Chapter 2
Agent-Oriented Software Engineering: Revisiting the State of the Art

Arnon Sturm and Onn Shehory

Abstract The field of Agent-Oriented Software Engineering (AOSE), which has evolved during the last two decades, attempts at introducing artificial intelligence concepts into the practice of software engineering. Despite considerable progress, it seems that the challenges the field encountered at its early days still hold. In particular, the adoption of AOSE principles in the academia, and even more so in the industry, is limited. This chapter aims to specify what AOSE is, to determine the related research areas, to examine historical perspectives of AOSE evolution, and to analyze its progress and point out challenges and future research directions.

Keywords Multi-agent systems • Agent-oriented software engineering • Survey • Adoption

1 Introduction

Agent-oriented software engineering (AOSE) entails the application of software engineering and artificial intelligence principles to the analysis, design, and implementation of software systems. Among the founders of the AOSE field are Nick Jennings and Mike Wooldridge who, via a series of seminal papers [1–5], introduced concepts and foundations for the domain. The intention at that time was to recruit Artificial Intelligence (AI) for the purpose of Software Engineering (SE), and in particular for building distributed complex systems.

A. Sturm (✉)
Department of Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel
e-mail: sturm@bgu.ac.il

O. Shehory
IBM—Haifa Research Lab, Haifa, Israel
e-mail: onn@il.ibm.com

O. Shehory and A. Sturm (eds.), *Agent-Oriented Software Engineering*,
DOI 10.1007/978-3-642-54432-3_2,

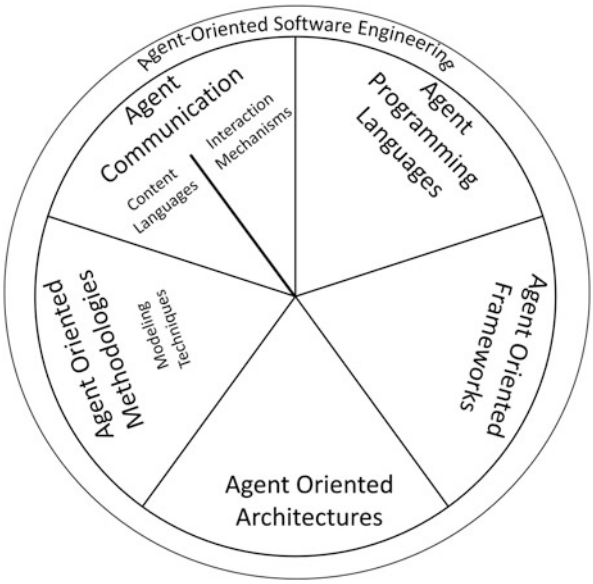

Fig. 2.1 Agent-oriented software engineering thematic map

Agent-based systems focus on dynamically interacting components. Each of these components has its own thread of control, and is engaged in complex coordination protocols. The emphasis is on the *interaction* of the components more than on the internals of the components themselves [1]. Agent-oriented software engineering is interpreted in two different ways. The first aims to support the development of agent-based system, whereas the second is more revolutionary and aims to support the development of complex systems where the notion of agent is being used. While it appears that the AOSE community made considerable progress towards the first, the second has not advanced much. This is stressed by Ricci and Santi [6] as well.

During the last two decades, the AOSE field has evolved to address many subjects. At present, the field includes the following high-level themes: methodologies, modeling techniques, framework implementations, agent-programming languages, and agent communication. The field's thematic map is depicted in Fig. 2.1.

This book provides a comprehensive view of the AOSE field and of the themes it comprises. This chapter merely aims to provide an overview of the research areas within the AOSE domain, focusing on progress and on challenges to be met. It also takes a historical perspective, examining the academic evolution of AOSE and pointing at future research. In the next section, we elaborate on the themes covered by the domain. Following, we take a historical viewpoint: we analyze surveys and overview papers discussing the AOSE domain to better understand its evolution. Finally, we conclude and point at voids and research challenges.

2 AOSE Themes

In this section, we elaborate on the aforementioned AOSE themes. This should facilitate the introduction of research questions and challenges the AOSE domain aims to address. We do not aim at providing a comprehensive list of topics, rather to reflect the type of research and development involved in the domain.

The AOSE paradigm is relatively new. As such, its establishment requires further considerations as we discuss below.

- A crisp definition of the agent notion and its most important properties is not yet fully agreed upon. There is much agreement on agent properties such as autonomy, dependability, and robustness; however, several other properties are not at consensus. Thus, the determination of the core set of agenthood properties is clearly a challenge.
- To promote AOSE adoption, there is a need to clarify how agent-based solutions facilitate complexity management in software engineering. The context and trade-offs of using the AOSE technology and paradigm require further deliberations.
- As a new discipline, the organizational implication of using AOSE should be carefully examined. In this respect, studies may take various viewpoints including technological, psychological, organizational, etc.
- Through its evolution, many alternative AOSE technologies were developed. To facilitate industrial adoption of AOSE, standardization is required. Indeed, standardization efforts of multi-agent systems are ongoing within working groups in FIPA, IEEE, and OMG. Nevertheless, these are progressing in low pace.
- There is a need to explore ways to integrate multi-agent and mainstream technologies to further increase agent technology usability and adoption.

2.1 Applications

There is an ongoing search for the types of applications for which agents are an adequate, valuable development approach. Below we list fields for which the agent-oriented paradigm was studied or examined, or is considered promising. We split the list into agent-oriented sub-list, which refers to the way in which agent technology is weaved in with other technologies and domain-oriented sub-list, which refers to application domains in which agents were found useful.

The agent-oriented fields consist of the following:

- Agent-based grid computing
- Agents and services—alignment of agents with service-oriented software development
- Agents for self-adaptive systems

- Integration of agent-oriented software into existing business processes and implications on business process reengineering
- Integration of agents with legacy systems
- Multi-agent-based simulation

The domain-oriented fields consist of the following:

- Self-organizing systems
- Social engineering
- Electronic commerce
- Tourism services
- Games

The chapter 3 of this book refers to applications and their impact on the agent technology.

2.2 Agent-Oriented Architectures

Multi-agent systems (MAS) can be considered as a new architecture style. Such architectures may offer diverse ways for developing MAS applications and frameworks. They may also suggest new ways for developing other types of large, complex software systems. The subfield of agent-oriented architecture studies, and likely needs to further explore, the following topics:

- New and adjusted software architectures for multi-agent systems
- The desired properties for such architectures, and the trade-offs among the architectures
- Reuse of agent-based systems design knowledge, patterns, and reference architectures

The chapter 4 of this book elaborates on issues related to software architectures of agents and MAS.

2.3 Agent Communication

An important aspect of agents is their ability to communicate. Communication can be exercised in many different ways. In the art, agent communication has been addressed from various viewpoints such as performance, scalability, reliability, security, and privacy. Various levels of abstraction also affect the focus of communication studies. As a result, multiple topics related to agent communication are found in research and practice, including the following:

- Agent Communication Languages (ACLs) including semantics and pragmatics
- Argumentation

- Commitments in communication
- Communication infrastructure
- Conversation
- Conversational agents
- Coordination and cooperation
- Dialogue games
- Human-agent communication
- Integration of interaction protocols within agents
- Interoperability
- Multiparty conversations
- Natural language processing application to communication
- Negotiation
- Ontologies and communication
- Reuse in communication

The chapter 6 further elaborates on agent communication.

2.4 Agent-Oriented Methodologies

Agent-oriented methodologies are a main research theme within the field of AOSE. It focuses on the development processes of multi-agent systems as well as the techniques to be applied in that context. It mainly covers the analysis and design stages within the development life cycle. Clearly, there is a need to further study other stages, including requirements, implementation, and testing. Topics related to agent-oriented methodologies are:

- Agent-oriented modeling techniques of various types (such as goal-oriented, BDI, etc.)
- Agents and model-driven approaches
- CASE tools to support agent-oriented software development in practice
- Evaluation and comparison of modeling techniques and methodologies

Part III of this book further elaborates on these issues.

2.5 Agent Programming Languages

Agent programming languages were an important step towards the adoption of the agent abstraction for software development [7]. One of the main ideas of agent programming languages is to introduce the notion of mentality (of agents) into the programming environment. Following this idea, research in the area addresses various topics including:

- Type of programming languages:
  - (Constraint) logic programming approaches to agent systems
  - Declarative approaches to engineering agent-based systems
  - Domain/purpose specific languages
- Applications of multi-agent programming languages including: legacy systems, pervasive applications, multi-robot systems, autonomous software (e.g., UAVs), (Semantic) Web and Grid-based applications, and deployed (industrial-strength) multi-agent systems
- Benchmarks and test-beds for comparing multi-agent programming languages and tools
- Evaluation of agent programming approaches including experiments and industrial experiences
- Verification of agent-based software
- Modal and epistemic logics for agent modeling
- Model checking agents and multi-agent systems

Part IV of the book further elaborates on these aspects.

2.6 Agent-Oriented Frameworks

Over the years, as agent technology evolved, there was a need to develop software tools and frameworks to accommodate the execution and development of such systems. Some software tools include frameworks to develop individual agents whereas others aim at MAS development. Many of these tools are open source in the sense that they are freely available to use and allow extensions. The majority of these frameworks was developed by the academia, and only a few were devised by the industry. This opens a new venue for applied research. Research questions addressed in this domain include the following:

- What should be the services included in agent-oriented frameworks?
- What are the required building blocks?
- How can such frameworks benefit from other evolving technologies?

The chapter 14 further elaborates on these agent-oriented frameworks, infrastructure, and platforms.

3 AOSE Surveys Analysis

Having explored major topics within the AOSE domain, in this section we review AOSE surveys performed over the years. From these, we extract (our subjective)

observations of the domain and its evolution and perform a chronological review to demonstrate the evolution of the field over time. Note that the observations we mention in this chapter are only those which are relevant to AOSE.

3.1 Agent-Oriented Software Engineering: The State of the Art (2001) [8]

In this study, the authors refer to AOSE as a means for managing software complexity, and examine the developed techniques. They focus on two issues: agent-oriented analysis and design and formal methods. In the context of agent-oriented analysis and design, they explore AOSE methodologies and have the following observations:

- AOSE methodologies can be classified as either Object-Oriented or Knowledge Engineering approaches. They suggest that the object-oriented approach dominates the area by either extending or adapting object-oriented methodologies.
- Following that observation they claim that the object-oriented approach is lacking in capturing most of the agent characteristics such as proactiveness and dynamic reaction. This led them to require further clarification regarding the relationship between objects and agents. In addition, they make an analogy to the object-oriented paradigm in which (according to an extreme approach) everything is an object and raise the question of whether everything should be an agent.

In the context of formal methods they assert the following challenges:

- Being used for specification, formal methods (such as BDI logic [9]) lack in their semantics. They assume that agents are perfect reasoners—which is not the case for real agents. Hence, their mapping of BDI elements into computational entities does not fully meet the capabilities of real agents.
- The survey refers to three implementation alternatives: (1) manual implementation through refinement; this is error-prone and a complex task; (2) a direct execution approach (such as Metatem [10]); this requires following a specific way of implementation; and (3) translation/compilation approach (such as in [11]), in which the code is automatically created from the specification language.
- The survey has identified two verification approaches: (1) the axiomatic approach in which one could take a specific program and systematically derive a logical theory (such an approach could be found in Hoare logic [12]); and (2) Model checking, which is more effective, yet hinders the problem of converting a program into a BDI logic.

In general, the issues that still need to be addressed by formal logic are complexity, decidability, and semantics.

In that survey, the authors also state challenges to be addressed. These include the following:

- The understanding of relationships to other paradigms
- The need for devising appropriate methodologies
- The need to address engineering for open systems
- The need to address engineering for scalability

3.2 A Manifesto for Agent Technology: Towards Next Generation Computing (2004) [13]

As part of their leadership of the AgentLink network of excellence [14], Luck et al. [13] published a comprehensive manifesto for agent technology. In that paper, they address various aspects of agents among which is the domain of AOSE. In the following, we summarize their observations at that time (2004) and their estimations and long-term vision of the field:

- Within the area of AOSE, the authors claim that the main focus in on analysis and design and on programming languages. They stress that the development of these methods is at the prototype stage and evaluations of these have been done in an ad hoc manner rather than in a systematic way.
- The authors assert that agent architectures focus on reactive and deliberative ones, and suggest that, at that time, agent-based solutions were tailor-made and not general ones based on generic architectures.
- When referring to communication language, the authors refer to KQML [15] and FIPA ACL [16] as the two main major languages. They suggest that, although both are well-developed, they do not address human and organization communication.
- Discussing applications, the authors mentioned many domains in which agents are being used. However, the use of AOSE within the development of these applications is neglected (both by the survey and the related publications).
- Referring to industrial interest, the authors report on many companies that employ agent-based research and development activities.
- The survey suggests several future AOSE challenges as follows:
  - Further develop AOSE methodologies
  - Provide developers with libraries of agent models
  - Improve tool support
  - Integrate existing and emerging technologies into agent development
  - Further develop the notion of open systems

3.3 Challenges and Research Directions in Agent-Oriented Software Engineering (2004) [17]

Zambonelli and Omicini [17] present AOSE as a new software engineering paradigm, which may contribute to various types of system engineering:

- First, they introduce AOSE as a novel approach to software engineering, which relies on the concept of an agent as its main new abstraction, encompassing properties such as autonomy, situatedness, and sociality. They consider the environment to be an important part of designing the system. They also make the distinction between objects and agents, where objects are statically connected and agents dynamically interact. They make the observation that other technologies such as components and active objects do progress towards agenthood.
- Second, they stress that agent properties address well-distributed systems' engineering needs.
- Third, they indicate the potential of using AOSE to promote artificial intelligence research from theory to practice.

Since AOSE is a new paradigm, Zambonelli and Omicini suggested that mainstream research should continue with the direction of agent modeling, MAS architectures, MAS methodologies, notation techniques, and MAS infrastructure, in order to increase the acceptance and the usability of AOSE. They divided the (other) research directions into three scales of observation: micro, macro, and meso, which differ in system size. In the following we summarize their observations:

- At the micro scale, there is a need
  - To assess the advantages of agents in software engineering
  - To explore nonstandard and extreme development processes
  - For novel modeling techniques (to address agent abstractions and needs)
  - For further exploration of formal methods
  - To further stress the interweaving of AI into AOSE
- At the macro scale, there is a need
  - To look for measurements for MAS in the macro scale, e.g., entropy [18] and coordination [19]
  - For general-purpose techniques to understand and control macro-level behavior
  - To devise a common model for different types of natural systems, to allow the comparison of such mechanisms
- At the meso scale, there is a need
  - To find ways of defining system boundaries
  - To find ways to formalize nonformalized phenomena
  - To adopt trust mechanisms

– To identify ways to empower social intelligence, as intelligence cannot be understood from a single agent or system

3.4 *Moving Multi-agent systems from Research to Practice (2009) [20]*

DeLoach [20] identified three main obstacles in adopting MAS for practical development:

- The lack of common interpretations of MAS concepts. This relates to two aspects: (1) no agreement on the concepts that constitute MAS and their interrelations; thus, DeLoach calls for reaching an agreement on these. (2) some of these concepts are too complex. For instance, some concepts are based on the BDI approach, which is more complex than the object-oriented approach. DeLoach further suggests to focus on the usefulness of AOSE and not on its agenthood properties.
- The lack of common notation and models. In that sense DeLoach calls for adopting common notation and models to allow for easy comparison among methods. He, however, suggests to avoiding formal standardization at this stage, as the AOSE community may need more time for its technologies to mature.
- The lack of industry acceptance of AOSE methodologies. DeLoach explains it by the lack of flexibility the AOSE methodologies introduce. Thus, he suggests to adopt the method engineering approach.

3.5 *Future Directions for Agent-Based Software Engineering (2009) [21]*

Winikoff [21] stated that there is a limited number of well-developed methodologies which have tool support and are used in various case studies. Nevertheless, their usefulness has not been proven yet, and there is a need for quantitative proofs rather high-level and abstract arguments. Thus, he suggested to act in various directions:

- Identifying the "value-added" of adopting agent technology
- Documenting case-studies
- Making other scientific communities such as service-oriented commuting, the grid, and autonomic computing, aware of the achievement made within the AOSE domain
- Developing useful tools to support the real users
- Understanding the problems in real settings

From the research point of view, Winikoff suggests to explore the following:

- From the foundation point of view, to examine the notion of goals and to further weave it with other important concepts such as interaction and organization
- From the engineering point of view, to further understand and develop the validation and verification aspects

3.6 Challenges and Directions for Engineering Multi-agent systems (2012) [22]

Later on, Winikoff [22] revised his observations, suggesting that there are many areas in which the agent paradigm was implemented and that some quantitative results were produced too. He also stressed that educational resources (e.g., books and tools) have become available as well. Having analyzed the industry involvement and adoption of agent technology, he found out that industrial interest in agent technology has declined. Thus, he suggested to reengage with the industry. Following the analysis performed by Dignum and Dignum [23] indicating that in agent-based projects AOSE techniques were used to a limited extent, he suggested to stop designing programming languages and methodologies, but rather to look at techniques that weave the micro and the macro aspects of such systems. In addition, he suggested to develop the thread of MAS assurance.

3.7 Engineering Multi-agent systems (2012) [24]

In 2012, a Dagstuhl seminar on the engineering of multi-agent systems was held [24]. The seminar's goal was to establish a roadmap for the AOSE domain. Thirty-seven leading researchers, mostly from the academia, participated and contributed. The seminar covered many aspects of agent-oriented software engineering. The observations produced in that seminar can be found in [25]. Following, we present the most relevant ones within the context of this book:

- With respect to verification of MAS, Natasha Alechina sets the following challenges: (1) Addressing scalability issues in model checking; (2) Addressing human agents; (3) Specifying systems at the right level of abstraction for the purpose of model checking; and (4) Dealing with undecidable situations.
- Rem Collier, based on his experience in developing an agent-based framework and in teaching MAS courses, stated that the agent community does not provide enough resources in order to support MAS development, and there is no good evidence that demonstrates the supremacy of agent technology. Thus, there is a need to develop better tools for debugging and testing as well as metrics

for evaluating the technology (and alternatives). These observations were also supported by Jeremy Baxter.

- Birna Van Riemsdijk suggested that there is a lack in evaluating various techniques and there is a need to employ empirical approaches that are based on data to demonstrate the usability of agent-based approaches. This was also reinforced by Tom Holvoet who concluded that there is a need for thorough evaluation and guarantees.
- Many of the participants agreed that the advances made within the domain of AOSE had a limited effect on communities other than the AOSE community. Tom Holvoet pointed out that the reason for low impact is that AOSE is 90 % software engineering and only 10 % MAS. As such, it is preferable that AOSE results are communicated to other relevant communities. Another important aspect that he referred to was real-world applications. He asserted that we need to further collaborate with the industry on this, as also mentioned by Jörg P. Müller and Amal El Fallah Seghrouchni. Christian Guttmann furthered that direction and stressed that there is a need to clearly point out the advantages of agent technology in order for the industry to adopt the agent notion.

In the following, we attempt to summarize the surveys discussed above and other publications made in that area. First, it seems that agent communication has gained much attention and has evolved continuously. This evolution includes various research directions, such as protocols, dialogs, and conversation mechanisms. The area of agent-oriented methodologies and programming languages received major attention in the early days of the AOSE field. In recent years, the pace of progress was much slower. This may suggest that there is a need to further explore this area. The same holds for the area of frameworks and architectures: in early AOSE times many frameworks, platforms, and architectures were devised; later on many of these became obsolete; nowadays only a few are active. From the research point of view, little innovation has been proposed in that area in recent years, although support tools and improved interfaces were provided. The area of AOSE adoption has advanced rather little over the years. It seems that the notion of agent-oriented software engineering is not well accepted within the industry. Although, various organizations, companies, and applications use the term agent, it is aimed at its weak notion of independent components, neglecting the strong notion of agenthood that includes autonomy, proactiveness, etc. Moreover, the engineering aspect is also hardly adopted by the industry, and AOSE methodologies and programming languages are rarely used.

4 Concluding Remarks

Agent-Oriented Software Engineering exists for almost 20 years. Its main purpose was to introduce Artificial Intelligence to Software Engineering and to suggest a new abstraction level in which agents play a major role. The domain has evolved

to encompass a wide range topics including formal specification and validation, programming languages, development methodologies, software engineering techniques, architectures and infrastructure, communications, and applications as well.

Despite the depth and breadth of research in the field, it seems that agent-based systems, and in particular AOSE, did not find their way to the industry. One of the main reasons for this is that the benefits of multi-agent systems have not been demonstrated, needless to say proved. We can say that instead of introducing AI to SE, SE was introduced to AI, and enabled its refreshment by establishing ways for implementing various theoretical AI approaches.

To address AOSE challenges and in particular the acceptance and adoption challenge, there is a need to clearly state for what purposes and under what conditions is the agent abstraction is better than alternative software engineering approaches. To this end, it could be useful to develop or adopt a set of metrics to measure the contribution of a specific approach to the industry. These metrics may include technical aspects, organizational aspects, financial aspects, and so on. Using the metrics, a comprehensive study should be performed to allow evaluation of AOSE comparison with other SE alternatives. Such evaluation and comparison can foster the AOSE research as they may reveal new improvement and innovation opportunities.

References

1. Jennings NR (2000) On agent-based software engineering. Art Intel 117(2):277–296
2. Wooldridge M, Jennings NR (1995) Intelligent agents: theory and practice. Knowledge Eng Rev 10(2):115–152
3. Wooldridge M, Jennings NR (1998) Pitfalls of agent-oriented development. In: Proceedings of the 2nd international conference on autonomous agents. Minneapolis/St. Paul, MN, pp 385–391
4. Wooldridge M, Jennings NR (1999) Software engineering with agents: pitfalls and pratfalls. IEEE Internet Comput 3(3):20–27
5. Wooldridge M (1997) Agent-based software engineering. Proc IEE Software Eng 144(1):26–37
6. Ricci A, Santi A (2012) Agent-oriented computing: agents as a paradigm for computer programming and software development. Int J Adv Software 5(1 & 2):36–52
7. Shoham Y (1993) Agent oriented programming. J Art Intel 60:51–92
8. Wooldridge M, Ciancarini P (2001) Agent-oriented software engineering: the state of the art. In: Proceedings of the first international workshop on agent-oriented software engineering. Springer, New York
9. Wooldridge M (2000) Reasoning about rational agents. MIT Press, Cambridge, MA
10. Fisher M (1994) A survey of concurrent METATEM — the language and its applications. In: Gabbay DM, Ohlbach HJ (eds) Temporal logic — proceedings of the first international conference (LNAI volume 827). Springer, Berlin, pp 480–505
11. Rosenschein SJ, Kaelbling LP (1996) A situated view of representation and control. In: Agre PE, Rosenschein SJ (eds) Computational theories of interaction and agency. MIT Press, Cambridge, MA, pp 515–540
12. Hoare CAR (1969) An axiomatic basis for computer programming. Commun ACM 12(10):576–583

13. Luck M, McBurney P, Preist C (2004) A manifesto for agent technology: towards next generation computing. Auton Agents Multi-Agent Syst 9(3):203–252
14. AgentLink: http://www.agentlink.org. Last accessed August 2013
15. Finin T, Labrou Y, Mayfield J (1997) KQML as an agent communication language. In: Bradshaw J (ed) Software agents. MIT Press, Cambridge, MA, pp 291–316
16. FIPA: Agent Communication (2013) http://www.fipa.org/repository/aclspecs.html. Last accessed August 2013
17. Zambonelli F, Omicini A (2004) Challenges and research directions in agent-oriented software engineering. Auton Agents Multi-Agent Syst 9(3):253–283
18. Parunak HVD, Brueckner S (2001) Entropy and self-organization, in multi-agent systems. In: Proceedings of the 5th international conference on autonomous agents. ACM Press, Montreal (CA), pp 124–130
19. Roli A, Mamei M, Zambonelli F (2003) What can cellular automata tell us about the behaviour of large-scale agent systems. In: Proceedings of software engineering for large scale agent systems, LNCS 2603. Springer, Heidelberg, pp 216–231
20. DeLoach SA (2009) Moving multi-agent systems from research to practice. Int J Agent-Orient Software Eng 3(4):378–382
21. Winikoff M (2008) Future directions for agent-based software engineering. Int J Agent-Oriented Software Engineering 3(4):402–410
22. Winikoff M (2012) Challenges and directions for engineering multi-agent Systems. CoRRabs/1209.1428
23. Dignum V, Dignum F (2010) Designing agent systems: state of the practice. Int J Agent-Orient Software Eng 4(3):224–243
24. Dix J, Hindriks KV, Logan B, Wobcke W (2012) Engineering multi-agent systems. Dagstuhl Report 2(8):74–98
25. Dix J, Hindriks KV, Logan B, Wobcke W (2013) Engineering multi-agent systems, seminar 12342, http://www.dagstuhl.de/mat/index.en.phtml?12342. Last accessed August 2013