The PASSI and Agile PASSI MAS Meta-models Compared with a Unifying Proposal

Massimo Cossentino[2], Salvatore Gaglio[1,2],
Luca Sabatucci[1], and Valeria Seidita[1]

[1] Dipartimento di Ingegneria Informatica (DINFO),
University of Palermo,
Viale delle Scienze, 90128 -Palermo- Italy
[2] Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR),
Consiglio Nazionale delle Ricerche(CNR),
Viale delle Scienze, 90128 -Palermo- Italy
cossentino@pa.icar.cnr.it, gaglio@unipa.it, sabatucci@csai.unipa.it,
seidita@csai.unipa.it

Abstract. A great number of processes for multi-agent systems design have been presented in last years to support the different approaches to agent-oriented design; each process is specific for a particular class of problems and it instantiates a specific MAS meta-model. These differences produce inconsistences and overlaps: a MAS meta-model may define a term not referred by another, or the same term can be used with a different meaning.

We think that the lack of a standardization may cause a significant delay to the diffusion of the agent paradigm outside research context. Working for this unification goal, it is also necessary to define in unambiguous way the terms of the agent model and their relationships thus obtaining a unified MAS meta-model. In this work we propose the PASSI MAS meta-model, the results of its adaptation to the needs of an agile process (Agile PASSI), and a comparison with an existing unifying proposal of MAS meta-model composed by considering three different processes (ADELFE, Gaia and PASSI).

1 Introduction

In order to approach the design and development of a multi-agent system (MAS) in a rigorous way, many approaches have been explored; all of these deal the development phases addressing high level terms such as agent, goal, role, task and collaboration. Hence, the design of a system may be seen as the instantiation of these elements in order to fulfill some specific problem requirements. The description of the elements involved in the design phases, and their relationships, can represent one of the fundamental steps in building a new one and specifically in the definition of its MAS meta-model (MMM hence afterward). The various agent-oriented design processes, presented in these years, are significantly different: goal-oriented, situation-oriented, requirement-oriented are examples of

M. Pěchouček, P. Petta, and L.Z. Varga (Eds.): CEEMAS 2005, LNAI 3690, pp. 183–192, 2005.

different philosophical possible approaches [1,2,3,4,5]. Each of these proposes a different way to face modeling; this diversity is caused by observing the system from different perspectives, considering different aspects of the problem and specific theoretical background or a specific application context. Besides each process forces the designer to assign an implicit meaning to each MMM component and that is often not coherent with the choices of other authors.
Even if this variety of design processes may be viewed as a richness, the differences among their meta-model components could create some perplexities when a designer moves from a design process to another, or when two designers try to communicate about a shared solution (pattern for agents).
A first step toward interoperability among different agent oriented design processes was the proposal of a unifying MAS meta-model [6] created starting from three different approaches (Adelfe, Gaia, and PASSI).

The purpose of this work is to present a description of the MAS meta-model of two design processes that use a quite different approach (PASSI[4,7] and Agile PASSI[8,9]). In this analysis we put in evidence two aspects related to the unifying MMM definition: i) we can highlight the differences among elements definition from comparing specific MMMs with the unifying one; ii) we can also show how to derive a specific MMM from the unifying one.
This paper is structured as follows: in section 2 we present the PASSI and Agile PASSI design processes while their meta-models are described in section 3 where we also underline the most relevant differences among them, finally in section 4 we present the unifying MMM resulting from the study in [6] and in 5 we compare the three presented MAS meta-models.

2 The PASSI and Agile PASSI Design Process

PASSI (Process for Agent Societies Specification and Implementation)[4,7] is a step-by-step requirement-to-code design process conceived for developing multi-agent systems. It is characterized by some distinctive features: (i) it is requirement driven, (ii) it is iterative and incremental, (iii) it focus the attention to ontological model of the domain in the design of agents.

PASSI is composed of five models addressing different design levels of abstraction. The **System Requirements Model** represents an anthropomorphic model of the system requirements in terms of agency and purpose. It consists of a functional description of the system: the designer identifies system requirements using use case diagrams, and organizes them in responsibility groups (that will assigned to agents). The next model, the **Agent Society Model**, fully exploits the agent paradigm: now an agent is seen as an autonomous entity capable of pursuing an objective through its autonomous decisions, actions and social relationships. The activities involved in this model aim to depict agent internal plans, knowledge and social abilities in order to model interactions and dependencies among entities of the society. The **Agent Implementation Model** defines the implementing details of the solution in terms of classes, attributes and methods. In this phase the designer uses conventional class diagrams, to

describe the static structure of the involved agents, and activity or state chart diagrams, to describe the behavior of individual agents. The **Code Model** is a representation of the solution at the code level while the **Deployment Model** describes the distribution (and their eventual migrations) of sub-systems across available hardware processing units.

Agile PASSI [9,8] derives from PASSI through the reuse of some of its fragments and it has been assembled complying a method engineering approach [10,11,12]. It is a light process created, according to the agile manifesto [13], with the aim to develop, in a short time, small-medium size systems. An agile process is easy to understand and to use because it is principally code oriented; for these reasons Agile PASSI comes to be well suited for those applications where coding is more important than documentation. In order to be compliant with agile modeling principles [13,14], Agile PASSI is an iterative process; it is composed of five steps (a low number) and it strongly oriented to communication among customers and developers during all the development phases (and in particular during the planning one). The fragments (portions of the design process) we have extracted from PASSI are: (i) *Domain Requirements description* (the description of system functionalities through use case diagrams), (ii) *Agent Identification* (the identification of logically related sets of functionalities that are put under an agent's responsibility), (iii) *Domain Ontology description* (the description of the agent knowledge in term of concepts, predicates and actions), (iv) *Code Reuse* (a technique for reuse portion of projects and code using design patterns) and (v) *Testing* (single agent and society test). This selection was done taking in consideration the PASSI philosophy: we have maintained use case diagrams as the base for agents identification and we have not changed the fundamental role of the ontology in the process. From the other side we respected the requirements for an agile process: low importance for the completeness of documentation and rapid code production.

The result of the composition of these fragments is a new process including five steps: **Requirements**, a model of the system requirements that is composed of two activities: Planning and Sub-Domain Requirements Description; **Agent Society**, a view of the agents involved in the solution, their interactions and their knowledge about the world. It is composed of two activities: Domain Ontology Description and Agent Identification; **Test Plan**, the phase of test planning; **Code**, a solution domain model at code level; **Testing**, the performing of the previous planned tests.

3 The PASSI and Agile PASSI MAS Meta-Models

The description of the PASSI MAS meta-model (Figure 1) addresses three logical areas: (i) the problem domain, (ii) the solution domain and (iii) the agency domain; they are introduced in an order that reflects our choice of an agent approach for solution refinement and modeling.

In the problem domain we include components describing the requirements the system is going to accomplish: these are directly related to the requirements

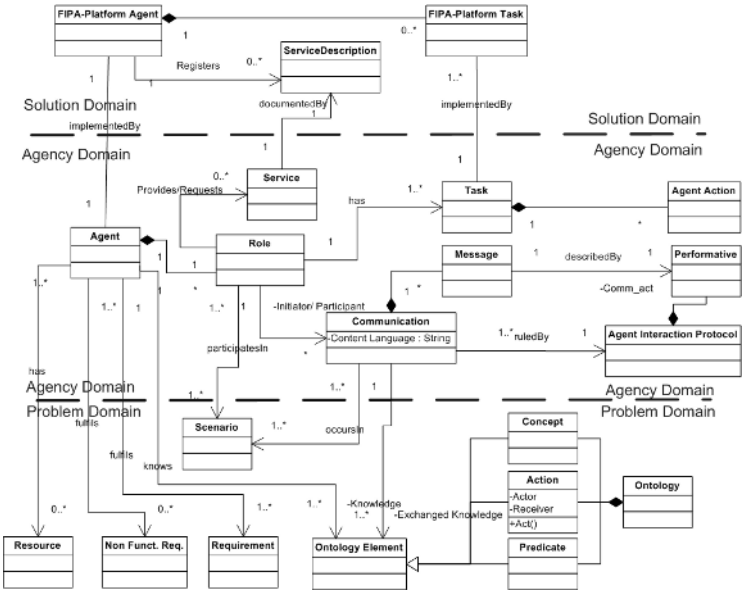

Fig. 1. The PASSI MAS meta-model

analysis phase of the PASSI process. Then we introduce the agency domain components; they are used to define an agent solution for the problem. Finally, in the PASSI MMM solution domain, agency-level components are mapped to the adopted FIPA-compliant implementation platform elements (we suppose the platform supports at least the concepts of agent and task); this represents the code-level part of the solution and the last refinement step.

Going into the details of the model, we can see that (Figure 1), the Problem Domain deals with the user's problem in terms of scenarios, requirements, ontology and resources. Scenarios describe a sequence of interactions among actors and the system to be built; Requirements are represented with conventional UML use case diagrams. The ontological description of the domain is composed of concepts (categories of the domain), actions (performed in the domain and effecting the status of concepts) and predicates (asserting something about a portion of the domain, i.e. the status of concepts). Resources are the last element of the problem domain. They can be accessed/shared/manipulated by agents. A resource could be a repository of data (like a relational database), an image/video file or also a good to be sold/bought. The Agency Domain contains the components of the agent-based solution. In PASSI an agent is responsible for realizing some functionalities descending from one or more functional requirements. It also has to respect some non functional requirement constraints (like for instance performance prescriptions). It lives in an environment from which it receives perceptions (the related knowledge is structured according to the designed domain ontology). Sometimes an agent has also access to available resources and it is capable of actions in order to pursue its own objectives or

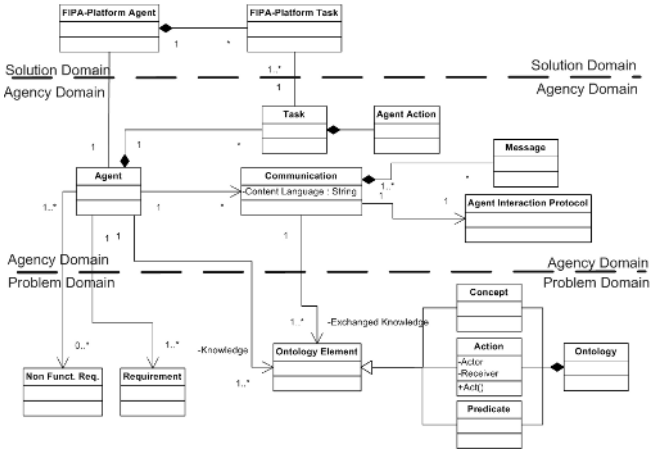

Fig. 2. The Agile PASSI MAS Meta-Model

to offer services to the community. Each agent during its life plays some roles. A role is a peculiarity of the social behavior of an agent. When playing a role, an agent may provide a service to other agents. In PASSI, a task specifies the computation that generates the effects of a specific agent behavioral feature. It is used with the significance of atomic part for defining the overall agent's behavior. This means that an agent's behavior can be composed by assembling its tasks and the list of actions that are executed within each task cannot be influenced by the behavior planning. Tasks are structural internal components of an agent and they contribute to define the agent's abilities; these cannot be directly accessed by other agents (autonomy) unless the agent offers them as a set of services. A communication is an interaction among two agents and it is composed of one or more messages. The information exchanged during a communication is composed of concepts, predicates or actions defined in the ontology. The flow of messages and the semantic of each message are ruled by an agent interaction protocol (AIP). The last Agency Domain element (Service) describes a set of coherent functionalities exported by the agent for the community.

The Implementation Domain describes the structure of the code solution in the chosen FIPA-compliant implementation platform and it is essentially composed of three elements: (i) the FIPA-Platform Agent that is the base class catching the implementation of the Agent entity represented in the Agency domain; (ii) the FIPA-Platform Task that is the implementation of the agent's Task, (iii) the ServiceDescription component that is the implementation-level description (for instance an OWL-S file) of each service specified in the Agent Domain.

Like the previous one, the Agile PASSI MMM (Figure 2) is partitioned in three logical areas: (i) problem domain, (ii) agency domain and (iii) solution domain. Agile PASSI was assembled starting from fragments extracted from PASSI, so the collection of MAS components descends from these fragments following

a particular design process [8] also considering the particular applications the agile process was conceived to solve. An agile process principally addresses code production, so in this case MAS meta-model components are mainly centered on the agent element and its related implementation parts. Using Agile PASSI a multi-agent system is conceived following five phases: planning, requirements design, agent society design, coding and testing; during the first two phases the elements present in problem domain area of MMM are instantiated; these elements are: i) functional and non functional requirements (used to describe the user point of view on the problem solution), and ii) domain ontology (composed of concepts, predicates and actions). As regards the Agency Domain, its central component is obviously the agent that is conceived in the same way as it is in conventional PASSI; it is composed of tasks representing significant (but not divisible) parts of its behavior and its capability of pursuing an objective realizing some functionalities, besides it uses communications to interact (communicating or requesting collaborations) with other agents, each communication being composed of messages ruled by an agent interaction protocol (like it is in PASSI). The solution domain is nearly the same of conventional PASSI since it is composed of the *Agent* and *Task* implementation elements.

4 A Unifying MAS Meta Model

An initial proposal of unifying MAS meta-model has been presented by C. Bernon et al. in [6] with the aim of contributing to the interoperability among agent oriented design processes; in their work the authors started with a comparison of some existing process with a specific attention for the differences among their MAS meta-models (MMM) components. The three studied design processes (ADELFE, Gaia and PASSI) are quite generic, in fact none of them refers to a specific agent architecture (like BDI or purely reactive agents).

The study was conducted using the classification of the terms (representing a MAS) accordingly to the following four categories: (1) Agent Structure: agent, role, responsibility, task, goal, plan and service; (2) Agent Interactions: (direct and undirect) communication, protocol, message; (3) Agent Society and organizational structure: social structure and organization rule; (4) Agent Implementation: FIPA-Platform agent and FIPA-Platform Task. The analysis confirmed that in the processes under exam, multiple definitions exist for the same component/concept; some are quite similar (there are small differences in the meaning), while some others are completely different. In order to maintain some level of generality they defined an unifying MMM (in Figure 3) with the aim to be used as a reference point for further comparisons and discussions about different design processes and related components.

5 A Comparison of the Presented Meta-models

It is a common belief that a general process, suitable to solve any kind of problems, does not exist, so it is clear that a MAS meta-model as huge as the one

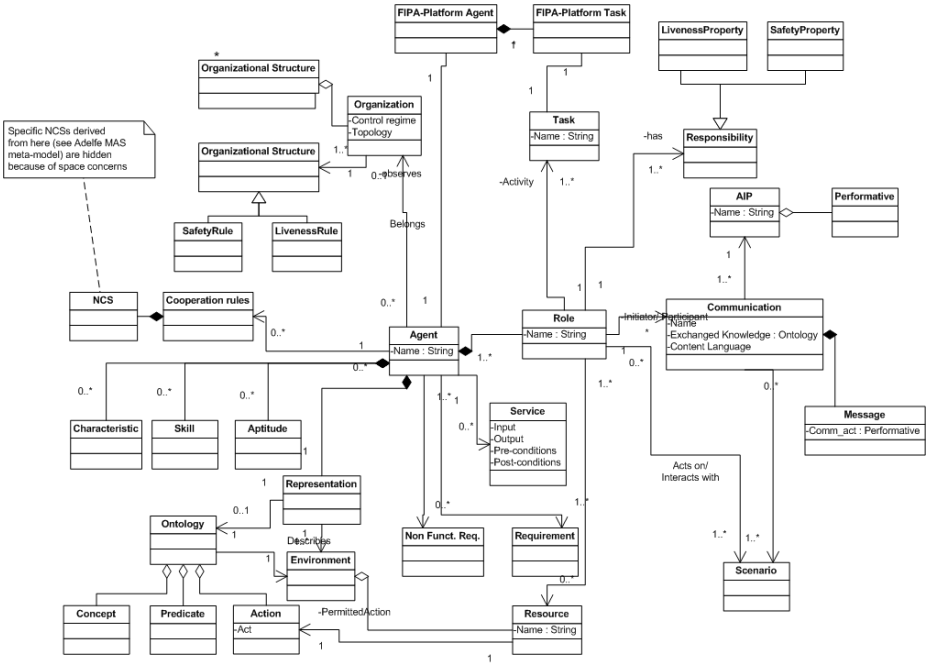

Fig. 3. The Unifying MAS meta-model (from [6])

described in section 4 cannot be used without some level of customization of its structure. Specializing different elements from MMM meets the different design philosophies on which each design process is based; now we will compare the PASSI and Agile PASSI MMMs with the unifying one with the aim of pointing out the differences among various concepts without forgetting that the unifying meta-model elements derives from the unification of three existing design processes already including PASSI.

Our comparison will deal with the four specific aspects of the MMM already discussed in the previous section. In the following we will refer to Figure 4 to show what elements and what relationships from the unifying meta-model we can find in the PASSI one (black drawn elements), what elements and relationships are newly introduced (dark-filled elements) and how the concepts and their definitions are specialized to create the MAS meta-model representing the PASSI design process. Elements from the unifying MMM that are not used in PASSI/Agile PASSI are gray-designed and as it can be seen they are a significant part of the model itself. Besides we will underline that because of slightly different interpretations of meta-model concepts (and philosophical choices that are beyond of each approach) it is possible that meta-models sharing the same elements can lead to significantly different design processes.

Agent Structure: in PASSI an agent is defined as a composition of roles; tasks, specifying a specific computation generating some kind of behavior, are associ-

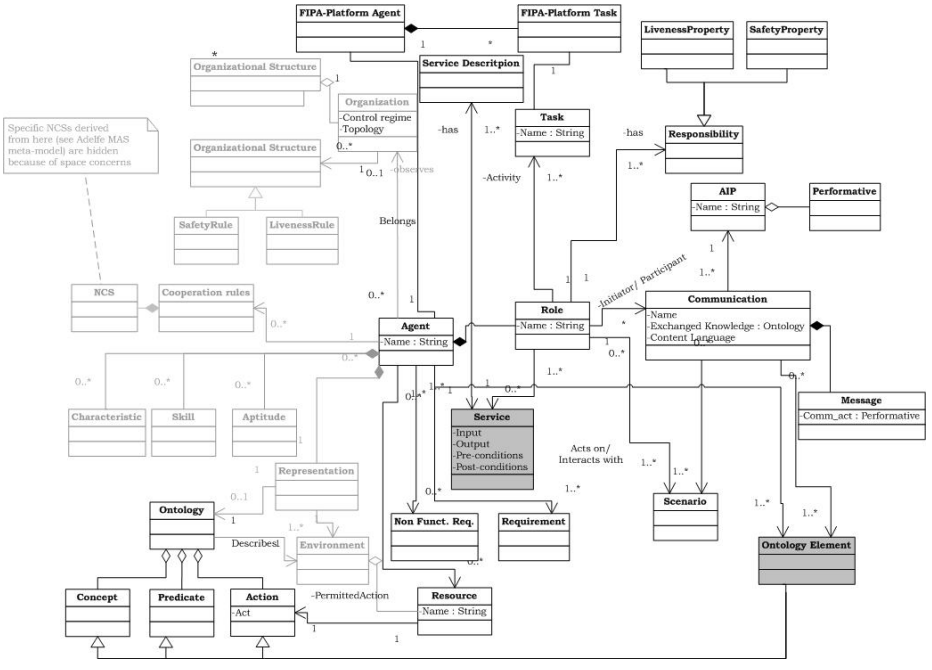

Fig. 4. The PASSI elements in the Unifying MAS meta-model

ated to each role. From all of this descends that a design process based on PASSI is performed through a sequence of steps leading from an early identification of agents to the definition of their roles and the description of their communications, while for instance in Gaia[15], being different the definitions of agent and role, we can see that the process is initially based on roles definition and only in a second time on the agents identification.

Instead in Agile PASSI, an agent is composed only by tasks, the concept of role is not necessary and so it is not present in Agile PASSI meta-model.

Agent Interaction: as regards the agent interaction capability in PASSI (and Agile PASSI) we see it is based on communications; they refer to an agent interaction protocol and the knowledge exchanged during the communication is seen as an instance of the domain ontology; this points out how important it is in PASSI to create a relationship between communications and the ontology through an *ontology element* while it is not necessary to introduce the concept of *environment* of Adelfe[16], where an agent can interact with another agent directly through communications or indirectly through the environment.

Agent Society: social relationships in PASSI are modeled through the definition of services that can be provided or accessed by agents. The service providing imply that agents would play social roles so to participate in scenarios interacting with other through communications; an agent can handle some resources that are relevant for the remaining part of society and accessing them can trigger

some kind of interactions. During the design flow in PASSI a static structural representation of the agent society is made through a class diagram where classes (agents) can be grouped in packages representing the social structures (groups, communities,...); differently in Gaia, the agent society is considered more than a collection of interacting agent but it is an entity with a well defined structure. From this structure a designer can identify agent activities, assigning a role for each social one; once all the roles that compose one agent are defined, their activities and responsibilities are converted into a set of services. Agent society is not modeled in Agile PASSI.

Agent Implementation: in PASSI and Agile PASSI a direct mapping exists between the elements of the MAS meta-model and their implementation; each agent is coded using the base agent class of the selected implementation platform and it contains the tasks that are used by roles (that have not a direct code level implementation). No similar mapping is provided by Gaia or Adelfe.

6 Conclusions

A large number of MAS design processes have been developed in the last years and probably others will be created in future; each of these is characterized by specific features characterizing the single approach. In all of these cases, differences among the various design processes (sometimes referred to as *methodologies*) reflect in correspondent differences among the MAS meta-models. In this work we presented the MAS meta-models of the PASSI and Agile PASSI design processes and compared them with a unifying proposal of MAS meta-model resulting from the study of three existing design process MMMs. Our aim was both to evaluate whether from the proposed unifying MMM we could derive a new design process (Agile PASSI) and to speculate about the fact that different processes, although similar in some parts of the MAS meta-models, can have very different approaches to the design of their systems (some examples dealt with the order in which the different elements are instantiated during the design time). We can conclude that the unifying proposal despite of the level of generality that it introduces still sufficiently supports the PASSI/Agile PASSI MAS meta-models and besides, in section 5, we observed that the unique schema representing a MMM is not sufficient to underline some specific design process; conversely, several different solutions can be drawn to instantiate the same meta-model; it still remains to explore the importance that the different definitions of the MMM elements can have in constraining the overall process.

References

1. Capera, D., Georg, J.P., Gleizes, M.P., Glize, P.: The amas theory for complex problem solving based on self-organizing cooperative agents. In: Proc. of the 1st International Workshop on Theory And Practice of Open Computational Systems (TAPOCS03@WETICE 2003), Linz (Austria) (2003)
2. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the tropos project. Inf. Syst. **27** (2002) 365–389

3. Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems **3** (2000) 285–315
4. Cossentino, M., Potts, C.: A case tool supported methodology for the design of multi-agent systems, Las Vegas (NV), USA, The 2002 International Conference on Software Engineering Research and Practice, SERP'02 (2002)
5. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent systems engineering. International Journal on Software Engineering and Knowledge Engineering (**11**) 231–258
6. Bernon, C., Cossentino, M., Gleizes, M., Turci, P., Zambonelli, F.: A study of some multi-agent meta-models. Lecture Notes in Computer Science **3382** (Jan 2005) 62 – 77
7. Cossentino, M.: From requirements to code with the passi methodology. In Henderson-Sellers, B., Giorgini, P., eds.: Agent-Oriented Methodologies, Idea Group Inc. (2005 (in printing))
8. Cossentino, M., Seidita, V.: Composition of a new process to meet agile needs using method engineering. In Ed., E., ed.: LNCS Series. (2004) 36–51
9. Chella, A., Cossentino, M., Sabatucci, L., Seidita, V.: From passi to agile passi : tailoring a design process to meet new needs. In: 2004 IEEE/WIC/ACM International Joint Conference on Intelligent Agent Technology (IAT-04), Beijing, China (2004)
10. Brinkkemper, S.: Method engineering: engineering the information systems development methods and tools. Information and Software Technology **37** (1995)
11. Kumar, K., Welke, R.: Methodology engineering: a proposal for situation-specific methodology construction. Challenges and Strategies for Research in Systems Development (1992) 257–269
12. Saeki, M.: Software specification & design methods and method engineering. International Journal of Software Engineering and Knowledge Engineering (1994)
13. Beck, K., al.M. Beedle, van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: (Agile manifesto) http://www.agilemanifesto.org.
14. Alliance, A.: (http://www.agilealliance.org)
15. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: the gaia methodology. ACM Transactions on Software Engineering and Methodology **12** (2003) 417–470
16. Bergenti, F., Gleizes, M.P., Zambonelli, F.: Methodologies and Software Engineering for Agent Systems. Kluwer (2004)