Challenges and Research Directions in Agent-Oriented Software Engineering

Franco Zambonelli (franco.zambonelli@unimore.it)
Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, Via Allegri 13 – 42100 Reggio Emilia, Italy

Andrea Omicini (aomicini@deis.unibo.it)
Dipartimento di Elettronica Informatica e Sistemistica, Università di Bologna, sede di Cesena, Via Venezia 52, 47023 Cesena, Italy

Abstract. Agent-based computing is a promising approach for developing applications in complex domains. However, despite the great deal of research in the area, a number of challenges still need to be faced *(i)* to make agent-based computing a widely accepted paradigm in software engineering practice, and *(ii)* to turn agent-oriented software abstractions into practical tools for facing the complexities of modern application areas. In this paper, after a short introduction to the key concepts of agent-based computing (as they pertain to software engineering), we characterize the key emerging issues in multiagent systems engineering. In particular, we shows that these issues can be analyzed in terms of three different "scales of observation", i.e., in analogy with the scales of observation of physical phenomena, in terms of *micro*, *macro*, and *meso* scales. Based on this characterization, we discuss, for each scale of observation, what are the peculiar engineering issues arising, the key research challenges to be solved, and the most promising research directions to be explored in the future.

Keywords: multiagent systems, agent-oriented software engineering, intelligence engineering, self-organization

1. Introduction

Agents and multiagent systems have recently emerged as a powerful technology to face the complexity of a variety of today's ICT scenarios. For instance, several industrial experiences already testify the advantages of using agents in manufacturing processes (Bussmann, 1998; Shen and Norrie, 1999), Web services and Web-based computational markets (Kephart, 2002), and distributed network management (Bieszczad et al., 1998). In addition, several studies advise on the possibility of exploiting agents and multiagent systems as enabling technologies for a variety of future scenarios, i.e., pervasive computing (Abelson et al., 2000; Tennenhouse, 2000), Grid computing (Foster and Kesselman, 1999), Semantic Web (Berners-Lee et al., 2001).

However, the emergent general understanding is that multiagent systems, more than an effective technology, represent indeed a novel general-purpose paradigm for software development (Zambonelli and Parunak, 2004; Jennings, 2001). Agent-based computing promotes designing and developing applications in terms of autonomous software entities (agents), situated in an environment, and that can flexibly achieve their goals by interacting with one another in terms of high-level protocols and languages. These features are well suited to tackle the complexity of developing software in modern scenarios: *(i)* the autonomy of application components reflects the intrinsically decentralized nature of modern distributed systems (Tennenhouse, 2000) and can be considered as the natural extension to the notions of modularity and encapsulation for systems that are owned by different stakeholders (Parunak, 1997); *(ii)* the flexible way in which agents operate and interact (both with each other and with the environment) is suited to the dynamic and unpredictable scenarios in which software is expected to operate (Zambonelli et al., 2001a); *(iii)* the concept of agency provides for an unified view of the artificial intelligence results and achievements, by making agents and multiagent systems act as sound and manageable repositories of intelligent behaviors (Russel and Norvig, 2003).

In the last few years, together with the increasing acceptance of agent-based computing as a novel software engineering paradigm, there has been a great deal of research related to the identification and definition of suitable models and techniques to support the development of complex software systems in terms of multiagent systems (Gervais et al., 2004). These researches, which can be roughly grouped under the term "agent-oriented software engineering" (Wooldridge, 1997; Jennings, 2001), are endlessly proposing a variety of new metaphors, formal modeling approaches, development methodologies and modeling techniques, specifically suited to the agent-oriented paradigm. Nevertheless, the research is still in its early stages, and several challenges need to be faced before agent-oriented software engineering can fulfill its promises, becoming a widely accepted and a practically usable paradigm for the development of complex software systems.

In this paper, we analyze the main open research challenges in agent-oriented software engineering, i.e., those issues related to multiagent systems engineering that challenge traditional and current approaches to software engineering, and that call for innovative approaches and solutions. To better organize the presentation, we argue that different issues may arise depending on the "scale of observation" adopted to model and build a software system. At one extreme, the *micro scale* of observation is that where the system to be engineered has to rely on the controllable and predictable behavior of (a typically limited number of) individual agents, as well as on their mutual interactions. There, the key engineering challenges are related to extending traditional software engineering approaches toward agent-oriented abstractions. At the other extreme, the *macro scale* is the one where a multiagent systems is conceived as a multitude of interacting agents, for which the overall behavior of the system, rather than the mere behavior of individuals, is the key of interest, and for which novel "systemic" approaches to software engineering are needed. In between, the *meso scale* of observation is that where the need of predictability and control typical of the micro scale clashes with the emergence of phenomena typical of the macro scale. Therefore, any engineering approach at the meso scale requires accounting for problems that are typical of both the micro and the macro scale, and possibly for new problems specific to the meso scale.

Of course, in this paper, we do not claim to cover all the problems of agent-oriented software engineering, nor to exhaust the list of potentially interesting research directions. Still, our discussion aims at sketching a scenario articulated enough to give readers the clue of the fascinating amount of research work to be undertaken. In any case, we emphasize that the goal of our discussion is not simply to advertise the personal viewpoints of the authors. Rather, we have tried to collect and organize in a rationale and readable way the outcomes of a number of stimulating discussions that took place during the meetings of the "Methodologies and Software Engineering for Agent Systems (MSEAS)" SIG (Zambonelli, 2001; Zambonelli, 2003; Zambonelli et al., 2002) of the EU-funded Network of Excellence "Agentlink" (Luck et al., 2003). While we fully endorse the responsibility for what we state in this paper, we are at the same time greatly indebted to the participants of the MSEAS SIG for having shared with us their opinions and knowledge. In our turn, with this paper, we hope to be able to transmit to others that knowledge.

The remainder of this paper is organized as follows. Section 2 introduces the key concepts and motivations behind agent-oriented software engineering researches, by showing the generality of the agent-oriented paradigm and its impact in areas such as distributed systems engineering and artificial intelligence. Section 3 represents the core of the paper: it introduces our "scale of observation" characterization and, for each scale, it discusses the main research challenges and the most promising research directions. Section 4 concludes by mentioning some additional research issues that, although not detailed by this paper, may be worth of some considerations and further work by researchers.

2. Agent-Oriented Software Engineering: Concepts and Driving Forces

Other than a technology, agent-based computing can be considered as a new general-purpose paradigm for software development, which tends to radically influence the way a software system is conceived and developed, and which calls for new, agent-specific, software engineering approaches.

2.1. Agent-based Computing as a Novel Software Engineering Paradigm

The core concept of agent-based computing is, of course, that of an *agent*. However, the definition of an agent comes along with a further set of relevant agent-specific concepts and abstractions.

Generally speaking, an agent can be viewed as a software entity with the following characteristics (Jennings, 2001; Lind, 2001):

- *autonomy*: an agent is not passively subject to a global, external flow of control in its actions. That is, an agent has its own internal execution activity (whether a Java thread or some other sort of goal-driven intelligent engine, this is irrelevant in this context), and it is pro-actively oriented to the achievement of a specific task.

- *situatedness*: an agent performs its actions while situated in a particular environment, whether a computational (e.g., a Web site) or a physical one (e.g., a manufacturing pipeline), and it is able to sense and effect (portions of) such an environment.

- *sociality*: in the majority of cases, agents execute in open operational environments hosting the execution of a multiplicity of agents, possibly belonging to different stakeholders (think, e.g., to agent-mediated marketplaces). In these *multiagent systems* (MASs for short), the global behavior derives from the interactions among the constituent agents. In fact, agents may communicate/coordinate with each other (in a dynamic way and possibly according to high-level languages and protocols) either to achieve a common objective or because this is necessary for them to achieve their own objectives.

Looking at the above definition, it is clear that a MAS cannot simply reduced to a group of interacting agents. Instead, the complete modeling of a MAS requires explicitly focusing also on the *environment* in which the MAS and its constituent agents situate and on the *society* that a group of interacting agents give rise to. Modeling the environment implies identifying its basic features, the resources that can be found in the environment, and the way via which agents can interact with it (Omicini, 2001). Modeling agent societies (Moses and Tennenholtz, 1995) (or agent organizations (Fox, 1981), or agent ecologies (Parunak, 1997), the specific metaphor to be adopted depending on the specific characteristics of the application goals, and of the operational environment as well) implies identifying the overall rules that should drive the expected evolution of the MAS and the various roles that agents can play in such a society (Zambonelli et al., 2003; Omicini, 2001; Shoham and Tennenholtz, 1995). All the above considerations lead to the very general characterization depicted in Figure 1-left, whose basic abstractions and overall architecture totally differ from that of traditional software engineering approaches (Figure 1-right).

When considering the traditional, (historical) object-oriented perspective (Booch, 1994), the differences between the object-oriented and the agent-oriented perspective on modeling and building a software system are sharp. An object, unlike an agent, is in principle neither autonomous nor proactive, in that its internal activity can be solicited only by service requests coming from an external thread of control. In traditional object applications, there is not any explicit modeling of external "environment": everything is modeled in terms of objects, and objects either wrap environmental resources in terms of internal attributes or perceive the world only in terms of other objects' names/references. In addition, traditional object-based computing promotes a perspective on software systems in which components are "functional" or "service-oriented" entities. A global system architecture is conceived as a static functional decomposition, where interactions between components/objects are simply an expression of inter-dependencies (Bass et al., 2003; Shaw and Garlan, 1996; Shaw et al., 1995), and where concepts such as society or roles simply do not make any sense.

The above considerations make us claim that agent-based computing represents a brand new software engineering paradigm calling (as better discussed later) for a new discipline of *agent-oriented software engineering* (AOSE for short). Of course, we are aware that objects and components in today's distributed and concurrent systems are somewhat removed from the historical definition and are starting to approach our view of agents. Aspect-oriented programming explicitly aims at overcoming the intrinsic limitations of functional decomposition (Kiczales et al., 1997). Active objects and reactive components exhibits at least some degree of autonomy (Eugster et al., 2003). Context-dependencies in component-based applications, together with the explicit distinction

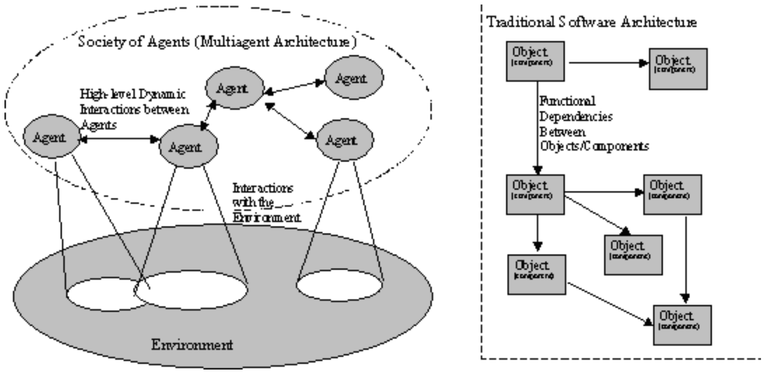

Figure 1. Multiagent Systems Architectures vs. Traditional Software Architectures

between active and passive objects, approach the distinction between agents and their environment (Cabri et al., 2002). The possibility, promoted by modern middleware (OMG, 1997; Ciancarini et al., 1999; Eugster et al., 2003), of both establishing open interactions and modeling interactions that are more articulated than simple request-response ones, makes complex object/component based system appear more like a dynamic society than as a static software architecture. In any case, the fact that traditional object- and component-based systems are abandoning traditional abstractions and are starting to adopt others, approaching those of agent-based computing, is the body of evidence that *(i)* a novel software engineering paradigm is needed and *(ii)* the agent-oriented is the right one.

2.2. The Promise of AOSE for Distributed Systems Engineering

As outlined above, today's software engineering approaches are increasingly adopting abstractions approaching that of agent-based computing. This trend can be better understood by recognizing that the vast majority of modern distributed systems scenarios are intrinsically prone to be developed in terms of MASs, and that modern distributed systems are already *de facto* MASs, i.e., they are indeed composed of autonomous, situated, and social components (Zambonelli and Parunak, 2004).

As far as autonomy is concerned, almost all of today's software systems already integrate autonomous components. At its weakest, autonomy reduces to the ability of a component to react to and handle events, as in the case of graphical interfaces or simple embedded sensors. However, in many cases, autonomy implies that a component integrates an autonomous thread of execution, and can execute in a proactive way.

This is the case of most modern control systems for physical domains, in which control is not simply reactive but proactive, implemented via a set of cooperative autonomous processes or, as is often the case, via embedded computer-based systems interacting with each other or via distributed sensor networks (Estrin et al., 2002). The integration in complex distributed applications and systems of (software running on) mobile devices can be tackled only by modeling them in terms of autonomous software components (Cabri et al., 2002). Internet based distributed applications are typically made up of autonomous processes, possibly executing on different nodes, and cooperating with each other, a choice driven by conceptual simplicity and by decentralized management rather than by the actual request for autonomous concurrent activities.

Today's computing systems are also typically situated. That is, they have an explicit notion of the environment where components are allocated and execute, and with which components explicitly interact. Control systems for physical domains, as well as sensor networks (Estrin et al., 2002), tends to be built by explicitly managing data from the surrounding physical environment, and by explicitly taking into account the unpredictable dynamics of the environment via specific event-handling policies. Mobile and pervasive computing applications recognize (under the general term of context-awareness) the need for applications to model explicitly environmental characteristics (such as, e.g., their position (Abelson et al., 2000)) and environmental data (e.g., as provided by some embedded infrastructure (Cabri et al., 2002)), rather than model them implicitly in terms of internal objects attributes. Internet applications and web-based systems, to be dived into the existing Internet environment, are typically engineered by clearly defining the boundaries of the system in terms of "application", including the new application components to be developed, and "middleware" level, as the environmental substrate in which components are to be embedded (Ciancarini et al., 2000).

Sociality in modern distributed systems comes in in different flavors: *(i)* the capability of components of supporting dynamic interactions, i.e., interaction established at run-time with previously unknown components; *(ii)* the somewhat higher interaction level, overcoming the traditional client-server scheme; *(iii)* the enforcement of some sorts of societal rules governing the interactions. Control systems for critical physical domains typically run forever, cannot be stopped, and sometimes cannot even be removed from the environment in which they are embedded. Nevertheless, these systems need to be continuously updated, and the environment in which they live is likely to change frequently, with the addition of new physical components and, consequently, of new software components and software systems (Tennenhouse, 2000; Kephart and Chess, 2003). For all these systems, managing openness and the capability to automatically re-organize interaction patterns is crucial, as is the ability of a component to enter new execution contexts in respect of the rules that are expected to drive the whole execution of the system. With reference to pervasive computing systems (Estrin et al., 2002), lack of resources, power, or simply communication unreachability, can make nodes come and go in unpredictable ways, calling for re-structuring of communication patterns, as well as for high-level negotiations for resource provision. Such issues are even exacerbated in mobile networking (Cabri et al., 2002; Mamei and Zambonelli, 2004) and P2P systems (Rowstron and Druschel, 2001; Ripeani et al., 2002), where interactions must be made fruitful and controllable despite the lack of any intrinsic structure and dynamics of connectivity. Similar considerations apply to Internet-based and open distributed computing. There, software services must survive the dynamics and uncertainty of the Internet, must be able to serve any client component, and must also be able to enact security and resource control policy in their local context, e.g., a given administrative domain (Ciancarini et al., 2000). E-marketplaces are the most typical examples of this class of open Internet applications (Noriega, 1997; Esteva et al., 2001).

In sum, today's distributed systems can be increasingly assimilated to the general MAS scheme of Figure 1-left. Thus, the explicit adoption of agent-based concepts in distributed systems engineering would carry on several advantages: (Jennings, 2001; Parunak, 1997):

- autonomy of application components, even if sometimes directly forced by the distributed characteristic of the operational environment, enforces a stronger notion of encapsulation (i.e., encapsulation of control other than of data and algorithms), which reduces the complexity of managing systems with a high and dynamically varying number of components;

- taking into account situatedness explicitly, and modeling environmental resources and active computational entities in a differentiated way, other than being the recognition of a matter of fact, provides for a better separation of concerns which, in turns, help reducing complexity;

- dealing with dynamic and high-level interactions (i.e., with societal rather than with architectural concepts) enables to address in a more flexible and structured way the intrinsic dynamics and uncertainties of modern distributed scenarios.

2.3. The Promise of AOSE for Intelligent Systems Engineering

MASs have been mainly developed as a concept within the AI research community, and earlier researches in the area disregarded software engineering aspects and focused only on AI ones. The recent recognition of agent-based computing as a novel software engineering paradigm – far from diminishing the importance of AI aspects – can even bring a renewed general interest to a variety of AI research findings.

Despite the different definitions and flavors of AI (which we are not going to discuss here any further), one should never forget that AI is mainly concerned with *building* intelligent systems: the very name of "Artificial Intelligence" literally suggests the notion of artifacts exhibiting intelligent behavior. Therefore, AI can be considered as an engineering field (dealing with constructive concerns), rather than simply a scientific one (dealing with understanding and predicting intelligent systems behavior). After all, one of the earliest and most influential AI papers (McCarthy, 1958) addresses the problem of how to actually write programs exhibiting some "common sense": besides showing the constructive concerns of early AI researchers, this clearly demonstrates how the notion of practical reasoning – reasoning about actions – plays a central role in AI from the very beginning.

Then, it is somewhat disappointing that nearly 30 years of AI research were conducted by having research groups concentrate on single, isolated aspects of AI (like artificial vision, knowledge representation, planning), and failed in producing a reasonable set of conceptual and practical tools, able to promote the integration of such a vast amount of research findings into the mainstream practice of software development. This is where agents are actually becoming a key abstraction in today AI.

The very notion of agent provides a uniform conceptual space where all the findings of the AI field can be easily framed and related, and can eventually find mainstream acceptance. First, agents are a practical and conceptually affordable entry point for new students and practitioners interested in AI, i.e., the right place where to experience with intelligent behaviors, and promotes the spreading of the results of AI research, as well as their exploitation in real world application domains. Also, the strong notion of encapsulation promoted by agents (which includes encapsulation of control) enables integrating in large software systems components with intelligent behavior (whatever the model, pattern, or technology actually used to embody intelligence), with no influence on the overall system architecture, nor on the overall development process. Clearly, this is likely to make even the more skeptical engineers more akin to experience with intelligent components. In addition, agent-oriented abstractions naturally provide for a new, powerful approach to the construction of intelligent systems, so that agents can not only pave the way for classical AI achievements toward industrial systems, but also promote original and more effective ways to solve highly-complex problems calling for system intelligence.

Correspondingly, the promise of AOSE is twofold. First, drawing from AI findings and making them part of the everyday software engineering practice. Then, raising the level of complexity of the problems that can be solved by human artifacts, by allowing artificial systems to incorporate ever-growing "amounts of intelligence" – whatever this could mean.

2.4. Current Directions in Agent-Oriented Software Engineering

A change of paradigm is always a dramatic event in any scientific and engineering field (Kuhn, 1962). As far as software engineering is concerned, the key implication is that the design and development of software systems according to a (new) paradigm can by no means rely on conceptual tools and methodologies conceived for a totally different (old) paradigm. Even if it is still possible to develop a complex distributed system in terms of objects and client-server interactions, such a choice appears odd and complicated when the system is a MAS or it can be assimilated to a MAS[1]. Rather, a brand new set of conceptual and practical tools – specifically suited to the abstractions of agent-based computing – is needed to facilitate, promote, and support the development of MASs, and to fulfill the great general-purpose potential of agent-based computing.

Researchers in the area of agent-based computing have recognized the above needs, and a vast amount of research work in now being focused on the above topics. It is out of the scope of this paper to survey all relevant work in the above areas of AOSE. A number of excellent and extensive articles have been written to this purpose (Iglesias et al., 1999; Ciancarini and Wooldridge, 2001; Gervais et al., 2004), and we

[1] The widespread acceptance of agent-based computing as a software engineering paradigm had and still has to fight against the opinions of those that either consider agents as a pure artificial intelligence technique (an endeavor that although originated from the urge to defend the artificial intelligence research community may end up contributing in leaving it in the ghetto) or simply consider agents as a buzz word to re-sell known research finding in the area of distributed systems (an endeavor that is mainly driven by quantitative and implementation-oriented considerations and that fully disregards qualitative and software engineering issues).

forward the interested reader to them. Still, a short summary of the current mainstream research directions is worth reporting.

- *Agent Modeling.* Novel formal and practical approaches to component modeling are required, to deal with autonomy, pro-activity, and situatedness. A variety of agent architectures are being investigated, each of which suitable to model different types of agents or specific aspects of an agents: purely reactive agents (Kiniry and Zimmerman, 1997; Parunak, 1997), logic agents (van der Hoek and Wooldridge, 2003), agents based on belief-desire and intentions (Kinny and Georgeff, 1996). Overall, these researches have so far notably clarified on the very concept of agency and on its different facets.

- *MAS Architectures.* The same as it is necessary to develop new ways of modeling the components of a MAS, it is necessary to develop new ways of modeling a MAS in its whole. Detaching from traditional functional-oriented perspectives, a variety of approaches are being investigated to model MASs. In particular, approaches inspired by societal (Moses and Tennenholtz, 1995; Omicini, 2001), organizational (Zambonelli et al., 2001b; Zambonelli et al., 2003), and biologically-inspired metaphors (Parunak, 1997; Bonabeau et al., 1999), are the subject of the majority of researches and are already showing the specific suitability of the different metaphors in different application areas.

- *MAS Methodologies.* Traditional methodologies of software development, driving engineers from analysis to design and development, must be tuned to match the abstractions of agent-oriented computing. To this end, a variety of novel methodologies to discipline and support the development process of a MAS have been defined in the past few years (Wooldridge et al., 2000; Zambonelli et al., 2003; Wood et al., 2001; Juan et al., 2002; Kolp et al., 2002), clarifying on the various set of abstractions that must come into play during MAS development and on the duties and responsibilities of software engineers.

- *Notation Techniques.* The development of specific notation techniques to express the outcome of the various phases of a MAS development process are needed, because traditional object- and component-oriented notation techniques cannot easily apply. In this context, the AUML proposal (Odell et al., 2001; Bauer et al., 2001), extending the standard UML towards agent-oriented systems, is the subject of a great deal of researches and it is rapidly becoming a *de facto* standard.

- *MAS Infrastructures.* To support the development and execution of MASs, novel tools and novel software infrastructures are needed. In this context: various tools are being proposed to transform standard MAS specifications (i.e., AUML specifications) into actual agent code (Poggi, 2002; Gomez-Sanz and Pavon, 2003); a variety of middleware infrastructures have been deployed to provide proper services supporting the execution of distributed MASs (Bergenti et al., 2002; Cabri et al., 2002; Noriega, 1997).

Clearly, all of the above works are contributing to increase the acceptance and the practical usability of the paradigm. Nevertheless, the number of challenging research problems to be solved and the number of potentially interesting research directions is much larger than it may appears from the above list.

3. Challenges in AOSE: the Micro, Macro, and Meso Scales

A key question that one should ask when facing the development of a MAS (or, which is the same, the development of any system in terms of AOSE abstractions and concepts) is: *what does it actually mean engineering a software system in modern and future scenarios?* A similar question arises when in need of discussing the key challenges and the promising research directions in AOSE.

Unfortunately, the more computing becomes ubiquitous and pervasive, the more answering the above question becomes difficult. Computational devices (from high-end computers, to wearables and micro sensors) and the associated software components will soon populate all of our physical spaces, homes, offices, and streets. All these components will interact with each other in the context of dynamic and complex networks. Also, by considering that the IPv6 addressing scheme will make it possible to assign an IP address to each and every square millimeter on the earth surface, the vision is that of an incredibly huge and open network, connecting billions of computer-based devices. In such a perspective, the very concept of software system becomes rather blurred. In fact, it is not easy to say what a software system actually is when: *(i)* the artifacts to be developed and engineered (e.g., a finite set of interacting software components) will be deployed in a pre-existing system of already executing software components; *(ii)* these artifacts will interact with a virtually infinite number of other components, in a scenario of transitive interactions and reciprocal influences that could possibly extend at a world-wide scale. Clearly, in such a scenario, any logical and physical boundary enabling to clearly define the subject of the engineering work vanishes as soon as the system is deployed or as soon as the engineer (as any good engineer should do) wonders about the impact that the deployment of its artifacts may have on the surrounding world. So, the very question raised at the beginning of this section (what does engineering a software system mean) should be better posed in a different way. That is: *what is the scale of observation at which one should situate the engineering work?*

It is a known fact that our universe is very different when observed from different perspectives. At our everyday scale of observation, classical mechanics applies, and relativity concepts – very relevant at cosmological scale – simply have no practical use. Nor it is relevant quantum mechanics, which becomes of some use only at a very small scales of observation or in very peculiar situations. Therefore, any study of our universe makes sense only when specifying the scale of observation at which the study apply. In a very similar way, any effective approach to AOSE requires fixing the scale of observation, to give meaning to the concept of software system and to enable identifying which issues have to be faced in its engineering. In particular, we distinguish here between three different scales of observation, i.e., the micro, the macro, and the meso scales, each raising very different issues in AOSE.

The identification of these three scales (and the terminology adopted for them) comes from an analogy with the world of nanoelectronics, MEMS, and molecule engineering. For the manufaturing of nanometer-size components, quantum phenomena appears and they have to be taken into account, since the characteristics and positions of each single atom become relevant. For the manufacturing of micrometer-size (and over) components, only collective phenomena can be observed, for which classical physical laws, disregarding the presence of and the behavior of single atoms, are more effective. In between, at what is called the meso scale, both the phenomena of classical and quantum mechanics can be observed, and both are required to be taken into account for the proper manufacturing of components.

By applying a similar characterization to MASs, we can identify three different scales of observation, as follows:

- *The Micro Scale.* The micro scale of observation is that which typically applies in traditional software development processes. Engineers involved in the analysis, design, and development of a MAS, in which a limited number (e.g., from a few units to a hundred) of agents have to be defined to interact towards the achievement of a specific application goal, may typically preserve and enforce a strict control and understanding over each and every component of the system. A software system of limited and identifiable size is the actual subject of the engineering work, and engineers' approach in building it is that of detailing the characteristics of each agent they develop, of the mechanisms underlying each inter-agent interaction, and of each interaction of the agents with their environment.

- *The Macro Scale.* The macro scale of observation is that in which the engineering work relates to understanding and controlling the behavior of huge software systems with a very large number of interacting agents (e.g., from several hundreds to billions), possibly distributed over a decentralized network and dived in a dynamic and uncontrollable operational environments. Here, due to the complex and decentralized nature of the systems subject of the engineering work, it possible neither to exert a strict control over each agent and each interaction nor to observe at a reasonable and manageable level of detail the single components of the system. Rather, the collective behavior of the system is what matters. We emphasize that a macro-scale approach to MAS engineering must be adopted both for the building of those systems that have been explicitly conceived for achieving their goals at a collective macro scale (e.g., swarm systems (Parunak, 1997) and sensor networks (Estrin et al., 2002)), as well as for the understanding and control of those systems that simply grow at a size such that the macro-scale approach becomes the only feasible one (e.g., global information economies (Kephart, 2002), world-wide P2P systems (Ripeani et al., 2002), the Grid (Foster and Kesselman, 1999)).

- *The Meso Scale.* The meso scale of observation is that which typically has to apply during the activity of deploying a micro-scale software system into a pre-existing macro-scale one. For what we previously said about future computing scenarios, a very limited number of software systems will exists in isolation, and mostly will be built for being dived into an existing networked scenario of high complexity. Therefore, working on the development of a system only in micro-scale terms will hardly be enough, and most of the times engineers will have to face the two basic questions of *(i)* what will be the impact on my system of being deployed it into an open and complex scenario and *(ii)* what will be the impact on the global system of the deployment of my own systems. In other words, there is a phase in the process of developing a MAS that requires taking into account both micro-scale and macro-scale aspects.

A simple example may be of help to clarify the above concepts. Consider the case of agent-mediated electronic marketplaces (e.g., auction sites). An engineer may wish to develop a MAS enabling a user to select specific shopping preferences and to have a set of agents travel over a world-wide network of auction sites to look for (and possibly buy) what the user has requested. At the micro scale of observation, the key issues are that the agents shall do their best to fulfill user requirements, not get cheated in any way, and get the best offers available on the network. At the other extreme (i.e., the macro scale of observation), the presence of a global agent-based information economy, in which prices are dynamically established by agents acting in auctions, introduces the problem of controlling the global degree of price fluctuations and the potential emergence of instabilities and economic crisis. In between, at the meso scale, one must ensure that agents being deployed on the network will be able to effectively interact with the global auction system in such a way that neither their efficiency will be affected (i.e., by starting buying unsatisfactory goods at unsatisfactory or totally unpredictable prices) nor the global stability of the system will be undermined (e.g., by having agents speculate on prices to influence them).

3.1. The Micro Scale

The micro scale is the subject of the vast majority of AOSE researches or, at least, of those researchers that classically tends to get categorized under the AOSE hat. This endeavor (which reflects also in the short state of the art summary reported in Subsection 2.4 is not surprising, because it naturally derives from an endeavor that has driven software engineering researches so far.

Until a few years ago, in fact, the micro one was the only meaningful scale of observation for software systems. Most software systems were closed, operating in isolation or, if somewhat open, interacting with the external world according to very static and predictable patterns of interaction. The duty of engineers was simply that of trying to build reliable and efficient self-contained software systems, and to ensure strict control and understanding over each and every component of these systems, for the sake of effective maintenance.

As a first consequence of this fact, most of earlier and current approaches to AOSE mainly focus on the building of small-size MAS (Wooldridge, 1997), and on the definition of suitable models, methodologies (Wooldridge et al., 2000; Wood et al., 2001) and tools (Odell et al., 2001; Bauer et al., 2001) for these kinds of systems. Even if these systems are sometimes claimed to be open, meso-scale and macro-scale issues are mostly disregarded. As another consequence, most AOSE practice at the micro scale has focused on trying to apply or extend traditional and well-assessed methods and tools (e.g., object-oriented ones (Bauer et al., 2001)) to AOSE. Nevertheless, neither all the potentially interesting research challenges at the micro scale have received enough attention, nor the practice of extending traditional methods has to be considered necessarily the best direction to follow, as we discuss below.

3.1.1. *Assessing the Advantages of Agents in Software Engineering*

The authors, most of the participants to the MSEAS SIG of Agentlink, and possibly most of the readers, are already confident about the potential advantages of the agent-oriented paradigm and about the fact that this is the paradigm to be adopted for the development of most complex software systems. However, this is not enough. All the considerations that we made in Section 2 about the potentials of the paradigm need to be supported by stronger quantitative arguments. Software engineers that are going to spend money and man months in the development of a complex software system will be hardly convinced to shift to a new paradigm (and pay the overhead involved in such a process) simply because it is conceptually elegant. Instead, they will require some evidence of the fact that this will help them save money and resources. We emphasize we are not referring here to the advantages that agents could bring to software systems, but to the advantages that agents could bring in the process of developing software systems.

At the moment, there are a very few works that prove, to some extent, the advantages of adopting an agent-oriented approach in software development. For instance, Cossentino et al. (Cossentino et al., 2003) describe the experience of their research group in building a robotics application according to an AOSE methodology, and they document in a quantitative way (i.e., man months effort) that the overall efficiency of the software development process notably improves over different (not specifically agent-based) approaches. As another examples, Cernuzzi and Rossi (Cernuzzi and Rossi, 2002) discuss the fact that the definition and evaluation of an AOSE methodology should take into account not only its suitability in matching the peculiar abstractions of agent-based computing (as most proposed methodologies do (Shehory and Sturm, 2001)), but also its suitability in facilitating software development and maintenance.

It is our opinion that, beside the justified research efforts in the attempt of identifying suitable AOSE methodologies for MASs engineering, much work is needed in the direction of evaluating (in a quantitative more than in a qualitative way) the agent-based paradigm and the associated methodologies, to asses their actual advantages over existing paradigms in software analysis, design, and maintenance.

3.1.2. *From Standard to Non-Standard and Extreme Processes*

The definition of agent-specific methodologies is definitely one of the most explored topics in AOSE, and a large number of AOSE methodologies – describing how the process of building a MAS should/could be organized – has been proposed so far (Shehory and Sturm, 2001). However, what characterizes most of the methodologies proposed so far is that they assume a very traditional cascade model (from analysis, to design, implementation, and maintenance) for organizing the process of building a MAS. This raises two key questions, which are also somewhat related the the previously identified challenge.

First, are we actually sure that the traditional software process model has to apply to MAS too? How can the abstractions of agent-based computing possibly impact on the very way that one should approach the building of a MAS? In a word of dynamic and complex software systems, do concepts such as requirements engineering, analysis, design, implementation, and maintenance still apply in the traditional way? We do not have any answer at the moment. Still, we think that scientists working in the area should really interrogate themselves about this problem, and possibly end up with novel software process models more suited to agent-based computing and (hopefully) more effective then traditional ones.

Second, it appears rather odd that most proposals for AOSE methodologies adopt a standard process model when, in the real world of industrial software development, such a standard model is rarely applied. It is a matter of fact that, in the 99% of the cases, software is developed following not structured process: analysis, design, and implementation, often collapse into the frenetic work of a bunch of technicians and programmers, directly interacting with clients (to refine typically vague specifications), and striving to deliver the work on time. In the mainstream community of software engineering, such a situation is getting properly attributed via the definition of novel software process models, specifically conceived to give some flavor of "engineering" to such chaotic and frenetic processes (e.g., agile and extreme software process models). In the area of AOSE, we think that a similar direction should be explored too, possibly exploiting the fact that the very abstractions of agents may promote the identification of different and more agile process models (as argued in the previous paragraph). A first promising approach in that direction is described by Knublauch in (Knublauch, 2002).

3.1.3. *Is AUML Enough?*

The acceptance of a new paradigm can be better promoted if it can be adopted by software engineers with a minimal effort, i.e., by letting them exploit as much as possible of the knowledge they already have, and by minimizing the need to acquire new knowledge and new states of mind. For these reasons (and also because the same considerations applies to scientists too), a large amount of research efforts are being spent in the area of AOSE to exploit and extend traditional (e.g., object-oriented) notation and modeling techniques for use in the context of MASs.

In particular, as we already anticipated in Subsection 2.4, agent-oriented extensions to UML (AUML) are the current subject of a great deal of researches (Odell et al., 2001; Bauer et al., 2001). For instance, extensions to UML diagrams has been proposed to account for the high-level nature of agent interactions (Odell et al., 2001) and for the societal aspects intrinsic in MAS architectures (Parunak and Odell, 2001). These extensions, focusing on specific aspects of the agent-paradigm with a set of intuitive and largely familiar diagrams, will turn out to be of great use towards acceptance of the paradigm. After all, also within the agent research community, AUML (even if it is neither fully specified nor standardized) is already becoming a *de facto* standard: newly proposed AOSE methodologies tend to adopt AUML as the basic notation techniques, and newly proposed interaction patterns in a variety of applications are usually expressed in terms of AUML diagram.

Despite the current enthusiasm for AUML, we are far from convinced that AUML is the ultimate answer. Beside the current period of transition, in which AUML will play an important role, we think that the complexity, dynamics, and situated nature of modern software systems cannot be effectively dealt with notations and modeling techniques originated for static and not situated software architectures. Whatever extensions will be proposed to AUML, they will intrinsically carry on the original shortcomings of the original object-oriented proposal. We are not the only ones thinking this: in the traditional software engineering community, the shortcomings of standard UML are becoming evident (Dori, 2002), and novel notations are being explored to account for higher dynamics and complexities. Accordingly, we think that a great challenge in the area of AOSE will be that of identifying brand new notations and modeling techniques, conceived from scratch to suit the specific characteristics of MASs.

We are not stating here that AUML should be abandoned. Simply, we are stating that, together with AUML, novel proposals should be very welcome and not simply discarded because they do not conform to widespread standards. The fact that reasonable proposals in this direction can be formulated and effectively compete with AUML can be verified by, e.g., the work of Sturm et al. (Sturm et al., 2003), describing a modified version of OPM specifically suited for MASs.

3.1.4. *Exploiting the Full Potential of Formal Models*

Formal models have always played a role in the context of software engineering researches (Meyer, 1985; Parnas, 1993): from formal notations for system design to formal frameworks for system verification, formal methods researches have tried to cover the whole spectrum of the engineering practice. Despite many efforts to show their usefulness in practice and to claim their usability and success (Hall, 1990; Bowen and Hinchey, 1995), formal methods seem to have mostly participated to the theory of software engineering rather than to its every day practice: there are far more university courses on "Formal Methods and Software Engineering" than formal methods actually used in the software engineering best practice.

Whichever is the reason of this, the unprecedented complexity of modern software systems will even more urgently call for formal methods helping engineers in designing, testing and verifying applications. Taking a look at the most recent results (SEFM 2003, 2003), the research focus appears to be shifting from providing a global formal approach for the engineering process as a whole to *(i)* addressing separate aspects with specialized formal methods and tools, and *(ii)* suitably integrating different formal approaches within the whole engineering process.

In this context, formal methods represent an obvious but fascinating challenge for AOSE. In particular, agents provide a new opportunity for formal approaches at the micro scale. At this level of observation, in fact, complexity is typically addressed not simply by adding new components, but also by increasing the capabilities of individual systems components. Given the natural encapsulation provided by agents, this means that suitably formalized agent architectures (like, say, BDI-like or logic-based agents) could be effectively used to build complex autonomous components whose behavior could be modeled using traditional AI techniques, like knowledge-based reasoning or first-order logics. Even more, compositional methods could be in principle used at the micro scale to foresee critical behavior of systems (and prevent undesired ones) based on the features of individual agent components. In particular, logic-based agent architectures seem to be particularly promising, since they could bring in the software engineering arena all the results that computational logics has achieved in the last 30 years of research. Several fora already exist that are trying to pave the way toward this direction (CLIMA IV, 2004; DALT 2003, 2003). Also, by their very nature, agent-oriented abstractions represent a conceptual framework promoting the seamless and clean integration of different and heterogeneous formal approaches. This vision of agent-oriented methodologies promoting the effective and factual use of formal methods in the (future) mainstream SE is thus another challenge for AOSE researchers.

3.1.5. *Promoting Intelligence Engineering*

As discussed in Subsection 2.3, intelligent behaviors have mostly been studied in isolation in the first decades of AI researches, with the only exception of robotics researches, having somehow assumed from the very beginning a unitary view of embodied intelligence. Furthermore, even though AI was born as a constructive discipline, methodological concerns have only occasionally found their way through AI researches. As a result, the distance between the practice of software engineering and the findings of AI has always been great.

An obvious challenge for AOSE today is then to provide an affordable way to introduce the engineering of intelligent behaviors in the mainstream software engineering practice. The point is not simply to exploit agents to provide a conceptual framework for AI techniques to be occasionally used within the standard software engineering practice. Rather, the key challenge for AOSE is to provide a methodological approach enabling software engineers to comfortably devise out and exploit selected AI solutions in their everyday practice. As an example, Operations Research – born as an independent research field long before AI – is today closely related to AI, and many AI tools are encapsulated and exploited in standard OR techniques. This provides programmers with a number of structured problems (like traveling salesman and Knapsack) that can be used as a reference and provide suitable algorithms for a vast amount of real-world application problems. Analogously, Data Mining has practically integrated a number of efficient AI techniques (for classification, clustering, temporal sequences, ...) that can be practically exploited within a multiplicity of different application scenarios, along with some criteria to properly select them.

Despite of the fact that some AI-related areas already provides engineers with criteria for techniques selection (pre-conditions to engineering methods), such efforts are again typically performed in isolation, without a global view of the system engineering issue. What is needed now – representing a challenge for AI in general, and for AOSE in particular – is an integrated approach enabling the engineering of complex application problems requiring system intelligence to be faced as a whole issue.

3.2. The Macro Scale

The macro scale of observation deals with engineering the overall behavior of large-scale MAS. It implies applying engineering methods at an observation level which abstract from the fact that a global system is made up of possibly individually deployed sub-systems and agents. The immediate consequence of this is that, at the macro scale, the behavior of individual agents or individual sub-systems, *per se*, is not relevant. What becomes instead relevant is the behavior of the system as a whole.

Skeptical readers would be tempted to say that the macro scale is by no means related to engineering (i.e., an activity of building something) but it is rather a matter of scientific investigation (i.e., an activity of observing, studying, and understanding). We agree that, traditionally, dealing with complex systems at the macro scale has mostly involved with such types of scientific activities. However, when human artifacts (as software systems are) grows to a level of complexity to make it impossible to control them at a micro scale of observation (as it is happening with our networks and distributed systems), we can no longer be satisfied by an activity of scientific investigation only. Instead, other than understanding these systems, we must find ways to engineer their behavior, i.e., to apply rigorous methodologies enabling us to exert – at least to some extent – some control over this systems and to direct as needed their behavior[2].

The Internet and the Web are the most sharp examples of this change of perspective. A few years ago, when the size of the Web (and of the underlying physical network of routers) grew dramatically huge and connected, researchers started investigating the structure and dynamics of such networks, and discovered peculiar and unexpected aspects (Crovella and Bestavros, 1996; Albert et al., 2000). Apart from the purely scientific interest of these discoveries, the newly acquired knowledge has become a basic background for the engineering of router topologies, of highly-accessed web sites, of HTTP caches, and of web indexing algorithms. For instance, it led to the identification of well-

[2] A similar change of endeavor, from scientific investigation to engineering, is taking place also in other scientific areas, such as ecology and climatology, where there is a great urge not only for observing and understanding systems but also for directing their evolution to meet specific goals (e.g., preserving bio-diversity or preventing catastrophic climate changes).

founded methodologies for evaluating the impact of the addition of new routers and new highways in the Internet.

Coming back to MASs, a large amount of experimental and simulation researches aimed at understanding what the behavior of very large size (possibly world wide) MASs will be is already available (Ripeani et al., 2002; Kephart, 2002; Roli et al., 2003). The next challenges in this area will be all somewhat related to promoting the emergence of a discipline of macro-scale AOSE.

3.2.1. *Measuring a System*

Engineering always implies some activity of measuring. At the micro scale, traditional software engineering has widely applied measuring methods to quantify, e.g., the complexity of a software system, its robustness, its mean time between failures, etc. Typically, all of these software metrics were proposed under the basic assumption of having micro-level visibility over the system's components. And, as far as the individual components of the system can be inspected and their behavior analyzed, it is rather clear that analogous sorts of metrics can applied to agents too (a specific micro-scale metric problem emerging at the meso scale, i.e., the measure of trust, will be analyzed in Subsection 3.3).

At the macro scale, the problem of measuring a system is possibly even more important. In fact, once we lose visibility and control over the individual components, the only way to characterize a system is to introduce some synthetic indicator of its behavior, i.e., some metrics able to capture some of its relevant characteristics and to quantitatively compare the behavior of two systems (or the behavior of the same system at different times and under different conditions). Unfortunately, the lack of micro-scale control makes most past work in the area of software measurement useless, and brand new approaches must be identified.

Some approaches to measure a MAS at the macro level have been recently proposed, motivated by contingent needs of showing specific characteristics of a MAS under exam. While some of these metrics appear to have a very restricted applicability scope, other promise to be of a much more general nature. For instance, Parunak and Bruckner, in (Parunak and Brueckner, 2001), introduced the general concept of "entropy" of a MAS, to show how a colony of ants globally impacts in the environment in which it is situated. Roli et al., in (Roli et al., 2003), adopted a compression algorithm to determine – in terms of compression rate – the degree of overall coordination achieved in a cellular MAS. Network science and earlier works on the Web topology (Albert et al., 2000) inspired the definition of very general macro-scale metrics to determine the connectivity characteristics of a network of agents.

In our opinion, there is a large amount of work to do to reach a higher understanding of what really needs to be measured in MASs (or in specific classes of MASs) at the macro scale, and of which of these metrics may be of a general nature. Getting inspiration from thermodynamics, information theory, network science, and from any science typically having to deal with macro-scale measures may be a good direction, as it may be starting from scratch in the search of distinguishing agent-specific macro-scale metrics.

3.2.2. *Controlling a System*

Measuring a system is very important to understand and characterize it. However, if the measure is not finalized to some consequent action, then measuring simply reduces to a scientific activity. Instead, in engineering, measuring a system is always finalized at ensuring that specific measurable values are within a pre-defined range, i.e., within that range characterizing an acceptable behavior of the system. In other words: the measuring of a system is finalized to its control; the controlling of a system is finalized to preserving specific observable (i.e., measurable) behaviors.

There are a variety of macro-scale control tasks that one may wish to enforce in a large MAS. Let us give a few examples. In a world-wide agent-based information economy, one may wish to avoid unpredictable large-range price fluctuations (Kephart, 2002), and ensure a reasonable stability of the global price systems. In sensor networks (Estrin et al., 2002) as well as in a Grids (Foster and Kesselman, 1999), one may wish that the various tasks performed by the agents on the sensors/nodes are properly distributed in a geographic area, and that highly overloaded or underloaded zones do not emerge. In a large-scale network of acquaintances, one may wish to ensure that pathological topologies leading to distorted information dissemination do not occur (Albert et al., 2000; Yu and Singh, 2003). Unfortunately, enforcing any type of control at the macro-level, as in the above examples, is indeed a challenging task.

Traditional control theory tell us that, once the overall parameters and laws governing the evolution of a system are known, enforcing an active control over it simply reduce at changing its operational parameters as required. However, in the area of MASs, this is far from being an easy task. First, in most of the cases, we have no way to know what are the operational parameters and the laws governing a MAS. Second, even if we knew, the lack of micro-scale control over the components of the system (decentralized and possibly belonging to different stakeholders) would require some novel approach to influence the behavior of the system from “out of the loop” (Tennenhouse, 2000), i.e., by adding new components to it or by changing the characteristics of its operational environment rather than by changing the behavior of its pre-existing components.

Some promising specific approaches in this direction are being undertaken. A large amount of experimental studies on global information economies (together with isomorphic studies performed on human economies) are telling us a lot on the actual laws governing them, and are paving the way for suitable methodologies and tools to control (at least to some extent) them (Kephart, 2002). Other studies on large networks of agents are telling us how and when specific pathological situations (e.g., not desired global synchronization and coordination of activities) may occur, and how one can (at least in principle) avoid them (Roli et al., 2003).

However, we still lack general-purpose models and tools (if any can be found) to understand and control at the macro-level the behavior of large-size MAS.

3.2.3. *On the Universality of MASs*

The very basic question of whether a general understanding and general control tools for MASs can exist is intriguing and challenging.

A variety of highly-heterogeneous physical systems, when analyzed in terms of their macro properties, exhibit surprisingly similar behaviors (Bar-Yam, 1992). For instance, phase transitions (i.e., bifurcations and shifts from ordered to chaotic behavior) always follow exactly the same laws, and a limited number of attractor classes can be used to describe the global dynamics of a huge number of physical systems. In other words, behind the high complexities ruling the macro-scale behavior of very diverse physical systems, there appear to be some unifying universal laws.

Given this, one could ask whether similar universal properties may be exhibited by MASs, and may be used to define general purpose engineering tools. Reasoning on such a possibility (suggested to us by Van Parunak (Parunak, 2003)) is not that weird. The different types of physical systems exhibiting the same universal behavior are typically made up of weakly correlated particles interacting according to some specific laws. While the degree of correlation may vary from system to system, as may do their specific interaction laws, the overall global behavior is – for some aspects – the same. MASs systems too, in general terms, can be all characterized by being made up of weakly correlated (i.e., autonomous) particles, interacting with each other according to some specific schemes. So, it would not be a big surprise (although definitely a scientific and technological breakthrough) discovering that MASs actually obey the same universal laws of complex physical systems.

In any case, it would be still very exciting to discover that, even if the general laws of MAS are completely different from that of physical systems, some underlying universal laws for computational systems of autonomous components existed. Such laws could then be used to identify unifying techniques to face (with different parameters) the macro-scale complexities of a variety of different MASs, from global information economies, to grids and sensor networks. What the promising directions to eventually identify such general universal laws are, it is hard to say. Nevertheless, a better understanding and exploitation of a number of findings in the study of complex natural and physical systems may be definitely of help.

3.2.4. *Sociology, Biology, and Beyond*

In the past few years, natural and social systems have played a very important role in many areas related to computer science. Biology acted as a major source of inspiration for the definition of a variety of novel search heuristics (from genetic algorithms to ant algorithms). Social systems, and specifically social networks, have played a major role in the understanding of the behavior of large computational networks (i.e., the Internet and the Web).

More recently, motivated both by the successes in the above areas and by the fact that MASs can be naturally abstracted in ecological or societal terms, the lessons of biology and sociology have started influencing the development of large-scale distributed computational systems and of large-scale MASs, with the goal of reproducing in MASs those robust and self-regulating macro-scale behaviors exhibited by ecological and social systems. For instance, algorithms inspired to ant foraging can be effectively exploited for mobile agents to find information distributed in a P2P network (Babaoglu et al., 2002). Similarly, the social phenomena of gossip, which turns out to be dramatically effective to propagate information in a social network, has been of inspiration for a variety of novel routing algorithms (Costa et al., 2004). In the specific area of MASs, a variety of other social and natural phenomena (e.g., negotiation-based interactions (Kephart, 2002), social conventions (Ciancarini et al., 2000; Omicini, 2001; Moses and Tennenholtz, 1995; Sichman et al., 1994), pheromone-based interactions (Shen et al., 2002)) have been exploited extensively towards the development of systems with robust and adaptive macro-scale behavior. Other than natural and social phenomena, some works also exploit specific classes of physical phenomena to achieve adaptive macro-scale behaviors. For instance, the Co-Fields approach (Mamei and Zambonelli, 2004) exploits virtual gravitational fields to orchestrate the overall movements of a large number of distributed mobile agents/robots.

In this context, we feel that the above studies – other than being of use for the building of specific classes of MASs – will be able to provide some general insights on how large-scale MASs work, and on how they can be effectively controlled with out-of-the-loop approaches. However, for these insights to be produced in the near future, *(i)* a larger variety of phenomena will have to be explored and *(ii)* novel formal modeling approaches will have to be produced.

With regard to the first point, there are a number of interesting physical and biological phenomena that are currently underestimated in the area of AOSE, and that instead have the potential to be effectively exploited in the building of robust and adaptive MASs. For instance, the emergence of regular spatial patterns observed in a variety of physical systems (Shinbrot and Muzzio, 2002) could be possibly exploited to implement novel and very effective strategies for distributed coordination in large-scale MASs. As another example, the patterns of distribution of specimen populations, as deriving from their specific interaction mechanisms (Wootton, 2001), could be of use towards the definition of adaptive MASs organizations and of effective strategies for dynamic division of labor.

With regard to the second point, the current lack of a common modeling of the different types of natural/biological/physical phenomena that are being exploited in large-scale MASs, is making impossible to compare heterogeneous approaches, as it is making impossible to document the performed experiences for the sake of reproducibility and reusability. The possibility of having a common modeling language would make it possible to build a catalogue of reusable patterns of global MASs behavior. Last but not least, the identification of a common modeling language would definitely make us very close to the identification (if any) of the universal laws of MASs at the macro scale.

3.3. The Meso Scale

The meso scale of observation comes into play whenever a (micro scale) software system, typically developed for some specific application goals to be achieved in the context of some (macro scale) open and complex operational scenario, has to be evaluated with regard to its actual deployment. We do not mean that meso-scale issues arises only in deployment phase, but that they arise whenever studying the characteristics of system from the deployment perspective. For instance, consider a system that has designed and developed as a micro-scale system, i.e., by adopting a micro-scale approach, with all its components at their place and working accordingly to well-defined application goals. The problem is that such system will be – sooner of after – dived for execution in a pre-existing, possibly world-wide, system, whose macro-scale characteristics can hardly be controlled from the micro scale. Thus, the clear need to preserve the micro-scale characteristics of the system has to harmonize with the macro-scale characteristics of the overall system and the reciprocal influences of the two.

Currently, several researches face the problem of enabling the execution of MASs to be dived in an open scenario. These include standardization efforts for enabling dynamic and open interoperability (for Intelligent Physical Agents, 2002), models for dynamic and open agent societies (Demazeau and Costa, 1996; Ferber and Gutknecht, 1998), as well AOSE methodologies explicitly conceived to deal with open agent systems (Zambonelli et al., 2003; Omicini, 2001). However, in our opinion, most of these works face meso-scale issues with a very restricted perspective, i.e., simply in terms of a slightly extended micro scale of observation. For instance, the problem of the openness of the systems is, in most of the cases, posed simply in terms of how the components of a software system can "connect" with components in the external world, and of how they can interact with them without getting damaged.

In our opinion, the meso scale of observation introduces much more critical issues than simply enabling interoperability (although this aspect is indeed necessary). Specifically, as we already anticipated, the key challenges are to face the dual aspects of: *(i)* understanding and controlling the impact on a software system of being dived in a macro-scale scenario and, viceversa, *(i)* understanding and controlling the potential impact that the deployment of a software system may have at a macro scale. Based on this formulation of the problem, it is rather clear that the meso scale is the one that poses the most challenging problems to AOSE. In fact, to effectively face the above two issues, one should assume: the availability of proper micro-scale models and tools (e.g., AOSE methodologies and agent-oriented formal methods); the availability of macro-scale ones (e.g., control methodologies and unifying models); and the possibility to organize the engineering work by taking them both into account.

We are tempted to say that the meso scale is so challenging that, beside the above two very general issues, it is very difficult to precisely identify more specific challenges and promising research directions. Nevertheless, we try below to go into details about a few more specific issues, well aware that most of the dimensions of the meso scale are likely to be largely unidentified.

3.3.1. *Identifying the Boundaries of a System*

When one starts considering the fact that a software system will become – at deployment time – part of a larger system, the problem of identifying the boundaries separating the two systems arise. In an open word where agents can get to life and die at any time, where mobile agents will roam across network domains accordingly to their own plans, and where the task to be accomplished by a software system can be delegated to external components (e.g., middleware services, mediators, brokers), the very concept of boundaries becomes very weak. In other words, it may be rather unclear what an engineer should consider as part of its own system (and thus include it in its work) and what should be instead simply considered as something not having to do with its work.

Some tentative solutions to deal with the boundary problem has been proposed so far. For instance, a common practice in open agent systems is to wrap any external entity into an agent that becomes part of the internal system, thus encapsulating any boundary effect into a set of well-identified wrapper agents. However, this practice does not solve the problem, but simply hides it inside some agents. Also, this practice is simply useless when, as it often happens, most of the agents of my system will have to interact with some parts of the external world. A more effective solution is to enforce all interactions in a MAS, both internal and external, to take place via some shared interaction infrastructure (e.g., a blackboard or a tuple space (Omicini and Zambonelli, 1999; Cabri et al., 2002)). In this case, the shared interaction infrastructure acts as a virtual space with well-defined boundaries and well-identifiable dynamic of interactions across boundaries. Unfortunately, this solution dramatically clashes with the usual perspective of inter-agent interactions: agents are typically assumed to be able to directly talk with each other in a peer-to-peer way, and without the mediation of some shared interaction space.

Possibly, a more general direction (abstracting from the actual presence of a shared infrastructure) could be of identifying and integrating in AOSE methodologies usable guidelines for the identification of boundaries, and of electing boundary conditions (that is, the modeling of the characteristics and dynamics of the interactions across boundaries) to a primary abstractions in open MAS development. What these guidelines should be and how boundary conditions could be modeled we just do not know. However, this issue directly leads us to the problem of formalization at the meso scale.

3.3.2. *Formal Models for Non-Formalizable Systems*
While it seems quite obvious what role could be played by formal methods at the micro scale, additional questions arise when considering the fuzzyness of such a concept at the meso scale. There, in fact, systems often come to be as unpredictable as at the macro scale, and non-formalizable as well – either for pragmatical or theoretical causes (Wegner, 1997). Even though parts of the system might be under the engineer's control, the external world here comes in across open boundaries, bringing about burdens of uncertainty and complexity.

Components to take into account in modeling may become too many to be represented at a glance; these components may have been developed by teams having worked separately on different portions of the system; multiple, heterogeneous and often dynamic application environments may have to be accounted for; and many different technologies, languages, paradigms and legacy systems have to be combined altogether in an effective and fruitful way. However, the meso scale is also the one where software engineers cannot afford to lose control of the components of the system they are building, despite the intrinsic explosion of complexity. Issues like security, quality of behavior, and automated verification are still absolutely critical for the engineer. Unfortunately, they are unlikely to be suitably addressed without the help of traditional formal methods. However, it often happens that at least some critical portions of a system can actually be modeled formally. This is the case for instance of meso-scale systems exploiting some shared interaction infrastructure.

Shared interaction infrastructures typically encapsulate and embody critical portions of the system behavior – by providing for knowledge management, security mechanisms, communication services, etc. – which are usually provided by the infrastructure to components in form of services by means of dedicated run-time abstractions – like daemons, servers, brokers, middle agents, etc. The key point is that if the shared interaction infrastructure itself comes along with a suitable formal characterization, or at least its key run-time abstractions are provided with a formal characterization, some system properties can be guaranteed and proven in principle (Omicini, 1999). In fact, if the dynamic behavior of critical run-time abstractions shared by the individual agents constituting a MAS can be modeled and predicted, the corresponding global system behavior can in principle be designed to be at least partially predictable independently of the individual (autonomous) behaviors of agents. An example is for instance discussed by the second author in (Omicini et al., 2004), where coordination abstractions provided at run-time by the agent coordination infrastructure are required to be predictable in their dynamics, and formally characterized. This fact ensures that some critical system behavior depending on the governance of agent interaction can be formally defined at the design stage and be ensured at run-time.

In general, a key challenge for AOSE at the meso scale of observation will be of devising formally-defined agent infrastructures, and of incorporating within suitable methodologies the associated formal methods and tools. This will allow engineers to super-impose at design time (and preserve down to execution time) critical system properties, and prove them formally despite of the general non-formalizability of complex MASs.

3.3.3. *Beyond Security: Infrastructures for Trust*

The dichotomy between the intrinsic complexity of software systems and the absence of accessible models to make system behavior both understandable and predictable – a dichotomy which becomes sharp at the meso scale of observation – raises the key issue of *trust* between humans and MASs. The difficulty in trusting systems that cannot be fully understood and whose behavior cannot be fully predicted do not affect only end-users, but also (and, in some sense, mostly) the engineers and developers which are responsible for the design and the actual functioning of such systems. Conceiving trustworthy models for the engineering of complex systems is of dramatic important for the technological progress characterizing our information society is a necessary condition for the widespread adoption and acceptance of MASs. So, trust becomes one of the most important "social" issue for MASs, as is already was for human systems. After all, when considering scenarios such as e-commerce or e-government, where the edge between human and artificial societies tends to blur, this appears as a quite natural consequence. In general, all the social issues involved in human societies, trust *in primis*, should be faced also in the construction of complex artificial systems like MASs.

Trust in information technology account for two main issues: *(i)* trust between humans and systems (in terms of both trust between users and systems, and trust between designers/engineers and systems), and *(ii)* trust between systems and systems (in terms of both trust among system components, and trust among components of different systems). The interpretation of MASs in terms of societies, promoted by agent-oriented approaches, makes it possible to face the two issues above within the same conceptual framework, adopting a uniform approach to explore general models and solutions.

Correspondingly, the identification of suitable models and technologies going beyond the mere concerns of security, and fully supporting instead the notion of trust in artificial systems, becomes a primary challenge for MAS research: workshops on trust are already deeply permeating MAS research, trying to address these concerns (TRUST 2003, 2003). However, this also implies the definition and development of infrastructures that not only provide for agents and MASs security, but also explicitly model and embed the notion of trust within suitable abstractions, with the expressiveness and effectiveness required by complex MAS engineering. As a trivial example, given an agent and/or a MAS, it is necessary for engineers to be able to characterize (i.e., measure) it in terms of how much it can be trusted upon, at any time in the system life, and to make such information accessible and understandable by users. This is relevant especially at the meso scale, where an agent has to interact in an open and uncertain world, thus making it even more difficult to understand and predict its course of actions, and trust it as well. The issue of making trust models of heterogeneous sources (psychology, economy, law, ...) match technology at the infrastructure level of MAS appears as one of the most challenging problems for future MAS research.

3.3.4. *Empowering Social Intelligence*

At the meso scale of observation, the complexity of systems does not allow any longer system components to be completely controlled/designed/governed merely as individuals. Correspondingly, at this scale of observation, intelligence embedded within agents (as from Subsection 3.1.4)is often not enough to build up intelligent systems. The very notion of situated intelligence, when seen through the eyes of intelligent systems' engineers, calls for a suitable design of what is outside the agents: *societies* that agents form and in which agents gets deployed, and *environments* where agents live (Omicini, 2001).

So, the design of intelligent systems seems to require: on the one hand, suitable design abstractions to support *social intelligence*, i.e., intelligence exhibited by agent societies, which cannot directly be ascribed to individual intelligent (component) agents (Ciancarini et al., 2000; Zambonelli et al., 2003); on the other hand, suitable infrastructures shaping the agent environment so as to fully enable and promote the exploitation of both individual and social intelligence. The former requirement clearly emerges from several AOSE methodologies, adopting organizational models to describe and design systems in terms of organizational structure (roles involved), organizational patterns (roles relationships), and organizational rules (constraints on roles and their interactions) (Zambonelli et al., 2003; Kendall, 1999). The latter requirement comes instead from many application scenarios characterized by articulated and dynamic organizational structures and coordination processes, where agent-oriented abstractions already proved their effectiveness, like inter-organization workflow management systems (Kappel et al., 1998), agent-based CSCW (Tripathi et al., 2002) and team-based cooperative military contexts (Giampapa and Sycara, 2002).

A key challenge for future AOSE research then is to provide models, technologies and methodologies for the support of social intelligence. Oh the one hand, this means defining suitable social abstractions that could incorporate some form of higher-level intelligence, governing agent interaction towards global (social) intelligent behavior. This is for instance, the case of notions like programmable coordination services (Viroli and Omicini, 2003) and e-institutions (Noriega and Sierra, 2002). On the other hand, this also involve enabling and promoting individual agent intelligent activity over the society structure and dynamics. This would clearly promote controlled self-reconfiguration and self-adaptation of intelligent systems: in fact, once enabled to inspect the social structure and dynamics, and allowed to affect it, an intelligent agent can in principle reason about the society, make inferences, and possibly plan its evolution, for instance to fix some undesired behavior, or to adapt to environmental changes (Omicini and Ricci, 2003).

4. Conclusions

The area of AOSE is definitely at a very early stage. While an increasing number of research groups gets involved in this topic to explore the implications of developing complex software systems according to the agent-oriented paradigm, a number of fascinating challenges are still open to investigation. As we have tried to overview in this paper, AOSE researches cannot simply reduce to define new agent-specific development methodology and to adapt existing notations. This research work is very important to promote the acceptance of the paradigm, but emerging application areas such as pervasive and grid computing will require much more than this. In the near future, software engineers will be asked to produce, deploy, and control, very complex software systems, made up of a possibly very large number of agents, to be dived in complex environments populated by millions of agents, and able to behave in a reliable, intelligent, and trusty way at any scale of observation. For these requirements to be fulfilled, the identification of novel approaches – possibly along the directions identified in this paper – will be necessary.

Of course, we have not the ambition of having covered, in the few pages of this paper, all of the possible challenges and research directions. It is very likely that a variety of other challenging engineering problems not discussed in this paper will affect the development of agent-based systems, and will call for the exploration of further promising research directions. For instance, we have not dealt here with the potential role that evolutionary (Sierra et al., 2002) and connectionistic approaches can play in the engineering of agent-based systems, at both the micro-level and the macro-level. We have not touched upon issues related to the engineering of user interfaces, although we are well aware that their scope and importance is increasing dramatically, as underlined by, e.g., the growing awareness of the relevance of the affective dimension in the area of emotional agents (Trappl et al., 2003). We have not been brave enough to hypothesize that the agent-based paradigm is possibly the right direction to eventually identify a sound hyper-Turing computational model (Teuscher and Sipper, 2002), and that research efforts in that directions would be worth. In addition, we have not extensively dealt with the fact that several future scenarios will not simply deal with intangible agents living in a cyberspace, but with physical agents (e.g., components of a modular robot (Shen et al., 2002) or smart dust (Estrin et al., 2002)) strongly interacting with the physical world and with our everyday environments. These sorts of interactions will indeed require novel modeling and engineering approaches, driven by new social and physical issues. Neither we have dealt with the rapid advances in nano- and bio-technologies (Bourianoff, 2003). What will it happen when our current concept of agent will actualize into a bio-mechanical, if not fully organic, being? How will the engineering issues implied in building a reliable machine will intertwine with the ethical issues implied in being the creators of novel specimens?

Or maybe we are simply going a bit too far, fantasizing on engineering issues for paradigms to come. For the likes of such as us, an exciting research future.

Acknowledgements

We thanks all the students and researchers that, in the past few years, have actively participated to the meetings of the MSEAS Agentlink SIG; and Michael Luck, for his efforts in making Agentlink a successful and exciting experience.

References

Abelson, H., D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Napal, E. Rauch, G. Sussmann, and R. Weiss: 2000, 'Amorphous Computing'. *Communications of the ACM* **43**(5), 43–50.

Albert, R., H. Jeong, and A. Barabasi: 2000, 'Error and Attack Tolerance of Complex Networks'. *Nature* **406**, 378–382.

Babaoglu, O., H. Meling, and A. Montresor: 2002, 'Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems'. In: *Proceedings of the 22nd International Conference on Distributed Computing Systems.* Vienna (A), pp. 15–22, IEEE CS Press.

Bar-Yam, Y.: 1992, *Dynamics of Complex Systems.* Reading (MA): Perseus Books.

Bass, L., P. Clements, and R. Kazman: 2003, *Software Architectures in Practice (2nd Edition).* Reading (MA): Addison-Wesley.

Bauer, B., J. P. Muller, and J. Odell: 2001, 'Agent UML: A Formalism for Specifying Multiagent Software Systems'. *International Journal of Software Engineering and Knowledge Engineering* **11**(3), 207–230.

Bergenti, F., G. Rimassa, A. Poggi, and P. Turci: 2002, 'Middleware and Programming Support for Agent Systems'. In: *Proceedings of the 2nd International Symposium from Agent Theory to Agent Implementation.* Vienna (A), pp. 617–622.

Berners-Lee, T., J. Hendler, and O. Lassila: 2001, 'The Semantic Web'. *Scientific American.*

Bieszczad, A., B. Pagurek, and T. White: 1998, 'Mobile Agents for Network Management'. *IEEE Communications Surveys* **1**(1), 2–9.

Bonabeau, E., M. Dorigo, and G. Theraulaz: 1999, *Swarm Intelligence. From Natural to Artificial Systems.* Oxford (UK): Oxford University Press.

Booch, G.: 1994, *Object-oriented Analysis and Design (2nd edition).* Reading (MA): Addison-Wesley.

Bourianoff, G.: 2003, 'The Future of Nanocomputing'. *IEEE Computer* **36**(8), 44–53.

Bowen, J. P. and M. Hinchey: 1995, 'Seven More Myths of Formal Methods'. *IEEE Software* **12**(4), 34–41.

Bussmann, S.: 1998, 'Agent-Oriented Programming of Manifacturing Control Tasks'. In: *Proceedings of the 3rd International Conference on Multi-Agent Systems.* Paris (F), pp. 57–63, IEEE CS Press.

Cabri, G., L. Leonardi, and F. Zambonelli: 2002, 'Engineering Mobile Agent Applications via Context-Dependent Coordination'. *IEEE Transactions on Software Engineering* **28**(11), 1034–1051.

Cernuzzi, L. and G. Rossi: 2002, 'On The Evaluation Of Agent Oriented Methodologies'. In: *Proceedings of the OOPSLA Workshop on Agent-oriented Methodologies.* Seattle (USA).

Ciancarini, P., A. Omicini, and F. Zambonelli: 1999, 'Coordination Technologies for Internet Agents'. *Nordic Journal of Computing* **6**(3), 215–240.

Ciancarini, P., A. Omicini, and F. Zambonelli: 2000, 'Multiagent Systems Engineering: the Coordination Viewpoint'. In: *Intelligents Agents VI: Agent Theories, Architectures, and Languages*, Vol. 1767 of *LNAI.* Springer Verlag, pp. 250–259.

Ciancarini, P. and M. Wooldridge: 2001, 'Agent-Oriented Software Engineering'. In: *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, Vol. 1957 of *LNCS.* Springer Verlag, pp. 1–24.

CLIMA IV: 2004, '4th International Workshop "Computational Logic in Multi-Agent Systems"'. Fort Lauderdale (FL).

Cossentino, M., L. Sabatucci, and A. Chella: 2003, 'A Possible Approach to the Development of Robotic Multi-Agent Systems'. In: *Proceedings of the 1st IEEE/WIC Conference on Intelligent Agent Technology.* Halifax (CA): IEEE CS Press.

Costa, P., M. Migliavacca, G. P. Picco, and G. Cugola: 2004, 'Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation'. In: *Proceedings of the 24th International Conference on Distributed Computing Systems*. Tokio (J), IEEE CS Press.

Crovella, M. and A. Bestavros: 1996, 'Self-Similarity in World Wide Web Traffic: Evidence and Causes'. *ACM Sigmetrics* **12**(4), 160–169.

DALT 2003: 2003, '1st International Workshop "Declarative Agent Languages & Technologies"'. Melbourne (AU).

Demazeau, Y. and A. C. R. Costa: 1996, 'Populations and Organizations in Open Multi-Agent Systems'. In: *Proceedings of the 1st National Symposium on Parallel and Distributed AI*. Hyderabad (IN).

Dori, D.: 2002, 'What UML should be: Why significant UML change is unlikely'. *Communications of the ACM* **45**(11), 82–85.

Esteva, M., J. A. Rodriguez-Aguilar, C. Sierra, P. Garcia, and J. L. Arcos: 2001, 'On the Formal Specifications of Agent Institutions'. In: *Agent-Mediated Electronic Commerce*, Vol. 1991 of *LNCS*. Springer Verlag, pp. 126–147.

Estrin, D., D. Culler, K. Pister, and G. Sukjatme: 2002, 'Connecting the Physical World with Pervasive Networks'. *IEEE Pervasive Computing* **1**(1), 59–69.

Eugster, P., P. A. Felber, R. Guerraoui, and A. Kermarrec: 2003, 'The Many Faces of Publish/Subscribe'. *ACM Computing Surveys* **35**(2), 114–131.

Ferber, J. and O. Gutknecht: 1998, 'A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems'. In: *Proceedings of the 3rd International Conference on Multi-Agent Systems*. Paris (F), pp. 128–135, IEEE CS Press.

for Intelligent Physical Agents, F.: 2002, 'FIPA Specifications'. `http:www.fipa.org`.

Foster, I. and C. Kesselman: 1999, *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco (CA): Morgan Kaufmann.

Fox, M. S.: 1981, 'An Organizational View of Distributed Systems'. *IEEE Transactions on Systems, Man, and Cybernetics* **11**(1), 70–80.

Gervais, M., J. Gomez, and G. Weiss: 2004, 'A Survey on Agent-Oriented Software Engineering Researches'. In: *Methodologies and Software Engineering for Agent Systems*. New York (NY): Kluwer.

Giampapa, J. and K. Sycara: 2002, 'Team-oriented Agent Coordination in the RETSINA Multi-Agent Systems'. In: *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*. Bologna (I), ACM Press.

Gomez-Sanz, J. J. and J. Pavon: 2003, 'Agent-oriented Software Engineering with INGENIAS'. In: *Proceedings of the 3rd Central and Eastern Europe Conference on Multiagent Systems*, Vol. 2691 of *LNCS*. pp. 394–403, Springer Verlag.

Hall, A.: 1990, 'Seven Myths of Formal Methods'. *IEEE Software* **7**(5), 11–19.

Iglesias, C., M. Garijo, and J. Gonzales: 1999, 'A Survey of Agent-Oriented Methodologies'. In: *Intelligents Agents IV: Agent Theories, Architectures, and Languages*, Vol. 1555 of *LNAI*. Springer-Verlag, pp. 317–330.

Jennings, N. R.: 2001, 'An Agent-based Approach for Building Complex Software Systems'. *Communications of the ACM* **44**(4), 35–41.

Juan, T., A. Pierce, and L. Sterling: 2002, 'ROADMAP: Extending the Gaia Methodology for Complex Open Systems'. In: *Proceedings of the 1st ACM Joint Conference on Autonomous Agents and Multi-Agent Systems*. Bologna (I): ACM Press, pp. 3–10.

Kappel, G., S. Rausch-Schott, and W. Retschitzegger: 1998, 'Coordination in Workflow Management Systems - A Rule-Based Approach'. In: *Coordination Technology for Collaboration Application*, Vol. 1316 of *LNCS*. Springer Verlag.

Kendall, E. A.: 1999, 'Role Modelling for Agent System analysis, Design, and Implementation'. In: *Proceedings of the 1st International Symposium on Agent Systems and Applications*. Palm Springs (CA), pp. 204–218, IEEE CS Press.

Kephart, J.: 2002, 'Software Agents and the Route to the Information Economy'. *Proceedings of the National Academy of Science* **99**(3), 7207–7213.

Kephart, J. O. and D. M. Chess: 2003, 'The Vision of Autonomic Computing'. *IEEE Computer* **36**(1), 41–50.

Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J. Loingtier, and J. Irwin: 1997, 'Aspect-Oriented Programming'. In: *Object-Oriented Programming*, Vol. 1241 of *LNCS*. Springer-Verlag, pp. 220–242.

Kiniry, J. and D. Zimmerman: 1997, 'A Hands-On Look at Java Mobile Agents'. *IEEE Internet Computing* **1**(4), 21–33.

Kinny, D. and M. Georgeff: 1996, 'A Methodology and Modelling Technique for Systems of BDI Agents'. In: *Proceedings of the 1st Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Vol. 1038 of *LNAI*. Springer-Verlag, pp. 56–71.

Knublauch, H.: 2002, 'Extreme programming of multi-agent systems'. In: *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*. Bologna (I), pp. 704–711, ACM Press.

Kolp, M., P. Giorgini, and J. Mylopoulos: 2002, 'A Goal-based Organizational Perspective on Multi-agent Architectures'. In: *Intelligent Agents VIII: Agent Theories, Architectures, and Languages*, Vol. 2333 of *LNAI*. Springer Verlag, pp. 128–140.

Kuhn, T.: 1962, *The Structure of Scientific Revolutions*. Chicago (IL): The University of Chicago.

Lind, J.: 2001, 'Issues in Agent-Oriented Software Engineering'. In: *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, Vol. 1957 of *LNCS*. Springer Verlag.

Luck, M., P. McBurney, and C. Priest: 2003, *Agent Technology: Enabling Next Generation Computing*. Southampton (UK): Agentlink II.

Mamei, M. and F. Zambonelli: 2004, 'Co-Fields: a Physically Inspired Approach to Distributed Motion Coordination'. *IEEE Pervasive Computing* **3**(1).

McCarthy, J.: 1958, 'Programs with Common Sense'. In: M. L. Minsky (ed.): *Semantic Information Processing*. Boston (MA): MIT Press, pp. 403–418.

Meyer, B.: 1985, 'On Formalism in Specifications'. *IEEE Software* **2**(1), 6–26.

Moses, Y. and M. Tennenholtz: 1995, 'Artificial Social Systems'. *Computers and Artificial Intelligence* **14**(3), 533–562.

Noriega, P.: 1997, *Agent-mediated Auctions: The Fishmarket Metaphor*. Barcelona (E): Ph.D Thesis, Universitat Autonoma de Barcelona.

Noriega, P. and C. Sierra: 2002, 'Electronic Institutions: Future Trends and Challenges'. In: *Cooperative Information Agents VI*, Vol. 2246 of *LNCS*. Springer-Verlag, pp. 14–17.

Odell, J., H. Parunak, and C. Bock: 2001, 'Representing Agent Interaction Protocols in UML'. In: *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, Vol. 1957 of *LNCS*. Springer Verlag, pp. 121–140.

OMG: 1997, 'CORBA 2.1 Specifications'. www.corba.org.

Omicini, A.: 1999, 'On the Semantics of Tuple-based Coordination Models'. In: *Proceedings of the 1999 ACM Symposium on Applied Computing*. San Antonio (TX), pp. 175–182, ACM.

Omicini, A.: 2001, 'SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems'. In: *Proceedings of the 1st International Workshop on*

Agent-Oriented Software Engineering, Vol. 1957 of *LNCS*. Springer Verlag, pp. 185–194.

Omicini, A., S. Ossowski, and A. Ricci: 2004, 'Coordination Infrastructures in the Engineering of Multiagent Systems'. In: *Methodologies and Software Engineering for Agent Systems*. New York (NY): Kluwer.

Omicini, A. and A. Ricci: 2003, 'Reasoning about Organisation: Shaping the Infrastructure'. *AI*IA Notizie* **XVI**(2), 7–16.

Omicini, A. and F. Zambonelli: 1999, 'Coordination for Internet Application Development'. *Journal of Autonomous Agents and Multi-Agent Systems* **2**(3), 251–269.

Parnas, D.: 1993, 'Predicate Logic for Software Engineering'. *IEEE Transactions on Software Engineering* **19**(9), 856–862.

Parunak, H.: 1997, 'Go to the Ant: Engineering Principles from Natural Agent Systems'. *Annals of Operations Research* **75**, 69–101.

Parunak, H.: 2003, 'Personal Communication'.

Parunak, H. and S. Brueckner: 2001, 'Entropy and self-organization in multi-agent systems'. In: *Proceedings of the 5th International Conference on Autonomous Agents*. Montreal (CA), pp. 124–130, ACM Press.

Parunak, H. and J. Odell: 2001, 'Representing Social Structures in UML'. In: *Proceedings of the 5th International Conference on Autonomous Agents*. pp. 100–101, ACM Press.

Poggi, F. B. A.: 2002, 'Agent-Oriented Software Construction with UML'. In: *The Handbook of Software Engineering and Knowledge Engineering - volume 2 - Emerging Technologies*. Singapore: World Scientific, pp. 757–769.

Ripeani, M., A. Iamnitchi, and I. Foster: 2002, 'Mapping the Gnutella Network'. *IEEE Internet Computing* **6**(1), 50–57.

Roli, A., M. Mamei, and F. Zambonelli: 2003, 'What Can Cellular Automata Tell Us About the Behaviour of Large-Scale Agent Systems'. In: *Software Engineering for Large Scale Agent Systems*, Vol. 2603 of *LNCS*. Springer-Verlag.

Rowstron, A. and P. Druschel: 2001, 'Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems'. In: *Proceedings of the 18th IFIP/ACM Conference on Distributed Systems Platforms*. Heidelberg (D), pp. 329–250.

Russel, S. and P. Norvig: 2003, *Artificial Intelligence: A Modern Approach*. Englewood Cliffs (NJ): Prentice Hall / Pearson Education International. 2nd Edition.

SEFM 2003: 2003, '1st International Conference "Software Engineering and Formal Methods"'. Brisbane, Australia.

Shaw, M., R. DeLine, D. Klein, T. Ross, D. Young, and G. Zelesnik: 1995, 'Abstractions for Software Architecture and Tools to Support Them'. *IEEE Transactions on Software Engineering* **21**(4), 314–335.

Shaw, M. and D. Garlan: 1996, *Software Architectures: Perspectives on an Emerging Discipline*. Englewood Cliffs (NJ): Prentice Hall, Englewood Cliffs (NJ).

Shehory, O. and A. Sturm: 2001, 'Evaluation of Modeling Techniques for Agent-Based Systems'. In: *Proceedings of the 5th International Conference on Autonomous Agents*. Montreal (CA): ACM Press.

Shen, W. and D. Norrie: 1999, 'Agent-Based Systems for Intelligent Manufacturing: A State of the Art Survey'. *International Journal on Knowledge and Information Systems* **1**(2), 129–156.

Shen, W., B. Salemi, and P. Will: 2002, 'Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots'. *IEEE Transactions on Robotics and Automation* **18**(5), 1–12.

Shinbrot, T. and F. J. Muzzio: 2002, 'From Noise to Order'. *Nature* **410**, 251–258.

Shoham, Y. and M. Tennenholtz: 1995, 'Social Laws for Artificial Agent Societies: Off-Line Design'. *Artificial Intelligence* **73**.

Sichman, J. S., R. Conte, C. Castelfranchi, and Y. Demazeau: 1994, 'A Social Reasoning Mechanism Based on Dependence Networks'. In: *Proceedings of the 12nd European Conference on Artificial Intelligence*. Amsterdam (NL), pp. 188–192.

Sierra, C., J. Sabater, J. Agust, and P. Garcia: 2002, 'Evolutionary Programming in SADDE'. In: *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*. Bologna (I), pp. 1270–1271, ACM Press.

Sturm, A., D. Dori, and O. Shehory: 2003, 'Single Model Method for Specifying Multiagent Systems'. In: *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems*. Melbourne (AU), pp. 121–128, ACM Press.

Tennenhouse, D.: 2000, 'Embedding the Internet: Proactive Computing'. *Communications of the ACM* **43**(5), 36–42.

Teuscher, C. and M. Sipper: 2002, 'Hypercomputation: Hype or Computation?'. *Communications of the ACM* **45**(8), 23–24.

Trappl, R., P. Petta, and S. Payr: 2003, *Emotions in Humans and Artifacts*. MIT Press.

Tripathi, A., T. Ahmed, R. Kumar, and S. Jaman: 2002, 'A Coordination Model for Secure Collaboration'. In: *Process Coordination and Ubiquitous Computing*. CRC Press, pp. 1–20.

TRUST 2003: 2003, '4th International Workshop on "Trust in Open Agent Societies"'. Melbourne (AU).

van der Hoek, W. and M. Wooldridge: 2003, 'Towards a Logic of Rational Agency'. *Logic Journal of the IGPL* **11**(2), 135–160.

Viroli, M. and A. Omicini: 2003, 'Coordination as a Service: Ontological and Formal Foundation'. *Electronic Notes in Theoretical Computer Science* **68**(3).

Wegner, P.: 1997, 'Why Interaction is more Powerful than Computing'. *Communications of the ACM* **40**(5), 80–91.

Wood, M., S. A. DeLoach, and C. Sparkman: 2001, 'Multiagent System Engineering'. *International Journal of Software Engineering and Knowledge Engineering* **11**(3), 231–258.

Wooldridge, M.: 1997, 'Agent-based Software Engineering'. *IEE Proceedings on Software Engineering* **144**(1), 26–37.

Wooldridge, M., N. R. Jennings, and D. Kinny: 2000, 'The Gaia Methodology for Agent-Oriented Analysis and Design'. *Journal of Autonomous Agents and Multi-Agent Systems* **3**(3), 285–312.

Wootton, T.: 2001, 'Local Interactions Predict Large-scale Patterns in Empirically Derived Cellular Automata'. *Nature* **413**, 841–844.

Yu, B. and M. P. Singh: 2003, 'Searching Social Networks'. In: *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*. Melbourne (AU), pp. 65–72, ACM Press.

Zambonelli, F.: 2001, 'Methodologies and Software Engineering for Agent Systems: SIG Introduction and Report of First Meeting'. *Agentlink News* **8**, 18–19.

Zambonelli, F.: 2003, 'SIG Report: Methodologies and Software Engineering for Agent Systems'. *Agentlink News* **13**, 16–17.

Zambonelli, F., F. Bergenti, and G. D. Marzo: 2002, 'SIG Report: Methodologies and Software Engineering for Agent Systems'. *Agentlink News* **9**, 23–25.

Zambonelli, F., N. Jennings, A. Omicini, and M. Wooldridge: 2001a, 'Agent-Oriented Software Engineering for Internet Applications'. In: *Coordination of Internet Agents: Models, Technologies, and Applications.* Berlin (D): Springer-Verlag, pp. 326–346.

Zambonelli, F., N. Jennings, and M. Wooldridge: 2001b, 'Organizational Abstractions for the Analysis and Design of Multi-Agent Systems'. In: *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, Vol. 1957 of *LNCS*. Springer Verlag, pp. 253–252.

Zambonelli, F., N. R. Jennings, and M. Wooldridge: 2003, 'Developing Multiagent Systems: The Gaia Methodology'. *ACM Transactions on Software Engineering and Methodology* **12**(3), 417–470.

Zambonelli, F. and H. Parunak: 2004, 'Toward a Change of Paradigm in Computer Science and Software Engineering: a Synthesis'. *The Knowledge Engineering Review* **18**.