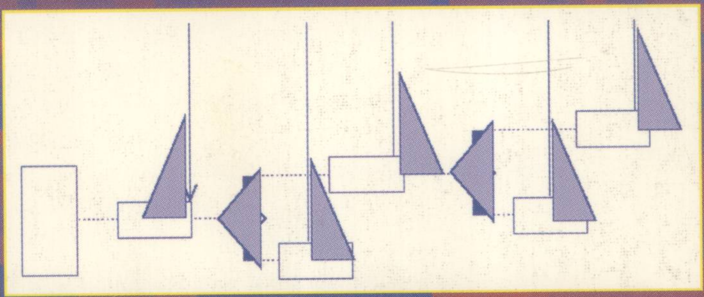
State-of-the-Art Survey

LNCS 3382

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

Agent-Oriented Software Engineering V

5th International Workshop, AOSE 2004
New York, NY, USA, July 2004
Revised Selected Papers



Springer

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

Agent-Oriented Software Engineering V

5th International Workshop, AOSE 2004
New York, NY, USA, July 19, 2004
Revised Selected Papers

Springer

Volume Editors

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

Paolo Giorgini
University of Trento, Department of Information and Communication Technology
Via Somamrive, 14, 38050 Provo, 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

Library of Congress Control Number: 2004117070

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

ISSN 0302-9743
ISBN 3-540-24286-4 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springeronline.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 11375104 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 naïve 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 which 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 modelling. 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.

As its very successful predecessors, AOSE 2000, AOSE 2001, AOSE 2002, and AOSE 2003 (Lecture Notes in Computer Science, Volumes 1957, 2222, 2585, and 2935), the AOSE 2004 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 2004 was hosted by the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004) held in New York, USA on July 2004. The workshop received 57 submissions, and 15 of them were accepted for presentation (which is an acceptance rate of 26%). These papers were reviewed by at least two members of an international program committee composed of 29 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 researchers, students and practitioners, who contributed to the discussion of research problems related to the main topics in AOSE.

This volume contains revised and improved versions of the 15 papers presented at the workshop, organized in three sections: *Modeling*, *Design*, and *Reuse and Platforms*. We believe that this thoroughly prepared volume is of particular value to all readers interested in key topics and the 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 2004 a high-quality scientific event.

November 2004

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.p.mueller@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)
Paolo Ciancarini (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)
David Kinny (Australia)
Manuel Kolp (Belgium)
Yannis Labrou (USA)
Juergen Lind (Germany)
Haralambos Mouratidis (UK)
Matthias Nickles (Germany)
Andrea Omicini (Italy)
Van Parunak (USA)
Juan Pavon (Spain)
Anna Perini (Italy)
Marco Pistore (Italy)
Onn Shehory (Israel)
Paola Turci (Italy)
Gerd Wagner (Germany)
Gerhard Weiss (Germany)
Mike Wooldridge (UK)
Eric Yu (Canada)
Franco Zambonelli (Italy)

Auxiliary Reviewers: Paolo Busetta, Giancarlo Guizzardi, Savas Konur, Viara Popova, Michael Rovatsos, Alexei Sharpanskykh, Arnon Sturm, Angelo Susi, Vera Werneck

Lecture Notes in Computer Science 3382

Lecture Notes in Computer Science

For information about Vols. 1–3247

please contact your bookseller or Springer

Vol. 3382: J. Odell, P. Giorgini, J.P. Müller (Eds.), Agent-Oriented Software Engineering V. X, 239 pages. 2004.

Vol. 3358: J. Cao, L.T. Yang, M. Guo, F. Lau (Eds.), Parallel and Distributed Processing and Applications. XXIV, 1058 pages. 2004.

Vol. 3356: G. Das, V.P. Gulati (Eds.), Intelligent Information Technology. XII, 428 pages. 2004.

Vol. 3348: A. Canteaut, K. Viswanathan (Eds.), Progress in Cryptology - INDOCRYPT 2004. XIV, 431 pages. 2004.

Vol. 3347: R.K. Ghosh, H. Mohanty (Eds.), Distributed Computing and Internet Technology. XX, 472 pages. 2004.

Vol. 3341: R. Fleischer, G. Trippen (Eds.), Algorithms and Computation. XVII, 935 pages. 2004.

Vol. 3340: C.S. Calude, E. Calude, M.J. Dinneen (Eds.), Developments in Language Theory. XI, 431 pages. 2004.

Vol. 3339: G.I. Webb, X. Yu (Eds.), AI 2004: Advances in Artificial Intelligence. XXII, 1272 pages. 2004. (Subseries LNAI).

Vol. 3338: S.Z. Li, J. Lai, T. Tan, G. Feng, Y. Wang (Eds.), Advances in Biometric Person Authentication. XVIII, 699 pages. 2004.

Vol. 3337: J.M. Barreiro, F. Martin-Sanchez, V. Maojo, F. Sanz (Eds.), Biological and Medical Data Analysis. XI, 508 pages. 2004.

Vol. 3336: D. Karagiannis, U. Reimer (Eds.), Practical Aspects of Knowledge Management. X, 523 pages. 2004. (Subseries LNAI).

Vol. 3334: Z. Chen, H. Chen, Q. Miao, Y. Fu, E. Fox, E.-p. Lim (Eds.), Digital Libraries: International Collaboration and Cross-Fertilization. XX, 690 pages. 2004.

Vol. 3333: K. Aizawa, Y. Nakamura, S. Satoh (Eds.), Advances in Multimedia Information Processing - PCM 2004. XXXV, 785 pages. 2004.

Vol. 3332: K. Aizawa, Y. Nakamura, S. Satoh (Eds.), Advances in Multimedia Information Processing - PCM 2004. XXXVI, 1051 pages. 2004.

Vol. 3331: K. Aizawa, Y. Nakamura, S. Satoh (Eds.), Advances in Multimedia Information Processing - PCM 2004. XXXVI, 667 pages. 2004.

Vol. 3329: P.J. Lee (Ed.), Advances in Cryptology - ASIACRYPT 2004. XVI, 546 pages. 2004.

Vol. 3328: K. Lodaya, M. Mahajan (Eds.), FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science. XVI, 532 pages. 2004.

Vol. 3326: A. Sen, N. Das, S.K. Das, B.P. Sinha (Eds.), Distributed Computing - IWDC 2004. XIX, 546 pages. 2004.

Vol. 3323: G. Antoniou, H. Boley (Eds.), Rules and Rule Markup Languages for the Semantic Web. X, 215 pages. 2004.

Vol. 3322: R. Klette, J. Žunić (Eds.), Combinatorial Image Analysis. XII, 760 pages. 2004.

Vol. 3321: M.J. Maher (Ed.), Advances in Computer Science - ASIAN 2004. XII, 510 pages. 2004.

Vol. 3320: K.-M. Liew, H. Shen, S. See, W. Cai (Eds.), Parallel and Distributed Computing: Applications and Technologies. XXIV, 891 pages. 2004.

Vol. 3316: N.R. Pal, N.K. Kasabov, R.K. Mudi, S. Pal, S.K. Parui (Eds.), Neural Information Processing. XXX, 1368 pages. 2004.

Vol. 3315: C. Lemaître, C.A. Reyes, J.A. González (Eds.), Advances in Artificial Intelligence – IBERAMIA 2004. XX, 987 pages. 2004. (Subseries LNAI).

Vol. 3314: J. Zhang, J.-H. He, Y. Fu (Eds.), Computational and Information Science. XXIV, 1259 pages. 2004.

Vol. 3312: A.J. Hu, A.K. Martin (Eds.), Formal Methods in Computer-Aided Design. XI, 445 pages. 2004.

Vol. 3311: V. Roca, F. Rousseau (Eds.), Interactive Multimedia and Next Generation Networks. XIII, 287 pages. 2004.

Vol. 3309: C.-H. Chi, K.-Y. Lam (Eds.), Content Computing. XII, 510 pages. 2004.

Vol. 3308: J. Davies, W. Schulte, M. Barnett (Eds.), Formal Methods and Software Engineering. XIII, 500 pages. 2004.

Vol. 3307: C. Bussler, S.-k. Hong, W. Jun, R. Kaschek, D.. Kinshuk, S. Krishnaswamy, S.W. Loke, D. Oberle, D. Richards, A. Sharma, Y. Sure, B. Thalheim (Eds.), Web Information Systems – WISE 2004 Workshops. XV, 277 pages. 2004.

Vol. 3306: X. Zhou, S. Su, M.P. Papazoglou, M.E. Orlowska, K.G. Jeffery (Eds.), Web Information Systems – WISE 2004. XVII, 745 pages. 2004.

Vol. 3305: P.M.A. Sloot, B. Chopard, A.G. Hoekstra (Eds.), Cellular Automata. XV, 883 pages. 2004.

Vol. 3303: J.A. López, E. Benfenati, W. Dubitzky (Eds.), Knowledge Exploration in Life Science Informatics. X, 249 pages. 2004. (Subseries LNAI).

Vol. 3302: W.-N. Chin (Ed.), Programming Languages and Systems. XIII, 453 pages. 2004.

Vol. 3299: F. Wang (Ed.), Automated Technology for Verification and Analysis. XII, 506 pages. 2004.

Vol. 3298: S.A. McIlraith, D. Plexousakis, F. van Harmelen (Eds.), The Semantic Web – ISWC 2004. XXI, 841 pages. 2004.

Vol. 3296: L. Bougé, V.K. Prasanna (Eds.), High Performance Computing - HiPC 2004. XXV, 530 pages. 2004.

Vol. 3295: P. Markopoulos, B. Eggen, E. Aarts, J.L. Crowley (Eds.), Ambient Intelligence. XIII, 388 pages. 2004.

Vol. 3294: C.N. Dean, R.T. Boute (Eds.), Teaching Formal Methods. X, 249 pages. 2004.

Vol. 3293: C.-H. Chi, M. van Steen, C. Wills (Eds.), Web Content Caching and Distribution. IX, 283 pages. 2004.

Vol. 3292: R. Meersman, Z. Tari, A. Corsaro (Eds.), On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops. XXIII, 885 pages. 2004.

Vol. 3291: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. XXV, 824 pages. 2004.

Vol. 3290: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE. XXV, 823 pages. 2004.

Vol. 3289: S. Wang, K. Tanaka, S. Zhou, T.W. Ling, J. Guan, D. Yang, F. Grandi, E. Mangina, I.-Y. Song, H.C. Mayr (Eds.), Conceptual Modeling for Advanced Application Domains. XXII, 692 pages. 2004.

Vol. 3288: P. Atzeni, W. Chu, H. Lu, S. Zhou, T.W. Ling (Eds.), Conceptual Modeling – ER 2004. XXI, 869 pages. 2004.

Vol. 3287: A. Sanfeliu, J.F. Martínez Trinidad, J.A. Carrasco Ochoa (Eds.), Progress in Pattern Recognition, Image Analysis and Applications. XVII, 703 pages. 2004.

Vol. 3286: G. Karsai, E. Visser (Eds.), Generative Programming and Component Engineering. XIII, 491 pages. 2004.

Vol. 3285: S. Manandhar, J. Austin, U.B. Desai, Y. Oyanagi, A. Talukder (Eds.), Applied Computing. XII, 334 pages. 2004.

Vol. 3284: A. Karmouch, L. Korba, E.R.M. Madeira (Eds.), Mobility Aware Technologies and Applications. XII, 382 pages. 2004.

Vol. 3283: F.A. Aagesen, C. Anutariya, V. Wuwongse (Eds.), Intelligence in Communication Systems. XIII, 327 pages. 2004.

Vol. 3282: V. Guruswami, List Decoding of Error-Correcting Codes. XIX, 350 pages. 2004.

Vol. 3281: T. Dingsøyr (Ed.), Software Process Improvement. X, 207 pages. 2004.

Vol. 3280: C. Aykanat, T. Dayar, İ. Körpeoğlu (Eds.), Computer and Information Sciences - ISCIS 2004. XVIII, 1009 pages. 2004.

Vol. 3278: A. Sahai, F. Wu (Eds.), Utility Computing. XI, 272 pages. 2004.

Vol. 3275: P. Perner (Ed.), Advances in Data Mining. VIII, 173 pages. 2004. (Subseries LNAI).

Vol. 3274: R. Guerraoui (Ed.), Distributed Computing. XIII, 465 pages. 2004.

Vol. 3273: T. Baar, A. Strohmeier, A. Moreira, S.J. Mellor (Eds.), <<UML>> 2004 - The Unified Modelling Language. XIII, 454 pages. 2004.

Vol. 3271: J. Vicente, D. Hutchison (Eds.), Management of Multimedia Networks and Services. XIII, 335 pages. 2004.

Vol. 3270: M. Jeckle, R. Kowalczyk, P. Braun (Eds.), Grid Services Engineering and Management. X, 165 pages. 2004.

Vol. 3269: J. Lopez, S. Qing, E. Okamoto (Eds.), Information and Communications Security. XI, 564 pages. 2004.

Vol. 3268: W. Lindner, M. Mesiti, C. Türker, Y. Tzitzikas, A. Vakali (Eds.), Current Trends in Database Technology - EDBT 2004 Workshops. XVIII, 608 pages. 2004.

Vol. 3267: C. Priami, P. Quaglia (Eds.), Global Computing. VIII, 377 pages. 2004.

Vol. 3266: J. Solé-Pareta, M. Smirnov, P.V. Mieghem, J. Domingo-Pascual, E. Monteiro, P. Reichl, B. Stiller, R.J. Gibbens (Eds.), Quality of Service in the Emerging Networking Panorama. XVI, 390 pages. 2004.

Vol. 3265: R.E. Frederking, K.B. Taylor (Eds.), Machine Translation: From Real Users to Research. XI, 392 pages. 2004. (Subseries LNAI).

Vol. 3264: G. Paliouras, Y. Sakakibara (Eds.), Grammatical Inference: Algorithms and Applications. XI, 291 pages. 2004. (Subseries LNAI).

Vol. 3263: M. Weske, P. Liggesmeyer (Eds.), Object-Oriented and Internet-Based Technologies. XII, 239 pages. 2004.

Vol. 3262: M.M. Freire, P. Chemouil, P. Lorenz, A. Gravey (Eds.), Universal Multiservice Networks. XIII, 556 pages. 2004.

Vol. 3261: T. Yakhno (Ed.), Advances in Information Systems. XIV, 617 pages. 2004.

Vol. 3260: I.G.M.M. Niemegeers, S.H. de Groot (Eds.), Personal Wireless Communications. XIV, 478 pages. 2004.

Vol. 3259: J. Dix, J. Leite (Eds.), Computational Logic in Multi-Agent Systems. XII, 251 pages. 2004. (Subseries LNAI).

Vol. 3258: M. Wallace (Ed.), Principles and Practice of Constraint Programming – CP 2004. XVII, 822 pages. 2004.

Vol. 3257: E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Eds.), Engineering Knowledge in the Age of the Semantic Web. XVII, 517 pages. 2004. (Subseries LNAI).

Vol. 3256: H. Ehrig, G. Engels, F. Parisi-Presicce, G. Rozenberg (Eds.), Graph Transformations. XII, 451 pages. 2004.

Vol. 3255: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), Advances in Databases and Information Systems. XI, 423 pages. 2004.

Vol. 3254: E. Macii, V. Paliouras, O. Koufopavlou (Eds.), Integrated Circuit and System Design. XVI, 910 pages. 2004.

Vol. 3253: Y. Lakhnech, S. Yovine (Eds.), Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. X, 397 pages. 2004.

Vol. 3252: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004 Workshops. XVIII, 785 pages. 2004.

Vol. 3251: H. Jin, Y. Pan, N. Xiao, J. Sun (Eds.), Grid and Cooperative Computing - GCC 2004. XXII, 1025 pages. 2004.

Vol. 3250: L.-J. (LJ) Zhang, M. Jeckle (Eds.), Web Services. X, 301 pages. 2004.

Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), Artificial Intelligence and Symbolic Computation. X, 285 pages. 2004. (Subseries LNAI).

Table of Contents

Modeling

Design

Reuse and Platforms

Organizational and Social Concepts in Agent Oriented Software Engineering

Xinjun Mao[1] and Eric Yu[2]

[1] Department of Computer Science, National University of Defense Technology, China
xjmao21@21cn.com
[2] Faculty of Information Studies, University of Toronto, Canada
eric.yu@utoronto.ca

Abstract. AOSE methodologies and models borrow abstractions and concepts from organizational and social disciplines. Although they all view multi-agent systems as organized society, the organizational abstractions, assumptions, concepts, and models in them are actually used in different ways. It is therefore desirable to have a systematic way of analyzing and comparing the organizational and social concepts in AOSE. The contribution of this paper is threefold. Firstly, we identify some premises behind the social conceptions adopted in multi-agent systems. Secondly, we define levels of modeling constructs and classify organizational and social concepts in the AOSE literature into categories according to their organizational abstractions. Finally, we analyze two representative AOSE methodologies and their models, explaining how they use organizational and social concepts to analyze and specify multi-agent system, reflecting various social premises at different levels.

1 Introduction

Multi-agent systems (MAS) are rapidly emerging as a powerful paradigm for developing complex system. However, if we want the paradigm to be successfully applied in the development of complex system, the models, technologies and even the methodologies should be developed to support the developers to engineer such systems in a robust, reliable, and repeatable fashion.

MAS research often draws on concepts from other disciplines such as psychology, economic, cognitive science, linguistics, artificial intelligence, etc. For example, we often analyze interaction protocols and communication actions among agents based on the speech acts theory, which comes from philosophy and linguistics. The abstraction of the intentional stance has been borrowed from cognitive science to reason about and analyze the autonomous behaviors of agents. Recently, many methodologies and models borrowing the abstractions and concepts from the organization and sociology disciplines have been put forward for modeling, analyzing and designing MAS. Although these methodologies all view multi-agent systems as organized society in a broad sense, the organization abstractions, concepts, assumptions and models that they adopt are actually varied. The proposed methodologies may vary in the stages of

[1] This research was conducted while the first author was visiting the University of Toronto.

J. Odell et al. (Eds.): AOSE 2004, LNCS 3382, pp. 1 – 15, 2005.

software engineering life cycle that they support, thus adopting different assumptions about organizations, and different levels of abstraction in their models. In addition, some of the organizational and social concepts, while using different terminology, may have similar meaning and purpose. Conversely, a given term may have different interpretations and definitions in various models and methodologies.

Many papers have provided comparisons and evaluations of the methodologies in agent oriented software engineering (AOSE), such as [23, 24, 34, 26, 27, 28]. However, there are few efforts to compare them from the standpoint of organizational and social abstractions, especially to analyze the organizational and social concepts in AOSE literature. Since organizational and social abstractions are playing central roles in the design of AOSE methodologies and the development of multi-agent systems, it is important to have a map of the research on organizational and social concepts in AOSE. The rest of this paper is structured as follows. Section 2 analyzes the assumptions in social abstractions. Section 3 defines the modeling construct levels of MAS, identify and classify the organizational and social concepts in AOSE literature and explain in detail how they are used to specify and analyze the MAS. Section 4 analyzes a number of AOSE methodologies that are influential in AOSE. Finally, conclusions and future work are discussed in section 5.

2 Simplifying Assumptions in Social Abstractions

In adopting concepts from the social and organizational sciences, AOSE methodologies are not attempting to capture the full richness of human social phenomena. The borrowed concepts are selected abstractions that are considered to be useful for the purpose of conceiving and designing multi-agent software systems. Thus, each methodology selects a set of concepts and modeling constructs appropriate for its intended purposes, and possibly for a specialized application area or context. In doing so, a methodology incorporates assumptions and premises about organizations and societies, either implicitly or explicitly. Typically, these are simplifying assumptions which reduce the complexity of social phenomena. Despite simplifications and restrictions, through these social and organizational concepts, AOSE methodologies offer higher level abstractions than conventional software engineering paradigms. Thus agent orientation can be seen as the latest step in the progression towards better modeling abstractions that are closer to the real world, shortening the conceptual distance between the full richness of the application domain and the models offered by the software methodology to describe the world.

In analyzing a variety of AOSE methodologies, we note that their premises may vary with regard to at least the following characteristics.

- **Open or Closed.** A system is open if it has no definite boundary, thus allowing new, possibly unknown agents to enter or leave from time to time in the life cycle of the system. Therefore, the collection of entities (e.g., agents) in an open system may change and cannot be completely defined at design time. For instance, the Internet is such an open system. For closed systems, the population of system elements does not vary at run time. Therefore, they can be defined at design time by the software developers. Clearly, open systems present design challenges that are not found in closed ones.

• **Dynamic or Static.** A system is dynamic if the system elements, especially the abilities of agents in the system and the services they provide and/or the inter-agent relationships, can change at run-time. For example, the roles that an agent plays may vary in different contexts and situations, and therefore the inter-agent relationships (e.g., the interactions and/or dependencies) may also change. For a static system, all of the system elements are invariable. Typically, dynamic systems are more complicated and more difficult to develop than static ones.

• **Cooperative or Self-Interested.** The agents in some system may be cooperative in certain social context. They share some common goals and interact with each other in a cooperatively way to willingly provide resources and services. Conversely, the self-interested agent does for itself, and may refuse to provide services or resources for other agents. In addition, conflicts are more likely to occur between self-interested agents, especially when scarce resources need to be shared.

• **Hierarchic.** Many systems are hierarchic, i.e., composed of interrelated sub-systems, each of which is in turn hierarchic in structure, until the lowest level of elementary sub-system is reached [9]. There can be various relationships among the sub-systems. In contrast, the hierarchic systems evolve more quickly than non-hierarchic ones of comparable size, which make them more difficult to deal with [9]. Hierarchic structures are used extensively in software engineering to reduce system complexity. However, many social structures are not hierarchic.

• **Global Constraint.** In some systems, there are global constraints that are respected by all agents in the system and thus govern the relationships and interactions among them. For example, a social law constrains the behavior of agents in the organization. The explicit identification of the global constraints is of particular importance in the context of open system with self-interested agents. Such constraints can simplify system analysis and design.

3 Analyzing Organizational and Social Concepts in AOSE

In this section, we identify several levels of modeling constructs that are used in modeling MAS. We then classify the organizational and social concepts found in the AOSE literature, explaining how they can be used to model MAS.

3.1 Modeling Construct Levels

For the purpose of analysis, we organize the modeling constructs into a number of levels.

• **Single Agent.** In this level, the autonomous behaviors of agents are specified and analyzed in an abstract way. Generally, the functionalities and activities of agents are the most important aspects that should be modeled. For example, what are the functionalities of agents? what the resources and/or activities should they have in order to accomplish their functionalities? etc. The models describing the single agent are important constituents of the system requirement specification to guide the design of software agents.

• **Two Agents.** Agents in MAS are not isolated from one to another. Two agents may have various relationships between them like structural ones and behavioral ones. For example, one agent depends on another agent to get the resources required to accomplish its tasks, or should explicitly interacts with other agent by some interaction protocol (e.g., contract net) to acquire the resources or the assigned tasks; one agent may be the supervisor of another one and has the authority to assign the tasks to it. The information about the relationships between agents should be explicitly specified and analyzed in support of the requirement specification and analysis and further guide the software architecture design.

- **Two or More Agents Acting in a Coordinated Way.** In some MASs, two or more agents are organized together as a group and act in a coordinated way in order to achieve some common purposes. Agents in one group are often cooperative and have some common goals and joint behaviors. For such MASs, it is necessary in the analysis and design phase to identify and define the groups in the system, specify them in detail about the structural information (e.g., how agents in the group are organized) and the behavioral information (e.g., what the common goals of agents in the group) of them.
- **All Agents.** In this level, all agents in the system are treated as one organization, which should be specified and analyzed. For example, what is the organization structure of the system? Are there any global constraints in the organization that govern all agents in it?

3.2 Modeling Concepts

Now we turn to analyzing what the social premises mean in different modeling construct levels, what the organizational and social concepts are required to model MAS in these levels, and how they are used to specify and analyze the systems with various social premises. Although the organizational and social concepts are diverse in AOSE literature, a clear taxonomy of these concepts can be made according to their modeling purpose and the system construct level that they intend to deal with. In each category, the organizational and social concepts can be further divided into a number of groups. The concepts in each group often have similar semantics and modeling purpose (see Table 1).

Table 1. A taxonomy of organizational and social concepts in AOSE literature

Construct Levels	Organizational and Social Concepts
Single agent	*role, position, actor*
	responsibility, goal
	permission, right, resource
	activities, plan, task
Two agents	*dependency, interaction*
Two or more agents acting in a coordinated way	*group, group structure*
	common goal, joint intention(commitment)
All agents	*organization*
	organization rule, social law, interaction rule
	organization structure, organization pattern

3.2.1 Concepts for Modeling Single Agent

The organizational and social concepts in this level are used to specify and model the individuals (i.e., agent) in MAS and relatively in a low and micro abstraction level. In general, the functionalities, activities and resources of agents should be specified and analyzed independently of their concrete details.

In addition, according to the social premises described in section 2, agents in MAS may be dynamic or static, cooperative or self-interested. Dynamic agents may have different functionalities and activities in their life cycles. For self-interested agents, their functionalities, activities and resources may conflict with each other. Therefore, these social premises about agents also should be explicitly modeled and analyzed if the target system has these social properties.

- **Role, Position and Actor.** A *role* is an abstract characterization of the behaviors of agents within some specified context of organization. Generally, an agent can play multiple roles and a role can be played by a number of agents in MAS. Other concepts similar to role are *position* and *actor* used in *i** and Tropos. *Position* is a collection of roles that are occupied by one agent and *actor* is a generic concept to denote the intentional entity that may be an agent or role or position.

These concepts are important to abstractly model the agents in MAS, and helpful to manage the complexity of MAS without considering the concrete details of agents (e.g., implementation architectures and technologies). They present an effective way to naturally model the entities in the system. In general, the system's roles that agents play are specified in the role model like ones in Gaia, MaSE, etc. Therefore, the *role* concept, we can find, has been integrated into almost all of the AOSE methodologies based on the organizational and social abstractions.

The dynamic properties of agent can be viewed as that agent plays different roles in different context and situation, which will facilitate to model the dynamic MAS. However, we believe, the traditional role models like ones in MaSE, Gaia, etc., are unable to model such dynamic information. Therefore, other system model based on the role concept should be developed like one in [36] to show how agents dynamically enter or leave roles in different social situations.

- **Responsibility and Goal.** These concepts are used to specify and analyze the functionalities of a role. In Gaia, *responsibilities* are divided into two types; *liveness properties* and *safety properties*. *Liveness properties* describe those states of affairs that an agent must bring about given certain environment conditions. In contrast, *safety properties* correspond to the invariants in multi-agent system that agent must maintain. The *goal* of a *role* represents its strategic interests or intentions. In *i** and Tropos, two kinds of *goals* can be distinguished: *HardGoal* and *SoftGoal*. The latter denotes the goal that has no clear-cut definition or criteria for decision whether it is satisfied or not, and is typically used to specify the non-functional requirements.

Generally the functionalities of roles should be specified and analyzed in requirement phase in order to understand the behaviors of roles and guide the software design that implements the roles' functionalities. In contrast to the tasks, actions and plans of roles, the responsibilities or goals of roles are relatively high-level and stable, even in open and dynamic system, and therefore easy to elicit and specify. In addition, roles are typically goal-driven, therefore the goals or responsibilities of roles are related with their tasks, plans and interactions. The explicit identification and specification of the goals or responsibilities of roles will facilitate to elicit and model the tasks or plans that roles have, the resources and interactions that roles need, the rule it should obey in order to achieve its goals or responsibilities. Moreover, they are also helpful to analyze the potential goals conflict between the self-interested agents.

- **Permission, Right and Resource.** These concepts are used to specify and analyze what the roles require in order to realize their functionalities. *Permissions* in Gaia are the "rights" associated with a *role*. The *permission* of a role identifies the *resources* that are available to that role in order to realize its *responsibilities*. In the information system, the *permission* tends to be the information *resources* [8]. Other analogous concepts are *rights* in [3] and *resource* representing a physical or an informational unintentional entity in *i**, Tropos, and SODA.

Usually the resources are needed when agents intend to achieve their goals or responsibilities. In most cases, they are distributed in the environments that agents situate and may be dynamic. The resources in the environment are often limited and shared by a number of agents. To explicitly specify permission or resource of roles and model the environment that agents situate is significant to analyze how agent interacts with the environment, and the dependency between roles (e.g., some agents need resources while others produce resources). It is of particular importance to investigate the resource or "right" conflicts that may occur between the self-interested agents in dynamic system with limited resources.

- **Activity, Plan, Task.** These concepts are used to specify and analyze the behaviors that roles should have in order to accomplish their functionalities. The *activity* of a role in Gaia is actually the "private" action that may be carried out by the agent without interacting with other agents in order to realize its *responsibilities*. The *plan* concept in Tropos (analogous to the concept *task* in *i** framework) represents, in an abstract level, a way of doing something. The execution of the *plan* can be a means for satisfying a goal [16]. The *tasks* in SODA, however, can be classified as individual ones and social ones and expressed in term of the *responsibilities* they involve, of the competence they require, and of the *resources* they depend on. Typical, *social tasks* are those that require a number of different competences and the access to several different *resources*, whereas *individual tasks* are more likely to require well-delimited competence and limited *resources* [29].

These concepts describe in more detail the behaviors of roles and are necessary in the requirement analysis phase to show how to accomplish the roles' goals or responsibilities, and guide the software design that naturally encapsulate and implement these behaviors. Therefore most of the methodologies in AOSE support to model the role's activity, plan, or task to some extent.

3.2.2 Concepts for Modeling Two Agents

The organizational and social concepts in this level are used to model the relationships between individual agents. In general, the structural relationship and the behavioral relationship between two agents should be modeled when developing MAS. The relationships between agents may change for the dynamic system when the roles that agents play vary. Therefore, such dynamic relationships between agents also should be specified and analyzed if the target systems are dynamic.