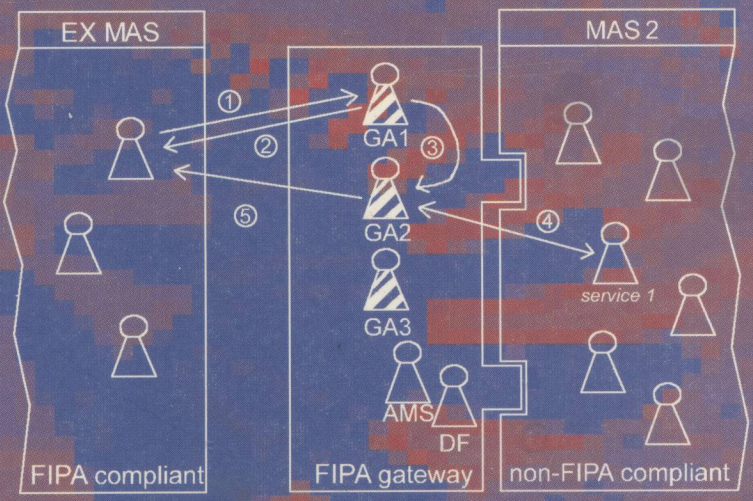
State-of-the-Art Survey

LNCS 2935

Paolo Giorgini
Jörg P. Müller
James Odell (Eds.)

Agent-Oriented Software Engineering IV

4th International Workshop, AOSE 2003
Melbourne, Australia, July 2003
Revised Papers



Springer

Paolo Giorgini Jörg P. Müller
James Odell (Eds.)

Agent-Oriented Software Engineering IV

4th International Workshop, AOSE 2003
Melbourne, Australia, July 15, 2003
Revised Papers

Springer

Volume Editors

Paolo Giorgini
University of Trento, Department of Information and Communication Technology
Via Sommarive, 14, 38050 Povo, Trento, Italy
E-mail: paolo.giorgini@dit.unitn.it

Jörg P. Müller
Siemens AG, Corporate Technology
Intelligent Autonomous Systems
Otto-Hahn-Ring 6, 81730 Munich, Germany
E-mail: joerg.p.mueller@siemens.com

James Odell
James Odell Associates
3646 West Huron River Drive, Ann Arbor, MI 48103, USA
E-mail: email@jamesodell.com

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

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

ISSN 0302-9743
ISBN 3-540-20826-7 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 10981368 06/3142 5 4 3 2 1 0

Preface

The explosive growth of application areas such as electronic commerce, enterprise resource planning and mobile computing has profoundly and irreversibly changed our views on software systems. Nowadays, software is to be based on open architectures that continuously change and evolve to accommodate new components and meet new requirements. Software must also operate on different platforms, without recompilation, and with minimal assumptions about its operating environment and its users. Furthermore, software must be robust and autonomous, capable of serving a naive user with a minimum of overhead and interference.

Agent concepts hold great promise for responding to the new realities of software systems. They offer higher-level abstractions and mechanisms that address issues such as knowledge representation and reasoning, communication, coordination, cooperation among heterogeneous and autonomous parties, perception, commitments, goals, beliefs, and intentions, all of which need conceptual modeling. On the one hand, the concrete implementation of these concepts can lead to advanced functionalities, e.g., in inference-based query answering, transaction control, adaptive workflows, brokering and integration of disparate information sources, and automated communication processes. On the other hand, their rich representational capabilities allow more faithful and flexible treatments of complex organizational processes, leading to more effective requirements analysis and architectural/detailed design.

In keeping with its very successful predecessors, AOSE 2000, AOSE 2001, and AOSE 2002 (Lecture Notes in Computer Science Volumes 1957, 2222, and 2585), the AOSE 2003 workshop sought to examine the credentials of agent-based approaches as a software engineering paradigm, and to gain an insight into what agent-oriented software engineering will look like.

AOSE 2003 was hosted by the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003) held in Melbourne, Australia on July 2003. The workshop received 43 submissions, and 15 of them were accepted for presentation (an acceptance rate of 30%). These papers were reviewed by at least 3 members of an international program committee composed of 25 researchers. The submissions followed a call for papers on all aspects of agent-oriented software engineering, and showed the range of results achieved in several areas, such as methodologies, modeling, architectures, and tools.

The workshop program included an invited talk, a technical session in which the accepted papers were presented and discussed, and a closing plenary session. It congregated more than 50 attendees, among them researchers, students, and practitioners, who contributed to the discussion of research problems related to the main topics in AOSE.

This volume contains revised versions of the 15 papers presented at the workshop. Additionally, it contains an invited contribution by Bernhard Bauer and Jörg Müller on "Using UML in the Context of Agent-Oriented Software Engineering: State of the Art." We believe that this thoroughly prepared volume

is of particular value to all readers interested in the key topics and most recent developments in the very exciting field of agent-oriented software engineering.

We thank the authors, the participants, and the reviwers for making AOSE 2003 a high-quality scientific event.

November 2003

Paolo Giorgini
Jörg P. Müller
James Odell

Organization

Organizing Committee

Paolo Giorgini (Co-chair)
Department of Information and Communication Technology
University of Trento, Italy
Email: paolo.giorgini@dit.unitn.it

Jörg P. Müller (Co-chair)
Siemens AG, Germany
Email: joerg.mueller@mchp.siemens.de

James Odell (Co-chair)
James Odell Associates, Ann Arbor, MI, USA
Email: email@jamesodell.com

Steering Committee

Paolo Ciancarini, University of Bologna, Italy
Gerhard Weiss, Technische Universitaet Muenchen, Germany
Michael Wooldridge, University of Liverpool, UK

Program Committee

Bernard Bauer (Germany)
Federico Bergenti (Italy)
Scott DeLoach (USA)
Marie-Pierre Gervais (France)
Olivier Gutknecht (France)
Brian Henderson-Sellers (Australia)
Michael Huhns (USA)
Carlos Iglesias (Spain)
Nicholas Jennings (UK)
Catholijn Jonker (Netherlands)
Liz Kendall (Australia)
David Kinny (Australia)
Manuel Kolp (Belgium)
Yannis Labrou (USA)
Juergen Lind (Germany)
John Mylopolous (Canada)
Andrea Omicini (Italy)
Van Parunak (USA)
Anna Perini (Italy)
Marco Pistore (Italy)
Onn Shehory (Israel)
Gerhard Weiss (Germany)
Paola Turci (Italy)
Eric Yu (Canada)
Franco Zambonelli (Italy)

Auxiliary Reviewers: Paolo Busetta, Julio Cesar Leite, Aizhong Lin, Matthias Nickles, Michael Rovatsos, Marco Roveri, Arnon Sturm, Angelo Susi, Martijn Schut

Lecture Notes in Computer Science 2935

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Author Index

Lecture Notes in Computer Science

For information about Vols. 1–2844
please contact your bookseller or Springer-Verlag

Vol. 2845: B. Christianson, B. Crispo, J.A. Malcolm, M. Roe (Eds.), Security Protocols. Proceedings, 2002. VIII, 243 pages. 2004.

Vol. 2847: R. de Lemos, T.S. Weber, J.B. Camargo Jr. (Eds.), Dependable Computing. Proceedings, 2003. XIV, 371 pages. 2003.

Vol. 2848: F.E. Fich (Ed.), Distributed Computing. Proceedings, 2003. X, 367 pages. 2003.

Vol. 2849: N. García, J.M. Martínez, L. Salgado (Eds.), Visual Content Processing and Representation. Proceedings, 2003. XII, 352 pages. 2003.

Vol. 2850: M.Y. Vardi, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. Proceedings, 2003. XIII, 437 pages. 2003. (Subseries LNAI)

Vol. 2851: C. Boyd, W. Mao (Eds.), Information Security. Proceedings, 2003. XI, 443 pages. 2003.

Vol. 2852: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. Proceedings, 2003. VIII, 509 pages. 2003.

Vol. 2853: M. Jeckle, L.-J. Zhang (Eds.), Web Services – ICWS-Europe 2003. Proceedings, 2003. VIII, 227 pages. 2003.

Vol. 2854: J. Hoffmann, Utilizing Problem Structure in Planning. XIII, 251 pages. 2003. (Subseries LNAI)

Vol. 2855: R. Alur, I. Lee (Eds.), Embedded Software. Proceedings, 2003. X, 373 pages. 2003.

Vol. 2856: M. Smirnov, E. Biersack, C. Blondia, O. Bonaventure, O. Casals, G. Karlsson, George Pavlou, B. Quoitin, J. Roberts, I. Stavrakakis, B. Stiller, P. Trimintzios, P. Van Mieghem (Eds.), Quality of Future Internet Services. IX, 293 pages. 2003.

Vol. 2857: M.A. Nascimento, E.S. de Moura, A.L. Oliveira (Eds.), String Processing and Information Retrieval. Proceedings, 2003. XI, 379 pages. 2003.

Vol. 2858: A. Veidenbaum, K. Joe, H. Amano, H. Aiso (Eds.), High Performance Computing. Proceedings, 2003. XV, 566 pages. 2003.

Vol. 2859: B. Apolloni, M. Marinaro, R. Tagliaferri (Eds.), Neural Nets. Proceedings, 2003. X, 376 pages. 2003.

Vol. 2860: D. Geist, E. Tronci (Eds.), Correct Hardware Design and Verification Methods. Proceedings, 2003. XII, 426 pages. 2003.

Vol. 2861: C. Bliek, C. Jermann, A. Neumaier (Eds.), Global Optimization and Constraint Satisfaction. Proceedings, 2002. XII, 239 pages. 2003.

Vol. 2862: D. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing. Proceedings, 2003. VII, 269 pages. 2003.

Vol. 2863: P. Stevens, J. Whittle, G. Booch (Eds.), «UML» 2003 – The Unified Modeling Language. Proceedings, 2003. XIV, 415 pages. 2003.

Vol. 2864: A.K. Dey, A. Schmidt, J.F. McCarthy (Eds.), UbiComp 2003: Ubiquitous Computing. Proceedings, 2003. XVII, 368 pages. 2003.

Vol. 2865: S. Pierre, M. Barbeau, E. Kranakis (Eds.), Ad-Hoc, Mobile, and Wireless Networks. Proceedings, 2003. X, 293 pages. 2003.

Vol. 2866: J. Akiyama, M. Kano (Eds.), Discrete and Computational Geometry. Proceedings, 2002. VIII, 285 pages. 2003.

Vol. 2867: M. Brunner, A. Keller (Eds.), Self-Managing Distributed Systems. Proceedings, 2003. XIII, 274 pages. 2003.

Vol. 2868: P. Perner, R. Brause, H.-G. Holzhütter (Eds.), Medical Data Analysis. Proceedings, 2003. VIII, 127 pages. 2003.

Vol. 2869: A. Yazici, C. Şener (Eds.), Computer and Information Sciences – ISCIS 2003. Proceedings, 2003. XIX, 1110 pages. 2003.

Vol. 2870: D. Fensel, K. Sycara, J. Mylopoulos (Eds.), The Semantic Web - ISWC 2003. Proceedings, 2003. XV, 931 pages. 2003.

Vol. 2871: N. Zhong, Z.W. Raś, S. Tsumoto, E. Suzuki (Eds.), Foundations of Intelligent Systems. Proceedings, 2003. XV, 697 pages. 2003. (Subseries LNAI)

Vol. 2873: J. Lawry, J. Shanahan, A. Ralescu (Eds.), Modelling with Words. XIII, 229 pages. 2003. (Subseries LNAI)

Vol. 2874: C. Priami (Ed.), Global Computing. Proceedings, 2003. XIX, 255 pages. 2003.

Vol. 2875: E. Aarts, R. Collier, E. van Loenen, B. de Ruyter (Eds.), Ambient Intelligence. Proceedings, 2003. XI, 432 pages. 2003.

Vol. 2876: M. Schroeder, G. Wagner (Eds.), Rules and Rule Markup Languages for the Semantic Web. Proceedings, 2003. VII, 173 pages. 2003.

Vol. 2877: T. Böhme, G. Heyer, H. Unger (Eds.), Innovative Internet Community Systems. Proceedings, 2003. VIII, 263 pages. 2003.

Vol. 2878: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. Part I. Proceedings, 2003. XXXIII, 819 pages. 2003.

Vol. 2879: R.E. Ellis, T.M. Peters (Eds.), Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003. Part II. Proceedings, 2003. XXXIV, 1003 pages. 2003.

Vol. 2880: H.L. Bodlaender (Ed.), Graph-Theoretic Concepts in Computer Science. Proceedings, 2003. XI, 386 pages. 2003.

Vol. 2881: E. Horlait, T. Magedanz, R.H. Glitho (Eds.), Mobile Agents for Telecommunication Applications. Proceedings, 2003. IX, 297 pages. 2003.

Vol. 2882: D. Veit, Matchmaking in Electronic Markets. XV, 180 pages. 2003. (Subseries LNAI)

Vol. 2883: J. Schaeffer, M. Müller, Y. Björnsson (Eds.), Computers and Games. Proceedings, 2002. XI, 431 pages. 2003.

Vol. 2884: E. Najm, U. Nestmann, P. Stevens (Eds.), Formal Methods for Open Object-Based Distributed Systems. Proceedings, 2003. X, 293 pages. 2003.

Vol. 2885: J.S. Dong, J. Woodcock (Eds.), Formal Methods and Software Engineering. Proceedings, 2003. XI, 683 pages. 2003.

Vol. 2886: I. Nyström, G. Sanniti di Baja, S. Svensson (Eds.), Discrete Geometry for Computer Imagery. Proceedings, 2003. XII, 556 pages. 2003.

Vol. 2887: T. Johansson (Ed.), Fast Software Encryption. Proceedings, 2003. IX, 397 pages. 2003.

Vol. 2888: R. Meersman, Zahir Tari, D.C. Schmidt et al. (Eds.), On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. Proceedings, 2003. XXI, 1546 pages. 2003.

Vol. 2889: Robert Meersman, Zahir Tari et al. (Eds.), On The Move to Meaningful Internet Systems 2003: OTM 2003 Workshops. Proceedings, 2003. XXI, 1096 pages. 2003.

Vol. 2890: M. Broy, A.V. Zamulin (Eds.), Perspectives of System Informatics. Proceedings, 2003. XV, 572 pages. 2003.

Vol. 2891: J. Lee, M. Barley (Eds.), Intelligent Agents and Multi-Agent Systems. Proceedings, 2003. X, 215 pages. 2003. (Subseries LNAI)

Vol. 2892: F. Dau, The Logic System of Concept Graphs with Negation. XI, 213 pages. 2003. (Subseries LNAI)

Vol. 2893: J.-B. Stefani, I. Demeure, D. Hagimont (Eds.), Distributed Applications and Interoperable Systems. Proceedings, 2003. XIII, 311 pages. 2003.

Vol. 2894: C.S. Laih (Ed.), Advances in Cryptology - ASIACRYPT 2003. Proceedings, 2003. XIII, 543 pages. 2003.

Vol. 2895: A. Ohori (Ed.), Programming Languages and Systems. Proceedings, 2003. XIII, 427 pages. 2003.

Vol. 2896: V.A. Saraswat (Ed.), Advances in Computing Science – ASIAN 2003. Proceedings, 2003. VIII, 305 pages. 2003.

Vol. 2897: O. Balet, G. Subsol, P. Torguet (Eds.), Virtual Storytelling. Proceedings, 2003. XI, 240 pages. 2003.

Vol. 2898: K.G. Paterson (Ed.), Cryptography and Coding. Proceedings, 2003. IX, 385 pages. 2003.

Vol. 2899: G. Ventre, R. Canonico (Eds.), Interactive Multimedia on Next Generation Networks. Proceedings, 2003. XIV, 420 pages. 2003.

Vol. 2900: M. Bidoit, P.D. Mosses, CASL User Manual. XIII, 240 pages. 2004.

Vol. 2901: F. Bry, N. Henze, J. Maluszyński (Eds.), Principles and Practice of Semantic Web Reasoning. Proceedings, 2003. X, 209 pages. 2003.

Vol. 2902: F. Moura Pires, S. Abreu (Eds.), Progress in Artificial Intelligence. Proceedings, 2003. XV, 504 pages. 2003. (Subseries LNAI).

Vol. 2903: T.D. Gedeon, L.C.C. Fung (Eds.), AI 2003: Advances in Artificial Intelligence. Proceedings, 2003. XVI, 1075 pages. 2003. (Subseries LNAI).

Vol. 2904: T. Johansson, S. Maitra (Eds.), Progress in Cryptology – INDOCRYPT 2003. Proceedings, 2003. XI, 431 pages. 2003.

Vol. 2905: A. Sanfeliu, J. Ruiz-Shulcloper (Eds.), Progress in Pattern Recognition, Speech and Image Analysis. Proceedings, 2003. XVII, 693 pages. 2003.

Vol. 2904: T. Johansson, S. Maitra (Eds.), Progress in Cryptology – INDOCRYPT 2003. Proceedings, 2003. XI, 431 pages. 2003.

Vol. 2905: A. Sanfeliu, J. Ruiz-Shulcloper (Eds.), Progress in Pattern Recognition, Speech and Image Analysis. Proceedings, 2003. XVII, 693 pages. 2003.

Vol. 2906: T. Ibaraki, N. Katoh, H. Ono (Eds.), Algorithms and Computation. Proceedings, 2003. XVII, 748 pages. 2003.

Vol. 2908: K. Chae, M. Yung (Eds.), Information Security Applications. Proceedings, 2003. XII, 506 pages. 2004.

Vol. 2910: M.E. Orlowska, S. Weerawarana, M.P. Papazoglou, J. Yang (Eds.), Service-Oriented Computing – ICSOC 2003. Proceedings, 2003. XIV, 576 pages. 2003.

Vol. 2911: T.M.T. Sembok, H.B. Zaman, H. Chen, S.R. Urs, S.H.Myaeng (Eds.), Digital Libraries: Technology and Management of Indigenous Knowledge for Global Access. Proceedings, 2003. XX, 703 pages. 2003.

Vol. 2913: T.M. Pinkston, V.K. Prasanna (Eds.), High Performance Computing – HiPC 2003. Proceedings, 2003. XX, 512 pages. 2003.

Vol. 2914: P.K. Pandya, J. Radhakrishnan (Eds.), FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science. Proceedings, 2003. XIII, 446 pages. 2003.

Vol. 2916: C. Palamidessi (Ed.), Logic Programming. Proceedings, 2003. XII, 520 pages. 2003.

Vol. 2918: S.R. Das, S.K. Das (Eds.), Distributed Computing – IWDC 2003. Proceedings, 2003. XIV, 394 pages. 2003.

Vol. 2920: H. Karl, A. Willig, A. Wolisz (Eds.), Wireless Sensor Networks. Proceedings, 2004. XIV, 365 pages. 2004.

Vol. 2922: F. Dignum (Ed.), Advances in Agent Communication. Proceedings, 2003. X, 403 pages. 2004. (Subseries LNAI).

Vol. 2923: V. Lifschitz, I. Niemelä (Eds.), Logic Programming and Nonmonotonic Reasoning. Proceedings, 2004. IX, 365 pages. 2004. (Subseries LNAI).

Vol. 2927: D. Hales, B. Edmonds, E. Norling, J. Rouchier (Eds.), Multi-Agent-Based Simulation III. Proceedings, 2003. X, 209 pages. 2003. (Subseries LNAI).

Vol. 2928: R. Battiti, M. Conti, R. Lo Cigno (Eds.), Wireless On-Demand Network Systems. Proceedings, 2004. XIV, 402 pages. 2004.

Vol. 2929: H. de Swart, E. Orlowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments. Proceedings. VII, 273 pages. 2003.

Vol. 2932: P. Van Emde Boas, J. Pokorný, M. Bieliková, J. Štuller (Eds.), SOFSEM 2004: Theory and Practice of Computer Science. Proceedings, 2004. XIII, 385 pages. 2004.

Vol. 2935: P. Giorgini, J.P. Müller, J. Odell (Eds.), Agent-Oriented Software Engineering IV. Proceedings, 2003. X, 247 pages. 2004.

Vol. 2937: B. Steffen, G. Levi (Eds.), Verification, Model Checking, and Abstract Interpretation. Proceedings, 2004. XI, 325 pages. 2004.

Vol. 2950: N. Jonoska, G. Păun, G. Rozenberg (Eds.), Aspects of Molecular Computing. XI, 391 pages. 2004.

Table of Contents

Modeling Agents and Multiagent Systems

Methodologies and Tools

Patterns, Architectures, and Reuse

Roles and Organizations

Using UML in the Context of Agent-Oriented Software Engineering: State of the Art

Bernhard Bauer[1] and Jörg P. Müller[2]

[1] Institute of Computer Science, University of Augsburg, D-86135 Augsburg, Germany
Bernhard.Bauer@informatik.uni-augsburg.de
[2] Siemens AG, Corporate Technology, CT IC 6, D-81730 Munich, Germany
joerg.p.mueller@siemens.com

Abstract. Most of the methodologies and notations for agent-oriented software engineering developed over the past few years are based on the Unified Modeling Language (UML) or proposed extensions of UML. However, at the moment an overview on the different approaches is missing. In this paper. we present a state-of-the-art survey of the different methodologies and notations that, in one way or the other, rely on the usage of UML for the specification of agent-based systems. We focus on two aspects, i.e., design methodologies for agent-oriented software engineering, and different types of notations (e.g., for interaction protocols, social structures, or ontologies) that rely on UML. [1]

1 Introduction

The complexity of commercial software development processes increasingly requires the usage of software engineering techniques, including methodologies and tools for building, deploying, and maintaining software systems and solutions. In this context, software methodologies play a key role. A *software methodology* is typically characterized by a *modeling language* – used for the description of models, defining the elements of the model together with a specific syntax (notation) and associated semantics – and a *software process* – defining the development activities, the interrelationships among the activities, and how the different activities are performed. In particular, the software process defines phases for process and project management as well as quality assurance. The three key phases that one is likely to find in any software engineering process are that of analysis, design and implementation. In a strict waterfall model these are the only phases; more recent software development process models employ a "round trip engineering" approach, i.e., provide an iteration of smaller granularity cycles, in which models developed in earlier phases can be refined and adapted in later phases.

Agent technology enables the realization of complex software systems characterized by situation awareness and intelligent behavior, a high degree of distribution, as well as mobility support. Over the past year, agents have been very successful from the scientific point of view; also, the beginning commercial success of agent technology at the application level (in the sense of: intelligent components

[1] This paper is a short and adapted version of [42]

P. Giorgini, J.P. Müller, J. Odell (Eds.): AOSE 2003, LNCS 2935, pp. 1–24, 2004.

supporting intelligent applications, see e.g., [44]) is evident today. However, the potential role of agent technology as a new paradigm for software engineering has not yet met with broad acceptance in industrial and commercial settings. We claim that the main reason for this is the lack of accepted methods for software development depending on widely standardized representations of artifacts supporting all phases of the software lifecycle. In particular, these standardized representations are needed by tool developers to provide commercial quality tools that mainstream software engineering departments need for industrial agent systems development.

Currently, most industrial methodologies are based on the Object Management Group's (OMG) Unified Modeling Language (UML) accompanied by process frameworks such as the Rational Unified Process (RUP), see [28] for details. The Model-Driven Architecture (MDA [40]) from the OMG allows a cascade if code generations from high-level models (platform independent model) via platform dependent models to directly executable code (e.g., see the tool offered by Kennedy Carter [39]).

Thus, one possibility to provide an answer regarding the state-of-the-art in agent-oriented software engineering is to look at the level of support currently provided for UML technologies by recent agent-based engineering approaches. In this paper we will provide a detailed survey of methodologies and notations for agent-based engineering of software systems based on UML.

In Section 2 we will have a closer look at different methodologies for designing agent-based systems. In Section 3 focuses on notations based on UML. In particular, we shall look at notations for interaction protocols, social structures, agent classes, ontologies, and goals and plans. The paper concludes with a summary and an outlook for further research in Section 4.

2 Methodologies

In this we will take a closer look at agent methodologies that directly extend object-oriented – UML approaches. In the next section we will also give an overview of UML notations and extensions available for the specification of agent-based systems. Since most of the notations use graphical representations of software artifacts we will use examples taken from the original research papers.

2.1 Agent Modeling Techniques for Systems of BDI Agents

One of the first methodologies for the development of BDI agents based on OO technologies was presented in [2][3][4][5]. The agent methodology distinguishes between the *external viewpoint* - the system is decomposed into agents, modeled as complex objects characterized by their purpose, their responsibilities, the services they perform, the information they require and maintain, and their external interactions - and the *internal viewpoint* - the elements required by a particular agent architecture must be modeled for each agent, i.e. an agent's beliefs, goals, and plans. For each of these views different models are described (based on [2] and [5]):

The *external view* is characterized by two models which are largely independent of the underlying BDI architecture:

Agent Model: This model describes the hierarchical relationship among different abstract and concrete agent classes (*Agent Class Model*) similar to a UML class diagram denoting both abstract and concrete (instantiable) agent classes, inheritance and aggregation as well as predefined reserved attributes, e.g., each class may have associated belief, goal, and plan models; and identifies the agent instances which may exist within the system, their multiplicity, and when they come into existence (*Agent Instance Model*) with the possibility to define *initial-belief-state* and *initial-goal-state* attributes.

Interaction Model: describes the responsibilities of an agent class, the services it provides, associated interactions, and control relationships between agent classes. This includes the syntax and semantics of messages used for inter-agent communication and communication between agents and other system components, such as user interfaces.

BDI agents are *internally viewed* as having certain mental attitudes, *Beliefs*, *Desires* and *Intentions*, which represent, respectively, their informational, motivational and deliberative states. These aspects are captured, for each agent class, by the following models.

Belief Model describes the information about the environment and internal state that an agent of that class may hold, and the actions it may perform. The possible beliefs of an agent and their properties, such as whether or not they may change over time, are described by a *belief set*. In addition, one or more *belief states* - particular instances of the belief set - may be defined and used to specify an agent's initial mental state. The *belief set* is specified by a set of object diagrams which define the domain of the beliefs of an agent class. A belief state is a set of instance diagrams which define a particular instance of the belief set. Formally, defined by a set of typed predicates whose arguments are terms over a universe of predefined and user-defined function symbols.

Goal Model describes the goals that an agent may possibly adopt, and the events to which it can respond. It consists of a *goal set* which specifies the goal and event domain and one or more *goal states* - sets of ground goals - used to specify an agent's initial mental state. A goal set is, formally, a set of goal formula signatures. Each such formula consists of a modal goal operator applied to a predicate from the belief set.

Plan Model describes the plans that an agent may possibly employ to achieve its goals. It consists of a *plan set* which describes the properties and control structure of *individual plans*. Plans are modeled similar to simple UML State Chart Diagrams, which can be directly executed showing how an agent should behave to achieve a goal or respond to an event. In contrast to UML activities may be sub-goals, denoted by formulae from the agent's goal set; conditions are predicates from the agent's belief set; actions include those defined in the belief set, and built-in actions. The latter include assert and retract, which update the belief state of the agent.

2.2 Message

MESSAGE (Methodology for Engineering Systems of Software Agents) [6][7] is a methodology which builds upon best practice methods in current software engineering such as for instance UML for the analysis and design of agent-based systems. It consists of (i) applicability guidelines; (ii) a modeling notation that

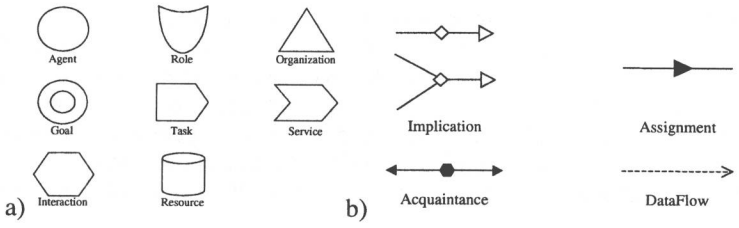


Fig. 1. a) concept symbols in MESSAGE; b) relations in MESSAGE

extends UML by agent-related concepts (inspired e.g. by Gaia); and (iii) a process for analysis and design of agent systems based on Rational unified Process. The MESSAGE modeling notation extends UML notation by key agent-related concepts. We describe the notation used in MESSAGE based on the example presented in [7]. For details on the example we refer to this paper. The used concept and relation symbols are shown in Fig. 1.

The main focus of MESSAGE is on the phase of analysis of agent-based systems. For this purpose, MESSAGE presents five analysis models, which analysts can use to capture different aspects of an agent-based system. The models are described in terms of sets of interrelated concepts. The five models are (following [7][6]):

Organization Model: The Organization Model captures the overall structure and the behavior of a group of agents and the external organization working together to reach common goals. In particular, it represents the responsibilities and authorities with respect to entities such as processes, information, and resources and the structure of the organization in terms of sub-organization such as departments, divisions, sections, etc. expressed through power relationships (e.g. superior-subordinate relationships). Moreover it provides the *social view* characterizing the overall behavior of the group, whereas the agent model covers the *individual view* dealing with the behavior of agents to achieve common/social goals. It offers software designers a useful abstraction for understanding the overall structure of the multi-agent system, what the agents are, what resources are involved, what the role of each agent is, what their responsibilities are, which tasks are achieved individually and which achieved through co-operation. Different types of organization diagrams are available in MESSAGE to support the graphical representation of social concepts (see Fig. 2).

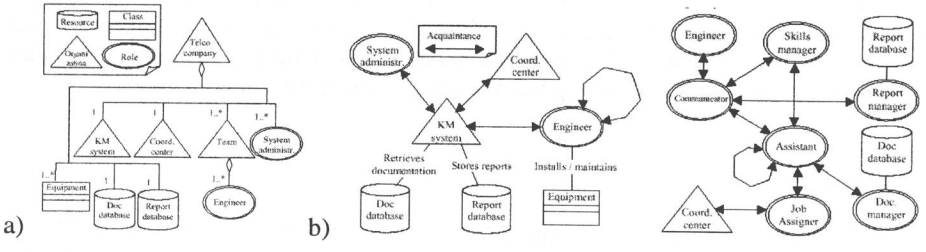


Fig. 2. Examples of organization diagrams: a) structural relationships, b) acquaintance relationships (analysis phase 0 and 1)