Lecture Notes in Computer Science 5386

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

Michael Luck Jorge J. Gomez-Sanz (Eds.)

Agent-Oriented Software Engineering IX

9th International Workshop, AOSE 2008
Estoril, Portugal, May 12-13, 2008
Revised Selected Papers

Springer

Volume Editors

Michael Luck
King's College London
Department of Computer Science
Strand, London WC2R 2LS, UK
E-mail: michael.luck@kcl.ac.uk

Jorge J. Gomez-Sanz
Universidad Complutense de Madrid
Facultad de Informatica
Avda. Complutense s/n, 28040 Madrid, Spain
E-mail: jjgomez@sip.ucm.es

Library of Congress Control Number: Applied for

CR Subject Classification (1998): D.2, I.2.11, F.3, D.1, C.2.4, D.3

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-642-01337-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-01337-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12649822 06/3180 5 4 3 2 1 0

Preface

Agent technology can be the key to a brand new family of applications providing outstanding features like autonomy and adaptation. However, the way in which such applications should be built is not yet clear to us. Just as with other kinds of software, we need expertise in the inherent problems, and we need to develop a substantial body of knowledge to enable others to take our efforts further. In the context of multi-agent systems, this knowledge is realized in the form of agent-oriented software engineering (AOSE). Thus, AOSE brings novel tools and methods with which developers can start to construct agent-oriented solutions for their problems.

AOSE has brought some important advances in agent research. These advances have been brought about because we tried to solve problems with what was available, and found that it was not sufficient. It is natural to conclude, therefore, that the more difficult the problems we consider, the more advances we will achieve.

Such advances necessarily mean research, and this is also consistent with the *engineering* spirit. AOSE provides a context in which formal and applied research meet, and it will remain so for a long time, and at the very least until AOSE matures to match object-oriented software engineering. For this to happen, again, increasing the complexity of our problems and showing the benefits of agent technology is required.

This complexity can come in different flavors. One such aspect is the problem size: can agent technology, as it is now, deal with development with a team of, say, eight people in a year? If verifiable evidence of this can be shown, it would provide a significant and welcome impetus to the area. With respect to the evaluation of the benefits to be gained, we have hypothesized for some time now that agent technology ought to find a natural niche in global computing, network enterprises, ubiquitous computing and sensor networks, to mention just a few examples. Undoubtedly, there is already functioning software in such domains, so the question arises as to why we should use agents instead. To address this concern, we need to demonstrate that an agent solution is better in at least one of the following aspects: it is more economical in cost, it is more robust and fails less often, it can be developed in less time, or it provides better performance.

The AOSE workshops aim to address these issues, both from a research perspective and from a perspective that is relevant to attracting the attention of developers. The issues all share a clear connection to the main problem: how to effectively and efficiently develop software systems using agent technologies. The AOSE workshops thus seek to contribute to the advance of AOSE by providing a forum for presentation, discussion and debate of exactly these concerns.

Building on the success of the eight previous workshops, the 9th International Workshop on Agent-Oriented Software Engineering (AOSE 2008) took place

in Lisbon in May 2008 as part of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008).

The 2008 workshop received 50 submitted papers, a very strong number, which reflects the continuing interest in the area. All papers were reviewed by at least three reviewers from an international Program Committee of 42 members and 17 auxiliary reviewers, and presented papers were then subject to a second round of reviews for inclusion in this volume.

In structuring this volume, we have organized the papers into four sections. The first deals with multi-agent organizations, which provides a valuable abstraction for agents and needs appropriate integration into development methods. The second section addresses method engineering and software development processes, including papers that focus on the ways in which multi-agent systems are developed, the activities involved, the products generated, and the assessment of the suitability of methods for particular domain problems. The third section is dedicated to testing and debugging activities. Finally, the last section deals with tools and case studies.

1 Multi-agent Organizations

Starting with organizations, the first section begins with a paper by Coutinho et al., in which they propose an integration process for *organizational models* based on concepts from model-driven engineering (MDE). The process is applied to five organizational meta-models (AGR, $\mathcal{M}$OISE+,TÆMS, ISLANDER and OperA) to obtain an integrated meta-model, which can be used as an interlingua for multi-agent systems built according to any of the five organizational meta-models.

Continuing this idea, the second paper, by Argente et al., introduces a meta-model for organizations as an evolution of the INGENIAS meta-model. This new meta-model includes primitives to express norms, services, and a holonic approach for the definition of organizations.

Finally, the paper by Sudeikat et al. closes the section. It is concerned with the validation of the dynamics within an organization, and proposes a guide to validation based on simulations of the MAS under development. The paper illustrates the guide through validation of an intrusion-detection system.

2 Method Engineering and Software Development Processes

The second section begins with a paper by Seidita et al., aimed at grounding situational method engineering to show how to express an agent-oriented software engineering process using a software process engineering meta-model (SPEM). Situational method engineering promotes the idea that no single method can account for all methods needed by engineers, especially due to changes in the application domain. As an example, the paper illustrates the construction procedure with the formalization of the PASSI methodology, exploring which SPEM primitives are best suited for representing each part.

Also in the line of method engineering, García-Magariño et al. introduce the different issues arising during the specification of a development process for the INGENIAS methodology. First, they identify problems with existing process modeling tools, concretely EPF, APES, and Metameth. Then, they present a new tool, built with the Eclipse Modeling Framework, overcoming most of the difficulties, and providing an alternative implementation of SPEM. The validation of the tool is achieved using as a case study of some preliminary work undertaken on the formalization of the INGENIAS methodology development process. This leads to the identification of steps and artifacts that may benefit the formalization of other methodologies.

The next paper, by Rougemaille et al., starts with an introduction to SPEM and situational method engineering, before introducing the fragment concept using the domain of agent-oriented software engineering. Fragment implementation is achieved by means of SPEM, and requires the combination of several SPEM primitives. As a proof of concept, the paper provides some fragments identified in the ADELFE and PASSI methodologies.

Cossentino et al. introduce a procedure to build an ad hoc agent-oriented software engineering process using fragments from a library of methods extracted from different agent-oriented methodologies. These libraries account for pieces of MAS meta-models as well as the procedures used to instantiate these pieces. By using the libraries and the knowledge of the domain problem, a developer can then follow the procedure described in the paper to create an ad hoc methodology. The paper presents as proof of concept the ASPECS process, which is constructed out of fragments from PASSI, CRIO and Janus.

Following this, García-Magariño et al. present a set of metrics that serves to quantitatively evaluate the *availability* (whether a meta-model has all required elements to deal with a problem domain), *specificity* (accounting for concepts that are not used), and *expressiveness* (covering the number of elements that are needed to represent a system specification). These metrics are applied to six different meta-models from six different agent-oriented methodologies: Tropos, PASSI, Agile_PASSI, Prometheus, MaSE, and INGENIAS.

In a different vein, the work by Padgham et al. proposes a new notation that may facilitate communication among different agent-oriented software engineers. Although meta-models of different agent-oriented methodologies remain unchanged, the use of a common representation of concepts can make these models appear to be more similar. Here, the semantics are still different so that an agent in PASSI is still different from an agent in MaSE. The effectiveness of the solution is still to be evaluated, but it provides an alternative to the unification of existing meta-models.

Continuing the methodology integration topic, Gascueña et al. show that it is not necessary to build a new methodology from fragments or to unify meta-models to gain the benefit of the different methodologies: their paper describes how to combine INGENIAS and Prometheus while keeping the benefits of both. The idea is to use each methodology only at concrete points in the development process, when a transition from Prometheus to INGENIAS is required. Since

this transition suggests a manual translation of Prometheus concepts into INGENIAS ones, the authors identify such a mapping, and describe how it can be accomplished by using concrete examples.

By contrast, Hahn et al. criticize the meta-model approach, due to limitations in expressing semantics, and instead propose specifying semantics with Object-Z, a state-based and object-oriented specification language. Their paper introduces the basics of Object-Z and explains how it can be used to add semantics to a concrete meta-model called DSML4MAS, claiming that this can also be applied to other existing meta-models. In particular, they argue that the Object-Z semantics can be partially translated in terms of the Object Constraint Language (OCL), a language traditionally used together with meta-models.

The section concludes with a paper from Dam et al., dealing with two important problems in a software system: its maintenance and evolution. The approach is based on a library of repair plans to fix inconsistencies in the design model, a meta-model of the problem, and consistency constraints. When a change is requested, the system analyzes the current specification and then chooses a repair plan to modify the specification. The procedure is implemented with the Change Propagation Assistant (CPA), a prototype system.

3 Testing and Debugging

The section on testing and debugging, an important area that brings AOSE closer to the concerns of real-world commercial deployment, commences with a paper by Ekinci et al. The paper suggests performing testing activities by using goals as the driving criteria, and defines the concept of *test goal*. This concept represents the group of tests needed in order to check if a goal is achieved correctly. Here, each of the needed tests is considered to have its own goal to check, so that a *test goal* has three subgoals: setup (prepare the system); goal under test (perform actions related to the goal); and assertion goal (check goal satisfaction). These ideas are implemented in the SEAUnit test tool.

Looking for other kinds of driving criteria in testing, Nguyen et al. propose using the ontologies extracted from the MAS under test and a set of OCL constraints, which act as a test oracle. Having as input a representation of the ontologies used, the idea is to construct an agent able to deliver messages whose content is inspired by these ontologies. The resulting behaviors are regarded as correct using the input set of OCL constraints: if the message content satisfies the constraints, the message is correct. The procedure is supported by *eCAT*, a software tool.

The third paper in the section, by Gomez-Sanz et al., describes progress made in the INGENIAS methodology in these areas. With respect to testing, the INGENIAS meta-model is extended with concepts for defining tests, and the code generation facilities are augmented to produce JUnit skeletons based on these definitions. With respect to debugging, the system is integrated with ACLAnalyzer, a data-mining facility for capturing agent communications and exploring them with different graphical representations. The paper finishes with a survey and categorization of different work on testing and debugging in the agent literature.

4 Tools and Case Studies

In the last section of the book, we focus on tools and case studies, seeking to provide valuable experience reports and practical assistance. The first paper of this section, by Cabrera-Paniagua et al., is an application paper describing development experience with the PASSI methodology. The chosen case study concerns a system simulating a passenger transportation enterprise within which agents represent different transport companies trying to satisfy the needs of simulated users.

The second paper, by Nunes et al., is also an application paper with different versions of a conference management system. Its focus lies in the analysis and comparison of the evolution from a non-agent-oriented system to a product-line agent-oriented system. The paper ends with an interesting evaluation of the different variability types identified in the agent system, as well as a discussion of how refactoring of the system could have been done in other ways.

The paper by Yoo et al., introduces a tool that combines JADE and Repast in order to provide the construction of simulations. This combination seems well suited for working with value-adding networks, which are complex networks of partners arranged by an enterprise. The use of Repast enables the problem to be modeled quickly, while the simulation itself is achieved by using JADE.

Following on from this, van Putten et al. present another tool resulting from the combination of OperA and Brahms, the former being a modeling language for MAS, and the latter a development environment for MAS based on the concept of activity. A simulation of air traffic is the chosen case study where this combination is put to the test.

Finally, the section (and the book) ends with a paper by Gorodetsky et al., presenting a support tool that can be used with the Gaia methodology, although the paper also provides additional guidelines to deal with the design and implementation stages. The tool enables the generation of executable code and implements non-trivial parts of Gaia, such as liveness expressions, and the paper provides examples of the tool applied to an air traffic management case study.

November 2008

Jorge J. Gomez-Sanz
Michael Luck

Organization

Workshop Chairs

Michael Luck (Co-chair)
Department of Computer Science
King's College London
UK
Email: michael.luck@kcl.ac.uk

Jorge J. Gomez-Sanz (Co-chair)
Facultad de Informatica
Universidad Complutense de Madrid
Spain
Email: jjgomez@sip.ucm.es

Steering Committee

Paolo Ciancarini	University of Bologna, Italy
Michael Wooldridge	University of Liverpool, UK
Jörg Müller	Technische Universität Clausthal, Germany
Gerhard Weiss	Software Competence Center Hagenberg GmbH, Austria

Program Committee

Claudio Bartolini (USA)
Bernard Bauer (Germany)
Federico Bergenti (Italy)
Carole Bernon (France)
Olivier Boissier (France)
Paolo Cianciarini (Italy)
Massimo Cossentino (Italy)
Keith Decker (USA)
Scott DeLoach (USA)
Noura Faci (UK)
Klaus Fischer (Germany)
Rubén Fuentes (Spain)
Paolo Giorgini (Italy)
Marie-Pierre Gleizes (France)
Nathan Griffiths (UK)
Michael Huhns (USA)
Vicent Julian Inglada (Spain)
Joao Leite (Portugal)
Jürgen Lind (Germany)
Carlos Jose Pereira de Lucena (Brazil)
Viviana Mascardi (Italy)
Simon Miles (UK)
Sanjay Modgil (UK)
Haris Mouratidis (UK)
Eugenio Oliveira (Portugal)
Andrea Omicini (Italy)
Nir Oren (UK)
Juan Pavón (Spain)
Michal Pechoucek (Czech Republic)
Joaquín Peña (Spain)
Anna Perini (Italy)
Eric Platon (Japan)

Alessandro Ricci (Italy)
Fariba Sadri (UK)
Brian Henderson-Sellers (Australia)
Onn Shehory (Israel)
Viviane Torres da Silva (Spain)
Arnon Sturm (Israel)
Laszlo Varga (Hungary)
Michael Winikoff (Australia)
Eric Yu (Canada)

Auxiliary Reviewers

Estefania Argente
Petr Benda
António Castro
Adriana Giret
Christian Hahn
Sachin Kamboj
Cristián Madrigal-Mora
Frederic Migeon
Ambra Molesini
Mirko Morandini
Elena Nardini
Cu Nguyen Duy
Cristina Ribeiro
Luca Sabatucci
Valeria Seidita
John Thangarajah
Jiri Vokrinek

Table of Contents

Multi-agent Organizations

Method Engineering and Software Development Processes

Testing and Debugging

Tools and Case Studies