Lecture Notes in Computer Science 4405

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

Lin Padgham Franco Zambonelli (Eds.)

Agent-Oriented Software Engineering VII

7th International Workshop, AOSE 2006
Hakodate, Japan, May 8, 2006
Revised and Invited Papers

Springer

Volume Editors

Lin Padgham
RMIT University, Melbourne, Australia
E-mail: linpa@cs.rmit.edu.au

Franco Zambonelli
Università di Modena e Reggio Emilia, DISMI
Via Allegri 13, Reggio Emilia, Italia
E-mail: franco.zambonelli@unimore.it

Library of Congress Control Number: 2007920434

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 3-540-70944-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-70944-2 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12021468 06/3142 5 4 3 2 1 0

Preface

Since the mid 1980s, software agents and multi-agent systems have grown into a very active area of research with some very successful examples of commercial development. At AAMAS 2006 Steve Benfield from Agentis described research on large scale industry system development, which indicated a savings of four to five times in development time and in cost when using agent technologies. However it is still the case that one of the limiting factors in industry take up of agent technology is the lack of adequate software engineering support, and knowledge in how to systematically develop agent systems.

The concept of an agent as an autonomous system, capable of interacting with other agents in order to satisfy its design objectives, is a natural one for software designers. Just as we can understand many systems as being composed of essentially passive objects, which have state, and upon which we can perform operations, so we can understand many others as being made up of interacting, semi-autonomous agents. This paradigm is especially suited to complex systems. However software architectures that contain many dynamically interacting components, each with their own thread of control, and engaging in complex coordination protocols, are difficult to correctly and efficiently engineer. Agent oriented modelling techniques are important for supporting the design and development of such applications.

The AOSE 2006 workshop was hosted by the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006) held in Hakodate, Japan. A selection of extended versions of papers from that workshop, along with some additional papers, are presented in this volume, which follows the successful predecessors of 2000 to 2005, published as *Lecture Notes in Computer Science*, volumes 1957, 2222, 2585, 2935, 3382 and 3950.

This book has been organised into four parts: Modelling and Design of Agent Systems, dealing with some specific aspects of modelling agent systems, Modelling Open Agent Systems, dealing with design issues that arise when dealing with agents in the Internet environment, Formal Reasoning About Designs, which looks at the use of reasoning methods to analyse designs, and finally Testing, Debugging and Evolvability.

Part I: Modelling and Design of Agent Systems

The first part focusses on issues of modelling and design in agent systems. This is an extremely important activity and one which has, over the last few years, received a great deal of attention within a range of AOSE methodologies. The three papers in this part address specific aspects of modelling and design for agent systems.

The first paper “An Agent Environment Interaction Model” by Scott DeLoach and Jorge Valenzuela looks in detail at how to design and specify the interface of an agent system with its environment, using *actions* to represent both sensors and effectors. Agents exist over time, in environments which are dynamic and changing, and they typically affect their environments. Consequently the specification of the environment and the agent’s interaction with it is a key part of modelling agent systems. The approach described is integrated into O-MASE, the extended version of the well established MASE methodology developed by the first author.

The second paper on “Allocating Goals to Agent Roles during MAS Requirements Engineering” by Jureta et al. explores how to design roles, and ultimately agents, to ensure that non-functional goals are addressed. They provide a systematic approach for assigning non-functional goals to roles, and heuristics for selecting between different options. Focussing on this assignment of goals to roles at an early stage in the process allows agent organisational structures to emerge from the role definitions.

The third and final paper in this part by Garcia, Choren and von Flach, entitled “An Aspect-Oriented Modeling Framework for Multi-Agent Systems Design” is about modelling concerns that cut across all or many parts of an agent application such as mobility, error handling or security. They build on *Aspect Oriented Programming*, introducing a meta-modelling framework for representing these crosscutting concerns in an agent oriented design. They integrate aspect-oriented abstractions into their agent oriented modelling language called ANote.

Part II: Modelling Open Agent Systems

Part two deals with some of the complexities that arise when dealing with agents in the Internet environment. Two papers deal with design of governance structures for providing some control over autonomous agents, while one deals with modelling agent mobility.

Kusek and Jezic’s paper “Extending UML Sequence Diagrams to Model Agent Mobility” looks at a number of different ways to potentially model agent mobility, using extensions of UML sequence diagrams. Their aim is to capture agent creation, migration paths, and current location. They evaluate the strengths and the weaknesses of the different approaches based on clarity, space needed for representing larger systems, and representation of mobility. They conclude that choice of the most preferred approach depends on the application characteristics of how many agents and nodes there are in the system to be modelled.

The papers “Applying the Governance Framework Technique to Promote Maintainability in Open Multi-Agent Systems” by Carvalho et al., and “Designing Institutional Multi-Agent Systems” by Sierra et al., both deal with specifying the institutional structures within which agents may interact, and which provide some guarantees about the behaviours. Both focus on specifying agent interaction patterns or templates, and on the ability to express norms or constraints regarding agent behaviour. Carvalho et al. use XMLaw and template structures. Sierra et al. describe the methodology for developing a design in the Islander tool, which also captures interaction specifications, and norms and constraints. The methodology used by Sierre et al. is integrated into the Prometheus methodology as a social or organisational design layer.

Part III: Formal Reasoning About Designs

One of the trends in software engineering, and certainly in agent oriented software engineering, is to incorporate automated reasoning into design tools to aid the designer in various ways. This part presents three papers with this general focus.

The first paper, "Modeling Mental States in the Analysis of Multiagent Systems Requirements" by Lapouchnian and Lespérance looks at formal analysis by taking an i* specification and mapping it to the Cognitive Agents Specification Language (CASL). CASL relies heavily on ConGolog for specification of procedural aspects, and also on modal logics and possible world semantics. The developer annotates an i* specification and specifies how elements are to be mapped to the procedural component of CASL. Some transformations are automated. Once the formal specification exists it becomes possible to do formal analysis of such things as epistemic feasibility of plans or termination.

The second paper, by Brandão et al., entitled "Observed-MAS: An Ontology-Based Method for Analyzing Multi-Agent Systems Design Models" focusses on translating design models to formal ontologies, which describe the Multi Agent Systems domain. The ontologies are represented in a Description Logic system and enable analysis of the design using defined queries, which are represented by ontology instances. Analysis is done in two phases–the first within individual diagrams while the second looks at relationships between diagrams. The authors argue that while it is difficult to analyze and establish the well-formedness of a set of diagrams of a UML-like object-oriented modeling language, it gets far more complex when the language is extended to add a set of agency related abstractions. Their approach helps to tame this complexity.

The third paper entitled "Using Risk Analysis to Evaluate Design Alternatives" by Asnar, Bryl and Giorgini, looks at using planning to propose design alternatives, based on risk-related metrics, which are particularly important in certain kinds of systems where availability and reliability are crucial. While the developer must be involved in the reasoning process to agree to any loosening of constraints, the system they describe provides automated reasoning to suggest viable alternatives. They illustrate their approach using an Air Traffic Management case study.

Part IV: Testing, Debugging and Evolvability

As is well known, implementation is not the final stage of system development. Systems must always be tested and debugged, and typically they also evolve once they are deployed, sometimes becoming whole product lines of related systems. These last four papers look at these aspects of developing agent systems.

Tiryaki et al. describe "SUNIT: A Unit Testing Framework for Test Driven Development of Multi-Agent Systems", which is based on an extension of the JUnit framework. They propose a test driven multi-agent system development approach that naturally supports iterative and incremental MAS construction. This approach is supported by their SUnit system.

The second paper in this part "Monitoring Group Behavior in Goal-Directed Agents Using Co-efficient Plan Observation" by Sudeikat and Renz describes an approach to validating the multi-agent cooperative behaviour of a system. They argue that goal hierarchies developed during requirements engineering, combined with Belief Desire Intention architectures, are suitable as a basis for development of a modular approach to checking crosscutting concerns in (BDI) agent implementations. They provide a case study to illustrate their approach.

The third paper, by Jayatilleke et al., "Evaluating a Model Driven Development Toolkit for Domain Experts to Modify Agent Based Systems" describes evaluation of a toolkit designed to allow domain experts to themselves modify and evolve an agent application that has been built using this toolkit. The toolkit builds on design documentation, but provides increased granularity at the detailed design level, enabling production of fully functional code. Domain experts then need only change at the design level, in order to obtain an enhanced implementation. Meteorologists were able to modify an example system that was based on a real application and actual evolutionary changes to the system.

Finally, the paper entitled "Building the Core Architecture of a NASA Multiagent System Product Line" by Peña et al. describes techniques adapted from the field of Software Product Lines (SPL) to enable building of the core architecture for a multiagent system where components can be reused to derive related concrete products with greatly reduced time-to-market and costs. They illustrate the approach with examples from a NASA mission.

These papers provide a diverse and interesting overview of the work that is currently being undertaken by a growing number of researchers and research groups in the area of Agent Oriented Software Engineering. They represent leading edge research in this field, which is of critical importance in facilitating industry take-up of powerful agent technologies.

December 2006

Lin Padgham
Franco Zambonelli

Organization

Organizing Committee

Lin Padgham (Co-chair)
RMIT, Australia
Email: linpa@cs.rmit.edu.au

Franco Zambonelli (Co-chair)
University of Modena e Reggio Emilia, Italy
Email: franco.zambonelli@unimore.it

Steering Committee

Paolo Ciancarini, University of Bologna, Italy
Jörg Müller, Clausthal University of Technology, Germany
Gerhard Weiß, Software Competence Center, Hagenberg
Michael Wooldridge, University of Liverpool, UK

Program Committee

Bernard Bauer (Germany)
Federico Bergenti (Italy)
Carole Bernon (France)
Giacomo Cabri (Italy)
Luca Cernuzzi (Paraguay)
Paolo Ciancarini (Italy)
Massimo Cossentino (Italy)
Keith Decker (USA)
Scott DeLoach (USA)
Klaus Fischer (Germany)
Paolo Giorgini (Italy)
Michael Huhns (USA)
Gaya Jayatilleke (Australia)
Juergen Lind (Germany)
Mike Luck (UK)
Andrea Omicini (Italy)
Van Parunak (USA)
Anna Perini (Italy)
Fariba Sadri (UK)
Onn Shehory (Israel)
Michael Winikoff (Australia)
Mike Wooldridge (UK)
Laura Zavala (USA)

Table of Contents

Modelling and Design of Agent Systems

Modelling Open Agent Systems

Formal Reasoning About Designs

Testing, Debugging and Evolvability