.), Pairing-Based Cryptography – Pairing 2007. XI, 408 pages. 2007.

Vol. 4574: J. Derrick, J. Vain (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2007. XI, 375 pages. 2007.

Vol. 4573: M. Kauers, M. Kerber, R. Miner, W. Windsteiger (Eds.), Towards Mechanized Mathematical Assistants. XIII, 407 pages. 2007. (Sublibrary LNAI).

Vol. 4572: F. Stajano, C. Meadows, S. Capkun, T. Moore (Eds.), Security and Privacy in Ad-hoc and Sensor Networks. X, 247 pages. 2007.

Vol. 4571: P. Perner (Ed.), Machine Learning and Data Mining in Pattern Recognition. XIV, 913 pages. 2007. (Sublibrary LNAI).

Vol. 4570: H.G. Okuno, M. Ali (Eds.), New Trends in Applied Artificial Intelligence. XXI, 1194 pages. 2007. (Sublibrary LNAI).

Vol. 4569: A. Butz, B. Fisher, A. Krüger, P. Olivier, S. Owada (Eds.), Smart Graphics. IX, 237 pages. 2007.

Vol. 4566: M.J. Dainoff (Ed.), Ergonomics and Health Aspects of Work with Computers. XVIII, 390 pages. 2007.

Vol. 4565: D.D. Schmorrow, L.M. Reeves (Eds.), Foundations of Augmented Cognition. XIX, 450 pages. 2007. (Sublibrary LNAI).

Vol. 4564: D. Schuler (Ed.), Online Communities and Social Computing. XVII, 520 pages. 2007.

Vol. 4563: R. Shumaker (Ed.), Virtual Reality. XXII, 762 pages. 2007.

Vol. 4562: D. Harris (Ed.), Engineering Psychology and Cognitive Ergonomics. XXIII, 879 pages. 2007. (Sublibrary LNAI).

Vol. 4561: V.G. Duffy (Ed.), Digital Human Modeling. XXIII, 1068 pages. 2007.

Vol. 4560: N. Aykin (Ed.), Usability and Internationalization, Part II. XVIII, 576 pages. 2007.

Vol. 4559: N. Aykin (Ed.), Usability and Internationalization, Part I. XVIII, 661 pages. 2007.

Vol. 4558: M.J. Smith, G. Salvendy (Eds.), Human Interface and the Management of Information, Part II. XXIII, 1162 pages. 2007.

Vol. 4557: M.J. Smith, G. Salvendy (Eds.), Human Interface and the Management of Information, Part I. XXII, 1030 pages. 2007.

Vol. 4556: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part III. XXII, 1020 pages. 2007.

Vol. 4555: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part II. XXII, 1066 pages. 2007.

Vol. 4554: C. Stephanidis (Ed.), Universal Acess in Human Computer Interaction, Part I. XXII, 1054 pages. 2007.

Vol. 4553: J.A. Jacko (Ed.), Human-Computer Interaction, Part IV. XXIV, 1225 pages. 2007.

Vol. 4552: J.A. Jacko (Ed.), Human-Computer Interaction, Part III. XXI, 1038 pages. 2007.

Vol. 4551: J.A. Jacko (Ed.), Human-Computer Interaction, Part II. XXIII, 1253 pages. 2007.

Vol. 4550: J.A. Jacko (Ed.), Human-Computer Interaction, Part I. XXIII, 1240 pages. 2007.

Vol. 4549: J. Aspnes, C. Scheideler, A. Arora, S. Madden (Eds.), Distributed Computing in Sensor Systems. XIII, 417 pages. 2007.

Vol. 4548: N. Olivetti (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods. X, 245 pages. 2007. (Sublibrary LNAI).

Vol. 4547: C. Carlet, B. Sunar (Eds.), Arithmetic of Finite Fields. XI, 355 pages. 2007.

Vol. 4546: J. Kleijn, A. Yakovlev (Eds.), Petri Nets and Other Models of Concurrency – ICATPN 2007. XI, 515 pages. 2007.

Vol. 4545: H. Anai, K. Horimoto, T. Kutsia (Eds.), Algebraic Biology. XIII, 379 pages. 2007.

Vol. 4544: S. Cohen-Boulakia, V. Tannen (Eds.), Data Integration in the Life Sciences. XI, 282 pages. 2007. (Sublibrary LNBI).

Vol. 4543: A.K. Bandara, M. Burgess (Eds.), Inter-Domain Management. XII, 237 pages. 2007.

Vol. 4542: P. Sawyer, B. Paech, P. Heymans (Eds.), Requirements Engineering: Foundation for Software Quality. IX, 384 pages. 2007.

Vol. 4541: T. Okadome, T. Yamazaki, M. Makhtari (Eds.), Pervasive Computing for Quality of Life Enhancement. IX, 248 pages. 2007.

Vol. 4539: N.H. Bshouty, C. Gentile (Eds.), Learning Theory. XII, 634 pages. 2007. (Sublibrary LNAI).

Vol. 4538: F. Escolano, M. Vento (Eds.), Graph-Based Representations in Pattern Recognition. XII, 416 pages. 2007.

Vol. 4537: K.C.-C. Chang, W. Wang, L. Chen, C.A. Ellis, C.-H. Hsu, A.C. Tsoi, H. Wang (Eds.), Advances in Web and Network Technologies, and Information Management. XXIII, 707 pages. 2007.

Vol. 4536: G. Concas, E. Damiani, M. Scotto, G. Succi (Eds.), Agile Processes in Software Engineering and Extreme Programming. XV, 276 pages. 2007.

Vol. 4534: I. Tomkos, F. Neri, J. Solé Pareta, X. Masip Bruin, S. Sánchez Lopez (Eds.), Optical Network Design and Modeling. XI, 460 pages. 2007.

Vol. 4533: F. Baader (Ed.), Term Rewriting and Applications. XII, 419 pages. 2007.

Vol. 4531: J. Indulska, K. Raymond (Eds.), Distributed Applications and Interoperable Systems. XI, 337 pages. 2007.

Vol. 4530: D.H. Akehurst, R. Vogel, R.F. Paige (Eds.), Model Driven Architecture- Foundations and Applications. X, 219 pages. 2007.

Vol. 4529: P. Melin, O. Castillo, L.T. Aguilar, J. Kacprzyk, W. Pedrycz (Eds.), Foundations of Fuzzy Logic and Soft Computing. XIX, 830 pages. 2007. (Sublibrary LNAI).

Vol. 4528: J. Mira, J.R. Álvarez (Eds.), Nature Inspired Problem-Solving Methods in Knowledge Engineering, Part II. XXII, 650 pages. 2007.

Vol. 4527: J. Mira, J.R. Álvarez (Eds.), Bio-inspired Modeling of Cognitive Tasks, Part I. XXII, 630 pages. 2007.

Vol. 4526: M. Malek, M. Reitenspieß, A. van Moorsel (Eds.), Service Availability. X, 155 pages. 2007.

Table of Contents

On Fault Tolerance in Law-Governed Multi-agent Systems

Maíra A. de C. Gatti[1], Gustavo R. de Carvalho[1], Rodrigo B. de Paes[1], Carlos J.P. de Lucena[1], and Jean-Pierre Briot[2]

[1] Software Engineering Laboratory, PUC-Rio, Rio de Janeiro, Brazil
{mgatti, guga, rbp, lucena}@les.inf.puc-rio.br
[2] Laboratoire d'informatique de Paris 6 (LIP6), Universit´e Pierre et Marie Curie, Paris, France
Jean-Pierre.Briot@lip6.fr

Abstract. The dependability of open multi-agent systems is a particular concern, notably because of their main characteristics as decentralization and no single point of control. This paper describes an approach to increase the availability of such systems through a technique of fault tolerance known as agent replication, and to increase their reliability through a mechanism of agent interaction regulation called law enforcement mechanism. Therefore, we combine two frameworks: one for law enforcement, named XMLaw, and another for agent adaptive replication, named DimaX, in which the decision of replicating an agent is based on a dynamic estimation of its criticality. Moreover, we will describe how we can reuse some of the information expressed by laws in order to help at the estimation of agent criticality, thus providing a better integration of the two frameworks. At the end of the paper, we recommend a means to specify criticality monitoring variation through a structured argumentation approach that documents the rationale around the decisions of the law elements derivation.

Keywords: Multi-Agent Systems; Open Systems; Law-Governed Approach; Dependability of Open Systems; Fault Tolerance; Requirements; Criticality; Availability, Reliability.

1 Introduction

There are many definitions in the literature for agents and, consequently, multi-agent systems. And despite their differences, all of them basically characterize a multi-agent system (MAS) as a computational environment in which individual software agents interact with each other, in a cooperative manner, or in a competitive manner, and sometimes autonomously pursuing their individual goals. During this process, they access the environment's resources and services and occasionally produce results for the entities that initiated these software agents [1]. As the agents interact in a concurrent, asynchronous and decentralized manner, this kind of system can be categorized as a complex system [2].

R. Choren et al. (Eds.): SELMAS 2006, LNCS 4408, pp. 1 – 20, 2007.

The absence of centralized coordination data makes it hard to determine the current state of the system and/or to predict the effects of actions. Moreover, all of the possible situations that may arise in the execution context led us to be uncertain about predicting the behavior of agents. However, in critical applications such as business environments or government agencies, the behavior of the global system must be taken into account and structural characteristics of the domain have to be incorporated [10].

A particular issue that arises from this kind of software is: how we can ensure their dependability (which is the ability of a computer system to deliver service that can justifiably be trusted [3]) considering the reliability of critical applications and availability of these agents. There are some proposals to address such problem ([3][4][5][6], for instance, for fault tolerance and [7][8] for reliability) which have been proposed in the last few years using different approaches; each one solved a restricted problem involving dependability.

In this paper we propose an approach to increase the availability of multi-agent systems through a technique of fault tolerance known as agent replication, and to increase its reliability through a mechanism of agent interaction regulation called law enforcement mechanism. Therefore, we will combine two frameworks. The first framework, named XMLaw, manages law enforcement to increase reliability and correctness. The second framework, named DimaX, manages adaptive replication of agents in order to increase fault-tolerance. In DimaX, the decision of replicating an agent is based on a dynamic estimation of its criticality given by a criticality monitoring strategy. The agent criticality defines how important the agent is to the organization and consequently to the system. The estimation of the criticality of an agent can be based on different information, as the messages it sends or receives, or the role it plays, etc. In this paper, we will describe how we can reuse some of the information expressed by laws, and supported by XMLaw, in order to further contribute to the estimation of agent criticality, thus providing a better integration of the two frameworks. The novelty of this contribution is in the proposed combination of law-based governance and replication-based fault-tolerance, rather than in specific contributions in law-based governance or in fault-tolerance.

We also propose a means to specify the criticality monitoring strategy through a structured argumentation [24] that documents the rationale around the decisions of the law elements derivation. However, it will not be detailed in this paper. Moreover, we also provide a framework that implements the criticality monitoring variation behavior specified.

The subsequent sections are organized as follows: Section 2 presents an introduction to the agent replication-based fault tolerance for multi-agent systems, and Section 3 presents the law enforcement approach for increasing the reliability of these systems. Section 4 states a scenario for the problem description. Section 5 details the proposed solution for the problem as an integrated architecture. This architecture is the integration of both approaches presented in Section 2 and 3. Section 6 presents one of the case studies implemented to validate the concepts and the architecture. And finally, Section 7 concludes this paper and presents future works.

2 Fault Tolerance in Multi-agent Systems: Agent Replication

The multi-agent systems deployed in an open environment, where agents from various organizations interact in the same MAS, are distributed over many hosts and communicate over public networks, hence more attention must be paid to fault tolerance.

Several approaches ([4][14][15]) address the multi-faced problem of fault tolerance in multi-agent systems. Some of them handle the problems of communication, interaction and coordination of agents with the other agents of the system. Others address the difficulties of making reliable mobile agents, which are more exposed to security problems. Some of them are based on replication mechanisms [9], and as mentioned before they have solved many problems of ubiquitous systems.

Agent replication is the act of creating one or more replicas of one or more agents, and the number of each agent replica is the replication degree; everything depends on how critical the agent is while executing its tasks. Among the significant advantages over other fault-tolerance solutions, first and foremost, agent replication provides the groundwork for the shortest recovery delays. Also, generally it is less intrusive with respect to execution time. And finally, it scales much better [9]. There is a framework, named DimaX [6], that allows dynamic replication and dynamic adaptation of the replication policy (e.g., passive to active, changing the number of replicas). It was designed to easily integrate various agent architectures, and the mechanisms that ensure dependability are kept as transparent as possible to the application. Basically, DimaX is the integration between a multi-agent system called Dima and the dynamic replication architecture for agents called DarX.

There are two cases that might be distinguished: 1) the agent's criticality is static and 2) the agent's criticality is dynamic. In the first case, multi-agent systems have often static organization structures, static behaviors of agents, and a small number of agents. Critical agents, therefore, can be identified by the designer and can be replicated by the programmer before run time.

In the second case, the agent criticality cannot be determined before run time due to the fact that the multi-agent systems may have dynamic organization structures, dynamic behaviors of agents and a large number of agents. Then it is important to determine these structures dynamically in order to evaluate agent criticality. The approach detailed in [16] proposes a way of determining it through role analysis. It could be done by some prior input from the designer of the application who specifies the roles' weights, or there would be an observation module for each server that collects the data through the agent execution and their interactions. In the second approach, global information is built and then used to obtain roles and degree of activity to compute the agent criticality.

Another way of dynamically determining these structures to evaluate agent criticality is to represent the emergent organizational structure of a multi-agent system by a graph [6]. The hypothesis is that the criticality of an agent relies on the interdependences of other agents on this agent. First, the interdependence graph is initialized by the designer, and then it is dynamically adapted by the system itself. Some algorithms to dynamically adapt and describe it are proposed in [6].

We will present here an enhancement of these approaches and it will be further described in Section 5. Basically, we improved the agent criticality calculation through dynamic elements present during interactions with other agents. These

elements will be described in the next section while the law enforcement approaches, especially the one that was chosen, are exposed.

3 Law-Governed Interaction

In open multi-agent systems the development takes place without a centralized control, thus it is necessary to ensure the reliability of these systems in a way that all the interactions between agents will occur according to the specification and that these agents will obey the specified scenario. For this, these applications must be built upon a law-governed architecture.

In this kind of architecture, enforcement that is responsible for the interception of messages and the interpreting of previously described laws is implemented. The core of a law-governed approach is the mechanism used by the mediator to monitor the conversations between agents.

Note that law-governed approaches have some relations with general coordination mechanisms (e.g., tuple-space mechanisms like Tucson [26]) in that they specify and control interactions between agents. However, the specificity of law-governed mechanisms is about controlling interactions and actions from a social (social norms) perspective, whereas general coordination languages and mechanisms focus on means for expressing synchronization and coordination of activities and exchange of information, at a lower (not social) computational level.

Among the models and frameworks that were developed to support law-governed mechanism (for instance, [7][8][17][18]), XMLaw [7] was chosen for three main reasons. First, because it implements a law enforcement approach as an object-oriented framework, which brings the benefits of reuse and flexibility. Second, it allows normative behavior that is more expressive than the others through the connection between norms and clocks. And finally, it permits the execution of Java code through the concept of actions. Thus, in this section, we explain the XMLaw description language [7] and the M-Law framework [19].

M-Law works by intercepting messages exchanged between agents, verifying the compliance of the messages with the laws and subsequently redirecting the message to the real addressee, if the laws allow it (Figure 1). If the message is not compliant, then the mediator blocks the message and applies the consequences specified in the law.

This infrastructure, whenever necessary, can be extended to fulfill open system requirements or interoperability concerns. M-Law architecture is based on a pool of mediators that intercept messages and interpret the previously described laws.

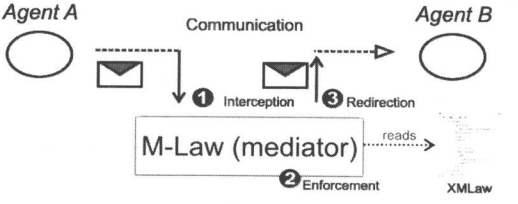


Fig. 1. M-Law Architecture