State-of-the-Art Survey

LNCS 4405

Lin Padgham
Franco Zambonelli (Eds.)

Agent-Oriented Software Engineering VII

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

Move
Flight and Orbit
Walk
Gas prop.
Use Sail to Orbit and flight
Snake
Amoeba
Rolling

Springer

5-53

TP311.5-53
A265
2006

Lin Padgham   Franco Zambonelli (Eds.)

Agent-Oriented Software Engineering VII

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

华南理工大学
图书馆藏书

Springer

E2007003169

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

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2007

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

此为试读，需要完整PDF请访问：www.ertongbook.com

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)

Lecture Notes in Computer Science 4405

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

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Lecture Notes in Computer Science

For information about Vols. 1–4289

please contact your bookseller or Springer

Vol. 4405: L. Padgham, F. Zambonelli (Eds.), Agent-Oriented Software Engineering VII. XII, 225 pages. 2007.

Vol. 4390: S.O. Kuznetsov, S. Schmidt (Eds.), Formal Concept Analysis. X, 329 pages. 2007. (Sublibrary LNAI).

Vol. 4385: K. Coninx, K. Luyten, K.A. Schneider (Eds.), Task Models and Diagrams for Users Interface Design. XI, 355 pages. 2007.

Vol. 4384: T. Washio, K. Satoh, H. Takeda, A. Inokuchi (Eds.), New Frontiers in Artificial Intelligence. IX, 401 pages. 2007. (Sublibrary LNAI).

Vol. 4381: J. Akiyama, W.Y.C. Chen, M. Kano, X. Li, Q. Yu (Eds.), Discrete Geometry, Combinatorics and Graph Theory. XI, 289 pages. 2007.

Vol. 4380: S. Spaccapietra, P. Atzeni, F. Fages, M.-S. Hacid, M. Kifer, J. Mylopoulos, B. Pernici, P. Shvaiko, J. Trujillo, I. Zaihrayeu (Eds.), Journal on Data Semantics VIII. XV, 219 pages. 2007.

Vol. 4378: I. Virbitskaite, A. Voronkov (Eds.), Perspectives of Systems Informatics. XIV, 496 pages. 2007.

Vol. 4377: M. Abe (Ed.), Topics in Cryptology – CT-RSA 2007. XI, 403 pages. 2006.

Vol. 4373: K. Langendoen, T. Voigt (Eds.), Wireless Sensor Networks. XIII, 358 pages. 2007.

Vol. 4372: M. Kaufmann, D. Wagner (Eds.), Graph Drawing. XIV, 454 pages. 2007.

Vol. 4371: K. Inoue, K. Satoh, F. Toni (Eds.), Computational Logic in Multi-Agent Systems. X, 315 pages. 2007. (Sublibrary LNAI).

Vol. 4369: M. Umeda, A. Wolf, O. Bartenstein, U. Geske, D. Seipel, O. Takata (Eds.), Declarative Programming for Knowledge Management. X, 229 pages. 2006. (Sublibrary LNAI).

Vol. 4368: T. Erlebach, C. Kaklamanis (Eds.), Approximation and Online Algorithms. X, 345 pages. 2007.

Vol. 4367: K. De Bosschere, D. Kaeli, P. Stenström, D. Whalley, T. Ungerer (Eds.), High Performance Embedded Architectures and Compilers. XI, 307 pages. 2007.

Vol. 4364: T. Kühne (Ed.), Models in Software Engineering. XI, 332 pages. 2007.

Vol. 4362: J. van Leeuwen, G.F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plášil (Eds.), SOFSEM 2007: Theory and Practice of Computer Science. XXI, 937 pages. 2007.

Vol. 4361: H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. IX, 555 pages. 2006.

Vol. 4360: W. Dubitzky, A. Schuster, P.M.A. Sloot, M. Schroeder, M. Romberg (Eds.), Distributed, High-Performance and Grid Computing in Computational Biology. X, 192 pages. 2007. (Sublibrary LNBI).

Vol. 4358: R. Vidal, A. Heyden, Y. Ma (Eds.), Dynamical Vision. IX, 329 pages. 2007.

Vol. 4357: L. Buttyán, V. Gligor, D. Westhoff (Eds.), Security and Privacy in Ad-Hoc and Sensor Networks. X, 193 pages. 2006.

Vol. 4355: J. Julliand, O. Kouchnarenko (Eds.), B 2007: Formal Specification and Development in B. XIII, 293 pages. 2006.

Vol. 4354: M. Hanus (Ed.), Practical Aspects of Declarative Languages. X, 335 pages. 2006.

Vol. 4353: T. Schwentick, D. Suciu (Eds.), Database Theory – ICDT 2007. XI, 419 pages. 2006.

Vol. 4352: T.-J. Cham, J. Cai, C. Dorai, D. Rajan, T.-S. Chua, L.-T. Chia (Eds.), Advances in Multimedia Modeling, Part II. XVIII, 743 pages. 2006.

Vol. 4351: T.-J. Cham, J. Cai, C. Dorai, D. Rajan, T.-S. Chua, L.-T. Chia (Eds.), Advances in Multimedia Modeling, Part I. XIX, 797 pages. 2006.

Vol. 4349: B. Cook, A. Podelski (Eds.), Verification, Model Checking, and Abstract Interpretation. XI, 395 pages. 2007.

Vol. 4348: S.T. Taft, R.A. Duff, R.L. Brukardt, E. Ploedereder, P. Leroy (Eds.), Ada 2005 Reference Manual. XXII, 765 pages. 2006.

Vol. 4347: J. Lopez (Ed.), Critical Information Infrastructures Security. X, 286 pages. 2006.

Vol. 4345: N. Maglaveras, I. Chouvarda, V. Koutkias, R. Brause (Eds.), Biological and Medical Data Analysis. XIII, 496 pages. 2006. (Sublibrary LNBI).

Vol. 4344: V. Gruhn, F. Oquendo (Eds.), Software Architecture. X, 245 pages. 2006.

Vol. 4342: H. de Swart, E. Orłowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments II. X, 373 pages. 2006. (Sublibrary LNAI).

Vol. 4341: P.Q. Nguyen (Ed.), Progress in Cryptology - VIETCRYPT 2006. XI, 385 pages. 2006.

Vol. 4340: R. Prodan, T. Fahringer, Grid Computing. XXIII, 317 pages. 2007.

Vol. 4339: E. Ayguadé, G. Baumgartner, J. Ramanujam, P. Sadayappan (Eds.), Languages and Compilers for Parallel Computing. XI, 476 pages. 2006.

Vol. 4338: P. Kalra, S. Peleg (Eds.), Computer Vision, Graphics and Image Processing. XV, 965 pages. 2006.

Vol. 4337: S. Arun-Kumar, N. Garg (Eds.), FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science. XIII, 430 pages. 2006.

Vol. 4335: S.A. Brueckner, S. Hassas, M. Jelasity, D. Yamins (Eds.), Engineering Self-Organising Systems. XII, 212 pages. 2007. (Sublibrary LNAI).

Vol. 4334: B. Beckert, R. Hähnle, P.H. Schmitt (Eds.), Verification of Object-Oriented Software. XXIX, 658 pages. 2007. (Sublibrary LNAI).

Vol. 4333: U. Reimer, D. Karagiannis (Eds.), Practical Aspects of Knowledge Management. XII, 338 pages. 2006. (Sublibrary LNAI).

Vol. 4332: A. Bagchi, V. Atluri (Eds.), Information Systems Security. XV, 382 pages. 2006.

Vol. 4331: G. Min, B. Di Martino, L.T. Yang, M. Guo, G. Ruenger (Eds.), Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops. XXXVII, 1141 pages. 2006.

Vol. 4330: M. Guo, L.T. Yang, B. Di Martino, H.P. Zima, J. Dongarra, F. Tang (Eds.), Parallel and Distributed Processing and Applications. XVIII, 953 pages. 2006.

Vol. 4329: R. Barua, T. Lange (Eds.), Progress in Cryptology - INDOCRYPT 2006. X, 454 pages. 2006.

Vol. 4328: D. Penkler, M. Reitenspiess, F. Tam (Eds.), Service Availability. X, 289 pages. 2006.

Vol. 4327: M. Baldoni, U. Endriss (Eds.), Declarative Agent Languages and Technologies IV. VIII, 257 pages. 2006. (Sublibrary LNAI).

Vol. 4326: S. Göbel, R. Malkewitz, I. Iurgel (Eds.), Technologies for Interactive Digital Storytelling and Entertainment. X, 384 pages. 2006.

Vol. 4325: J. Cao, I. Stojmenovic, X. Jia, S.K. Das (Eds.), Mobile Ad-hoc and Sensor Networks. XIX, 887 pages. 2006.

Vol. 4323: G. Doherty, A. Blandford (Eds.), Interactive Systems. XI, 269 pages. 2007.

Vol. 4320: R. Gotzhein, R. Reed (Eds.), System Analysis and Modeling: Language Profiles. X, 229 pages. 2006.

Vol. 4319: L.-W. Chang, W.-N. Lie (Eds.), Advances in Image and Video Technology. XXVI, 1347 pages. 2006.

Vol. 4318: H. Lipmaa, M. Yung, D. Lin (Eds.), Information Security and Cryptology. XI, 305 pages. 2006.

Vol. 4317: S.K. Madria, K.T. Claypool, R. Kannan, P. Uppuluri, M.M. Gore (Eds.), Distributed Computing and Internet Technology. XIX, 466 pages. 2006.

Vol. 4316: M.M. Dalkilic, S. Kim, J. Yang (Eds.), Data Mining and Bioinformatics. VIII, 197 pages. 2006. (Sublibrary LNBI).

Vol. 4314: C. Freksa, M. Kohlhase, K. Schill (Eds.), KI 2006: Advances in Artificial Intelligence. XII, 458 pages. 2007. (Sublibrary LNAI).

Vol. 4313: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods. IX, 197 pages. 2006.

Vol. 4312: S. Sugimoto, J. Hunter, A. Rauber, A. Morishima (Eds.), Digital Libraries: Achievements, Challenges and Opportunities. XVIII, 571 pages. 2006.

Vol. 4311: K. Cho, P. Jacquet (Eds.), Technologies for Advanced Heterogeneous Networks II. XI, 253 pages. 2006.

Vol. 4309: P. Inverardi, M. Jazayeri (Eds.), Software Engineering Education in the Modern Age. VIII, 207 pages. 2006.

Vol. 4308: S. Chaudhuri, S.R. Das, H.S. Paul, S. Tirthapura (Eds.), Distributed Computing and Networking. XIX, 608 pages. 2006.

Vol. 4307: P. Ning, S. Qing, N. Li (Eds.), Information and Communications Security. XIV, 558 pages. 2006.

Vol. 4306: Y. Avrithis, Y. Kompatsiaris, S. Staab, N.E. O'Connor (Eds.), Semantic Multimedia. XII, 241 pages. 2006.

Vol. 4305: A.A. Shvartsman (Ed.), Principles of Distributed Systems. XIII, 441 pages. 2006.

Vol. 4304: A. Sattar, B.-H. Kang (Eds.), AI 2006: Advances in Artificial Intelligence. XXVII, 1303 pages. 2006. (Sublibrary LNAI).

Vol. 4303: A. Hoffmann, B.-H. Kang, D. Richards, S. Tsumoto (Eds.), Advances in Knowledge Acquisition and Management. XI, 259 pages. 2006. (Sublibrary LNAI).

Vol. 4302: J. Domingo-Ferrer, L. Franconi (Eds.), Privacy in Statistical Databases. XI, 383 pages. 2006.

Vol. 4301: D. Pointcheval, Y. Mu, K. Chen (Eds.), Cryptology and Network Security. XIII, 381 pages. 2006.

Vol. 4300: Y.Q. Shi (Ed.), Transactions on Data Hiding and Multimedia Security I. IX, 139 pages. 2006.

Vol. 4299: S. Renals, S. Bengio, J.G. Fiscus (Eds.), Machine Learning for Multimodal Interaction. XII, 470 pages. 2006.

Vol. 4297: Y. Robert, M. Parashar, R. Badrinath, V.K. Prasanna (Eds.), High Performance Computing - HiPC 2006. XXIV, 642 pages. 2006.

Vol. 4296: M.S. Rhee, B. Lee (Eds.), Information Security and Cryptology – ICISC 2006. XIII, 358 pages. 2006.

Vol. 4295: J.D. Carswell, T. Tezuka (Eds.), Web and Wireless Geographical Information Systems. XI, 269 pages. 2006.

Vol. 4294: A. Dan, W. Lamersdorf (Eds.), Service-Oriented Computing – ICSOC 2006. XIX, 653 pages. 2006.

Vol. 4293: A. Gelbukh, C.A. Reyes-Garcia (Eds.), MICAI 2006: Advances in Artificial Intelligence. XXVIII, 1232 pages. 2006. (Sublibrary LNAI).

Vol. 4292: G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, A. Nefian, G. Meenakshisundaram, V. Pascucci, J. Zara, J. Molineros, H. Theisel, T. Malzbender (Eds.), Advances in Visual Computing, Part II. XXXII, 906 pages. 2006.

Vol. 4291: G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, A. Nefian, G. Meenakshisundaram, V. Pascucci, J. Zara, J. Molineros, H. Theisel, T. Malzbender (Eds.), Advances in Visual Computing, Part I. XXXI, 916 pages. 2006.

Vol. 4290: M. van Steen, M. Henning (Eds.), Middleware 2006. XIII, 425 pages. 2006.

¥581.00元

此为试读,需要完整PDF请访问: www.ertongbook.com

Table of Contents

Modelling and Design of Agent Systems

Modelling Open Agent Systems

Formal Reasoning About Designs

Testing, Debugging and Evolvability

An Agent-Environment Interaction Model

Scott A. DeLoach and Jorge L. Valenzuela

Department of Computing and Information Sciences, Kansas State University
234 Nichols Hall, Manhattan, KS 66506
{sdeloach, jvalenzu}@cis.ksu.edu

Abstract. This paper develops a model for precisely defining how an agent interacts with objects in its environment through the use of its capabilities. Capabilities are recursively defined in terms of lower-level capabilities and actions, which represent atomic interactions with the environment. Actions are used to represent both sensors and effectors. The paper shows how the model can be used to represent both software and physical agents and their capabilities. The paper also shows how the model can be integrated into the Organization-based Multiagent Systems Engineering methodology.

1 Introduction

There is widespread agreement that the environment in which a multiagent system is situated is of fundamental importance in the analysis, design, and operation of the system. However, even with this agreement, few multiagent methodologies include the modeling of the environment or the agent's interactions with it as first class entities [10]. In situated multiagent systems, the *environment* is the entity in which agents exist and communicate [6]. Communication is a critical factor that enables agents to interact and coordinate. Typically, this interaction and coordination is modeled using direct communication through the social environment; however, it can also be modeling indirectly through the physical environment. A *social environment* is the entity that provides the principles, processes and structures that enable the agents to communicate while the *physical environment* provides principles and processes that affect objects within an environment [6]. In [4], Ferber defines a multiagent system as having six basic entities:

- An environment, E
- A set of objects, O, that exist in E
- A set of agents, A, which are active objects (i.e., a subset of O)
- A set of relations, R, that define relationships between objects in O
- A set of operations, O, that agents can use to sense and affect objects in O
- A set of universal laws that define the reaction of the environment to agent operations

Based on Ferber's definition, we have identified *five requirements* for specifying agent-environment interaction model. Essentially, an AEI should define:

L. Padgham and F. Zambonelli (Eds.): AOSE 2006, LNCS 4405, pp. 1–18, 2007.
© Springer-Verlag Berlin Heidelberg 2007

1. A unique entity called the environment
2. The set of objects in the environment (which includes agents)
3. Specific types of relations that may exist between objects in the environment
4. The set of operations that agents may perform upon objects in the environment
5. The laws that govern the effect of those operations on objects in the environment

While capturing these elements is essential, we believe it is also critical that these concepts be captured using a model that shows direct relations between the objects, agents, and actions as well as specifies the intended effect of each action unambiguously. We believe it is also important to provide a model that allows these concepts to be specified and viewed at the appropriate level of abstraction.

While most current multiagent methodologies provide some notion of the environment or the agent's interactions with it, no major methodologies possess a detailed agent-environment interaction model that explicitly defines how the environment is affected by agents or how the agent perceives the environment. Including such an agent-environment interaction model is important because it allows us to explicitly identify (1) how agents directly interact/coordinate with each other, (2) how agents indirectly interact/coordinate with each other, and (3) the effect of agents on objects in the environment, which in situated multiagent systems often determines whether the system has accomplished its goals. In addition, agents in situated multiagent systems also generally require some representation of the environment in order to effectively communicate with other agents and to achieve their goals. By including a well-defined model of the environment in the agent-environment interaction model, the analysis, and design of these agents should be clearer and thus improved over implicit approaches.

The goal of this paper is to present an Agent-Environment Interaction model (AEI) that can be integrated into appropriate multiagent systems methodologies. Specifically, we will integrate the AEI Model into the Organization-based Multiagent Systems Engineering methodology (O-MaSE) [1]. To make the notation as clear and unambiguous as possible, we use standard UML notation with liberal use of keywords to denote specific concepts in the model. Obviously, if our AEI Model is integrated into other methodologies and modeling approaches, the notation can be adapted as needed.

The paper is organized as follow. In Section 2, we discuss how some current multiagent methodologies address environmental issues and provide an overview of O-MaSE [1]. In Section 3, we present our AEI Model and integrate our AEI Model into O-MaSE. In Section 4, we present a detailed example of the AEI Model using a robotics Weapons of Mass Destruction (WMD) simulation system. Finally, in Section 5, we present our conclusions and areas for future work.

2 Related Work

In this section we review four prominent multiagent systems methodologies and how they model interactions with the environment: Gaia, Message, Prometheus, and O-MaSE. We also analyze how well each of these methodologies meets the agent environment interaction requirements stated above.

2.1 Gaia

The extended version of Gaia [12] adds some basic concepts and organizational abstractions to the original version of GAIA [11]. Among these additions is an Environment Model, which is introduced during the analysis phase. Because the authors believe that "it is difficult to provide a general modeling abstraction and general modeling techniques because the environment for different applications can be very different in nature" [12], they model environmental entities in terms of abstract computational resources. These resources are modeled as tuples that the agents may read, (sense), effect, (change), or consume, (remove). Thus the Gaia Environment Model can be viewed as a list of resources that can be accessed using an associated name and acted upon based on the type of action associated with them. An example of a Gaia Environment Model is shown below [12].

```
reads var1 // readable resource of the environment.
var2 // another readable resource.
change var3 // a variable that can be also changed by the agent.
```

Analyzing the Gaia Environment Model using our five AEI model requirements shows that, while it does include a limited notion of objects, it does not include any notion of agents (requirement 2). In addition, the Gaia Environment Model severely limits the types of relations (requirement 3) and actions (requirement 4) that can be performed on those objects. Finally, the Environment Model has no notion of environmental laws that affect the environment objects independently of the agents (requirement 5). A more general notion of an AEI could be of benefit in the Gaia methodology.

2.2 MESSAGE

In the MESSAGE methodology [5], the MESSAGE modeling language defines some knowledge-level-concepts like *Concrete-Entity*, *Activity*, and *MentalStateEntity*. One of the concrete entities defined is *Agents*, which are autonomous entities that can perform actions that affect resources. The *Actions/Activities* are concrete entities and include *Tasks* and *Interaction Protocols*. Agents can also perceive information entries that describe the state of a resource. Another concrete entity is a *Resource*, which represents a non-autonomous entity that agents can access/use.MESSAGE builds five views of the Analysis Model: Organization, Goal/Task, Agent/Role, Interaction and Domain views. The *Organization view* shows the concrete entities in the system, the environment and the relationship among them.

Based on our requirements, we see that MESSAGE defines elements of its environment as containing objects (both agents and resources) that can interact using actions and messages. However, MESSAGE does not include the notion of environmental laws that affect the objects in the environment (requirement 5).

Even though MESSAGE captures most of the required information, it does not explicitly define an agent-environment interaction model and does not provide a flexible way to represent or define actions at an appropriate level of abstraction.

2.3 Prometheus

The aim of the Prometheus System Identification Phase is to identify the basic functionality of the system along with the inputs, outputs, and important data structures [7]. Prometheus models these inputs as *percepts* and defines them as raw data coming from the environment. Outputs are modeled as *actions*, which are defined as the agent's way to modify the environment. Scenarios are used in Prometheus to describe how the system operates nominally. Each scenario consists of a set of steps that can include goals, actions, percepts, scenarios, or "other" for special types of steps.

The architectural design phase focuses on identifying the agents in the system and their interaction. Once the agents are identified, the next step is to define the percepts each agent reacts to and the actions it may perform. Agent interaction is specified by defining messages and the different repositories to be used. All these items are depicted in the system overview diagram. The Detailed Design Phase focuses on defining the capabilities, which are defined in terms of internal events, plans, and detailed data structures of the agents. Each capability is described by a descriptor, which includes the definition of its percepts, actions, data read or written, interaction with other capabilities, and sub-capabilities.

Our analysis reveals that Prometheus does not explicitly define the environment. It does not define the objects in the environment (requirement 1), the relationships between them (requirement 2), or the laws that govern the effect of agent's actions on the environment (requirement 5). However, Prometheus does capture the operations that it uses to get percepts from the environment and perform actions on the environment. Thus Prometheus too could benefit from an explicit AEI Model.

2.4 Organization-Based Multiagent Systems Engineering

The Organization-Based Multiagent System Engineering (O-MaSE) [1] methodology extends the original MaSE [3] methodology to allow the design of organizational multiagent systems. Some of the weaknesses of MaSE addressed by O-MaSE include the tendency to generate static organizations, the inability to model sub-organizations/systems, and the lack of explicit concepts for modeling interactions with the environment. To model interactions with the environment, O-MaSE represents both the sensing and manipulation of the environment as a type of *Capability*, which is defined as an "atomic entity that defines the agents' abilities; these abilities include *soft* abilities such as access to resources or computational algorithms, as well as *hard* capabilities such as sensors and effectors [1]. We use this notion of capabilities as the foundation for our AEI Model, extending it to allow capability composition as well as to model direct interaction with the environment. The current version of the O-MaSE metamodel is

此为试读,需要完整PDF请访问: www.ertongbook.com