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

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

Table of Contents