THE MAR&A METHODOLOGY TO DEVELOP AGENT SYSTEMS

Giacomo Cabri, Mariachiara Puviani and Letizia Leonardi
Dipartimento di Ingegneria dell'Informazione, Università di Modena e Reggio Emilia
Via Vignolese 905, 41100 Modena, Italy

Keywords: Methodology, Infrastructure, SPEM.

Abstract: In this paper we present a new agent methodology called MAR&A. Its aim is to better connect agent methodologies and agent infrastructures, since in the agent development we can find a "gap" between them. Our approach was not to build a new agent methodology *from scratch*, but to reuse "fragments" of existing methodologies. Besides presenting the methodology, we propose its use in a case study, to help readers understand the exploitation of this methodology and to sketch the connections with agent infrastructures.

1 INTRODUCTION

The agent paradigm has proved to be useful in the development of today's systems, in particular those that exhibit a certain degree of complexity (Jennings, 2001). Nevertheless, the support for developers shows some lacks and some fragmentations that decrease the help that can be concretely provided to developers. From the one hand, there exist appropriate methodologies that support in particular the analysis and the design of agent-based systems; on the other hand, the agent community has provided agent infrastructures to support the implementation of agent-based systems. The main fragmentation stems from the fact that these two opposite approaches (one *top-down* and the other *bottom-up*) hardly converge to a common point: the agent infrastructures often do not provide abstractions for the outcome of the methodologies, while methodologies fail in proposing artifacts suitable to be implemented by infrastructures. This leads to a "gap" between methodologies and infrastructures. Our aim is to overcome the previously explained limitations by bridging the gap existing between agent methodologies and infrastructures. To this purpose, we have evaluated different agent methodologies and infrastructures, in particular considering the connections that exist but are not so explicit. Exploiting *meta-models*, we were able to focus on the entities that are shared between methodologies and infrastructures. Of course, some entities are easy to connect (such as "agent"), while others require more work. Then, we have evaluated different possible directions, sketched in this paper, but we felt that they were not enough.

So, we decided to start building a new agent methodology, called MAR&A, that focuses on the main infrastructures' abstractions. To avoid re-inventing the wheel, we have not started *from scratch*, but we have decided to reuse existing "fragments" of methodologies; this is possible by means of the SPEM approach (Object Management Group, 2007), which decomposes agent methodologies into fragments that can be composed somehow.

The aim of this paper is to describe the work that has led to the definition of MAR&A and to present the main feature of the methodology, sketching its connections with infrastructures.

2 THE GAP

With regard to methodologies for developing agent systems, we have evaluated the most spread ones: *ADELFE* (Picard and Gleizes, 2004), *Gaia* (Zambonelli et al., 2003), *PASSI* (Cossentino et al., 2004), *Prometheus* (Winikoff and Padgham, 2004), SODA (Molesini et al., 2006) and *Tropos* (Bresciani et al., 2004).

With regard to the agent infrastructures, the following ones have been considered: CArtAgO (Common Artifact for Agent Open environment) (Ricci et al., 2007), *JACK* (AOS Autonomous Decition-Making Software, 2008), *JADE* (Bellifemine, 1999), *RoleX* (Cabri et al., 2003), *TOTA* (Mamei and Zambonelli, 2005), *TuCSoN* (Omicini and Zambonelli, 1999).

Cabri G., Puviani M. and Leonardi L. (2009).
THE MAR&A METHODOLOGY TO DEVELOP AGENT SYSTEMS .
In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 501-506
DOI: 10.5220/0001661005010506

We have considered the chance of integrating methodologies and infrastructures, evaluating possible matchings between the concepts described by their meta-models.

From our study emerges that there is no continuity between the methodologies and the infrastructures, presenting a gap between analysis and design on the one hand and implementation on the other hand.

In fact, only few entities can be found in both methodologies and infrastructures with the same meaning, while others are present either in methodologies or infrastructures. As an example, the *goal* entity is significantly considered by the methodologies, but cannot be directly found as an entity in infrastructures. Even if this is caused by a "natural" difference between methodologies and infrastructures, it could lead to some problems; first, resulting in a fragmented development; but, more important, making maintenance very difficult: if a methodology entity has not a corresponding infrastructure entity, its change requires to find out how such an entity was implemented.

The root of such a gap is likely to derive from the different origins of methodologies and infrastructures: from the one hand, the traditional software engineering approaches follow a *top-down* direction; on the other hand, concrete requirements have called for implementation-oriented solutions providing platforms and frameworks to build agent applications, in a *bottom-up* fashion.

3 TOWARD A NEW METHODOLOGY

To bridge agent methodologies and agent infrastructures our first effort tried to map the existing agent methodologies' entities with the existing agent infrastructures' ones. To this purpose, first of all we have evaluated the entities common to the different infrastructures. This was useful to extrapolate the "core" entities, which deserve a support by the methodologies. We spent an effort to use the meaning of the entities, in order to map also entities with different names but the same meaning. As a concrete example, we have considered the PASSI methodology and the RoleX infrastructure, and we have produced a mapping based on their meta-models (Cabri et al., 2008). Molesini et al. have performed a similar attempt (Molesini et al., 2006), considering one methodology (SODA) and three infrastructures (CArtAgO, TuCSoN and TOTA). As another attempt, we have focused on the *role* entity, and we have evaluated how it is dealt with in methodologies and in infrastructures, in order to map the different approaches (Puviani et al., 2008b).

The lesson learned from these experiences is that exact mappings (1 to 1) are not enough, and a more global approach is needed. This is due also to the fact that a lot of different methodologies and infrastructures exist, and point-to-point mappings would be complex and perhaps not so useful.

So, we decided to propose a new agent methodology, which has the infrastructures' main abstractions as goal for its outcome. Nevertheless, our aim was not to define a completely "new brand methodology" because it risks to be "yet another methodology". The trade-off we propose is to create a new methodology starting from the "fragments" of existing agent methodologies exploiting SPEM (Software Process Engineering Metamodel) (Object Management Group, 2007), an approach that decomposes the methodologies in "fragments", which can be assembled to form new methodologies.

From the evaluation of the infrastructures, we remark that three entities emerge as common: *Agent*, *Role* and *Action*. Such entities are not only common to different infrastructures, but also part of their foundation. This makes them the best candidates to be considered as the outcome of the new methodology; this also lead the new methodology being developed using different infrastructures.

4 THE MAR&A METHODOLOGY

In our study about the new methodology, we have focused on (i) the common processes and entities of the methodologies and (ii) the entities that enable a connection with the main entities of the infrastructures. MAR&A, "Methodology for Agent: Roles and Actions", has as main entities the same used in infrastructures (as said before), which helps developer to implement applications developed by this methodology. The chosen model is the waterfall one, and it relies on different important fragments of PASSI, Gaia and Tropos. We have defined three main phases of MAR&A: *Requirement* (Subsection 4.1), *Analysis* (Subsection 4.2) and *Design* (Subsection 4.3), while the *Implementation* phase is under definition. In the following we briefly explain the exploited fragments. Note that not all mentioned entities have a corresponding fragment in the figures.

4.1 Requirement Phase

In the *Domain Description Fragment*, taken from Passi, the concepts of *goal* and *actor* have been integrated, to create the Actor Diagram, that it is very useful in the *Analysis Fragment* taken from Tropos (Figure 1). Here some further work has to be done to convert Tropos Goal Diagram (output of *Analysis Fragment)*in a UML one.

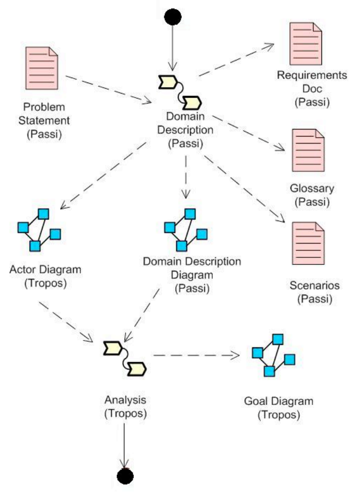

Figure 1: Requirement phase.

Very important in this phase is the creation of a glossary that will report all the created entities, and that has to be updated during all the other phases (and fragments). Here some important concepts are defined: **Requirement** - A feature that the System-to-Be (representing the general environment where other actors live) must exhibit, it can be functional or non-functional. **Scenario** - Represents a concrete sequence of interactions between the system and the actors. **Actors** - Entities that have strategic objectives (goals) and wishes, in the system. **Goals** - Actors' strategic interests. They can be divided in hard or soft goals, based on their fulfilment. **Plan** - Procedures that have to be executed to reach goals.

4.2 Analysis Phase

In the *Agent Identification Fragment*, taken from Passi, the Actor Diagram has been introduced, to help specifying Agent (Figure 2).Here we have to notice that the Requirement Doc, is the same as the Requirement Statement used in Gaia, so it can be integrated in a Gaia Fragment without modification.
In this phase, we have seen that the way of exploiting*Identify the Role in the System Fragment* from Passi and of *Roles Identification Fragment* from Gaia, is very similar, so we have mixed together these two fragments. The only thing that is still missing is a stronger connection between the Prototypical Role Model and the Role Identification Diagram, that now remain separated.

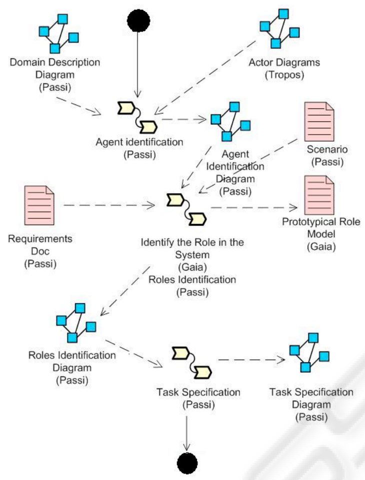
Domain Description Diagram (Passi)
Actor Diagrams (Tropos)
Agent identification (Passi)
Agent Identification Diagram (Passi)
Scenario (Passi)
Requirements Doc (Passi)
Identify the Role in the System (Gaia) Roles Identification (Passi)
Prototypical Role Model (Gaia)
Roles Identification Diagram (Passi)
Task Specification (Passi)
Task Specification Diagram (Passi)

Figure 2: Analysis phase.

In this phase the following concepts are identified: **Agent** - An autonomous entity that is composed of roles and has a knowledge. An agent can be seen from different levels of abstraction: it is a significant software unit at both the abstract and concrete levels of design. In this phase, an agent is specified as an instance of an agent class. An agent may undertake several functional roles during interactions with other agents to achieve its goals. Here we tried to model an agent that is independent of a specific platform, and that can be used in different infrastructures without a lot of changes. **Role** - A collection of tasks performed by agent in pursuing a sub-goal; an agent could play one or more roles in the system. Each role describes an aspect of agent life cycle and it is often related to a service offered by the agent to the society or to the achievement of one of its goals (Social Role). **Action** - The identification of activities that an agent may perform. Each action is composed of one or more tasks.

Task - A logical unit of individual or interactive behavior. An agent uses tasks to execute its plan(s). Each task is an entity that aims to reach a sub-goal. The term "task" can be used as synonymous of Behavior but with the meaning of atomic part of the overall agent behavior.

4.3 Design Phase

Here the *Create an Agent Model Fragment* (from Gaia) has been changed because it has to consider Passi-like agents (and not Gaia-like ones), and to be based on Single Agent Structure Diagram Figure 3).
The defined concepts are: **Ontology** - An ontology is composed of concepts, actions and predicates. **Communication** - An interaction between two agents, described in terms of: ontology (related to the part of knowledge exchanged by the agents), content lan-

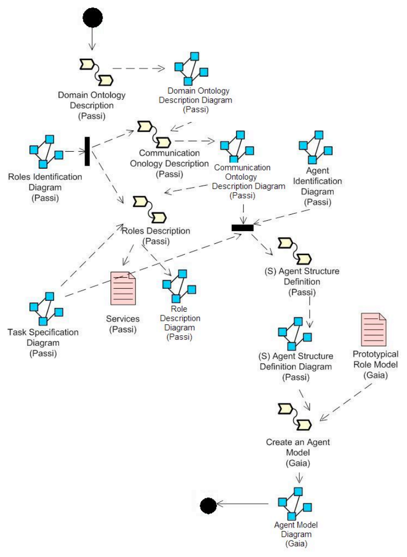
Domain Ontology Description (Passi)
Domain Ontology Description Diagram (Passi)
Roles Identification Diagram (Passi)
Communication Onology Description (Passi)
Communication Ontology Description Diagram (Passi)
Agent Identification Diagram (Passi)
Roles Description (Passi)
(S) Agent Structure Definition (Passi)
Task Specification Diagram (Passi)
Services (Passi)
Role Description Diagram (Passi)
(S) Agent Structure Definition Diagram (Passi)
Prototypical Role Model (Gaia)
Create an Agent Model (Gaia)
Agent Model Diagram (Gaia)

Figure 3: Design phase.

guage and interaction protocol. **Message** - An individual unit of communication between two or more agents. Messages are based on the standard FIPA message format.
Some of these different fragments are well integrated, even if they come from different methodologies, while other fragments have been changed inside to match input and output of the different phases. This work presents some difficulties because, as said, entities with the same name but coming from different methodologies cannot be used in the same way (e.g. the concept of "agent" in ADELFE is adaptive, in PASSI is FIPA-like and in Prometheus is a BDI one). In other cases, we have chosen the most suitable concepts for our scope (e.g., we do not have chosen ADELFE agent because it needs some concepts, like skill, aptitude, NCS, etc., which need an *ad hoc* platform to be implemented, and it is out of our scope). The implementation and validation of this composed methodology is still in progress, trying to adjust the different fragments, in particular in the Design phase.

4.4 Case Study

As case study we exploit a conference organization, which is a multiphase process that involves different kinds of people and groups, and can be easily implemented (and used) via Web. For length reasons, in this paper we report a summary of the application of MAR&A to the case study; interested readers can find a more complete description in (Puviani et al., 2008a).

Here, the Conference Organization System is briefly described: during the submission phase, authors send papers, and a submission number is given back to them when the paper is received. During the review phase, the Program Committee (PC), composed of a PC Chair and some PC Members, registers papers reviews: they will contact some reviewer, asking them to do the review for some papers. With the compiled review forms, the PC decides if the paper has to be accepted or rejected. During the editing phase, authors whose paper has been accepted, have to write a new version of their paper according to the suggestions of the reviewer. At the end, editors have to collect the reviewed paper and to print the conference proceedings. In the following the case study developed by MAR&A is presented.

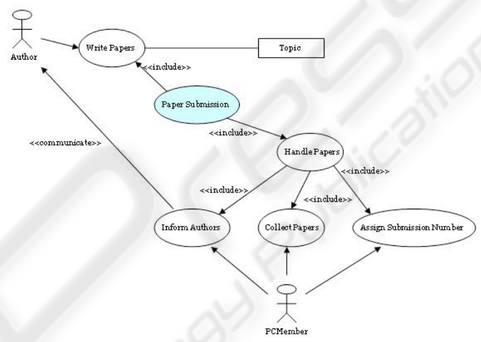

Figure 4: Domain Description Diagram.

In the *Requirement Phase*, system requirements are described in term of objectives (goals) and actors; the problem to solve is presented, along with its constrains and limits. Here we can see that the concept of Agent (as used in infrastructures) is introduced using the concept of Actor. In the Domain Description Fragment, the system's requirements are described using Use Case Diagrams: system goals and actors are specified along with their relations. Figure 4 reports an example of Review Phase Domain Description Diagram.

The following step identifies and models Actors as social actors that depend one to each other considering goals to reach, tasks to obtain, and resources to use. The Actor Diagram (Figure 5) underlines actors and the dependences between them. This phase is very relevant because it begins to create an interaction map, useful when we need to build this application using infrastructures, where usually interactions are very important and well defined. Another actor that has to be considered is the System-to-Be, and the Actor Diagram can be redefined with this actor in an easy way thanks to the adaptability of the methodology. Defining actors is very useful because, unlike agents, they are always platform independent, and so they can help developer to build agents in different infrastructures. Then Actor Diagram is extended with the relationship taken from systems goals.

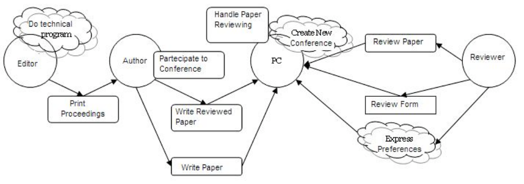

Figure 5: Actor Diagram.

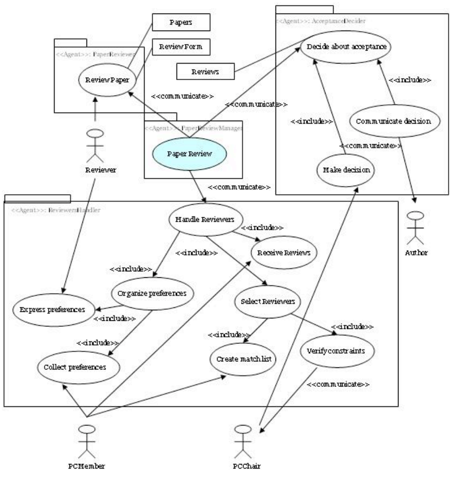

Figure 6: Agent Class.

In the *Analysis Phase* agents are defined as well as tasks and roles they play, and actions are introduced. In the Agent Identification Fragment, diagrams of the Requirement Phase are used to analyze actors: agents can be seen as single Use-Case or as a Use-Case package: every package defines agents' functionalities. In Figure 6 we can see an example of Agent Classes that are very useful to be used during agents implementation in infrastructures.

During the Identify Roles Fragment, a Prototypical Roles Model is defined: it presents roles along with their responsibilities (liveness and safety) and permissions, using Gaia notation. Then, using the Roles Identification Diagram (e.g. Figure 7), inter-agent communications are well described, defining every path of the Agent Identification Diagram. In the Task Specification Fragment, for each Agent, a Task Specification Diagram is defined: it contains agent's tasks on the right and interacting agents on the left. At the end of this phase, starting from Roles and Tasks, a preliminary list of Actions is created, to help developers deal with infrastructures.

During the *Design Phase*, in the Domain Ontology Description Fragment, some Domain Ontology Diagrams (Class Diagrams) that describe the used ontology in terms of concepts, predicate and actions are built. The Communication Ontology Diagrams Fragment presents agent interaction along with do-

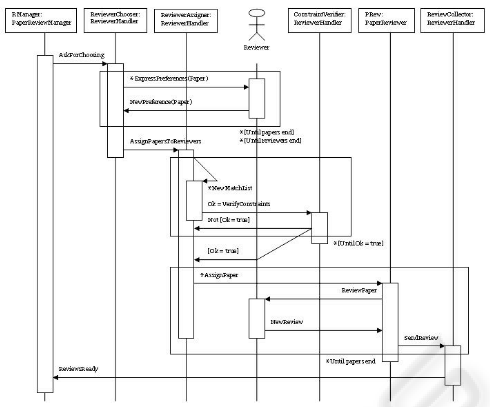

Figure 7: Role Identification Diagram.

main ontology. They are very important because can help developer map entities between the methodology and the chosen infrastructure. These diagrams further develop actions played by agents. Then, during the Roles Description Fragment, agents' life cycle is modeled, considering roles, collaborations, and conversations; the result is a Role Description Diagram, where each agent is identified by a package containing its roles' classes, and each role is obtained composing different tasks. In this latter diagram, connections between roles played by the same agent are reported and they can represent role changes.

During the Agent Structure Definition Fragment, which uses Agent Identification Diagram and the Communication Ontology Diagram as input, a Single-Agent Class Diagram is built (e.g. Figure 8), focused on the internal structure of each agent. At the end, during the Create an Agent Model Fragment, an Agent Model is defined with a set of roles that can be mapped by each agent class.

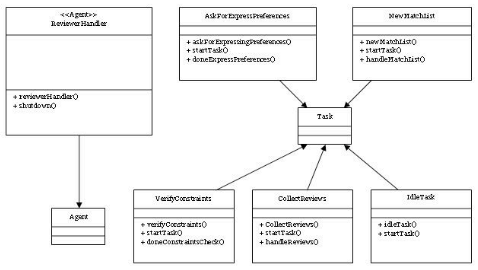

Figure 8: Single-Agent Structure Definition.

With the help of this case study, we can see that the composed methodology stressed out the entities relevant for infrastructure, that will help developers in system implementation using not an a-priori decided infrastructure. For instance, the implementers can chose the Jade and the RoleX platforms, and map the agent artifacts produced by MAR&A onto the agent abstraction of Jade; the MAR&A roles onto the RoleX role descriptors; and the MAR&A actions on the AgentAction Jade class or to the Action class of RoleX.

5 CONCLUSIONS

In this paper we have presented MAR&A, a methodology that aims at being infrastructure-oriented, in the sense that it is oriented to the main abstractions of the existing agent infrastructures, in order to propose a step toward bridging the gap between agent methodologies and agent infrastructures.

The two peculiar aspects of MAR&A are: (i) it has been built starting from the infrastructures' abstractions, so to make the passage from the methodology design phase to the infrastructure implementation phase easier; and (ii) it is not a completely new methodology, but it is "composed" from fragments of existing agent methodologies.

There are some advantages in exploiting fragments of existing methodologies. First, it is not "yet another methodology", because it takes concrete parts of the other methodologies, not only ideas and concepts; then, the exploited fragments have been tested by developer for different years and in different scenarios; further, there are already related documentation, case studies, tools and practice; moreover, fragments can be exchanged on the base of the application needs; finally, some developers can find some parts of the methodology they are used to.
With regard to future work, we are defining the Implementation Phase in details, to make it possible to produce code that can be used by an agent infrastructure. Of course the methodology must be tested also with other case studies, to better validate its usability. Another important thing that is still missing, but which we are working on, is the introduction of AMAS Adequacy from Adelfe, and a better specification of a Communication Protocol in the Design Phase.

ACKNOWLEDGEMENTS

Work supported by the Italian MiUR in the frame of the PRIN project MeNSA. The author thank Caterina Barbieri for her work in the development of the MAR&A methodology.

REFERENCES

AOS Autonomous Decition-Making Software (2008). Agent oriented software, jack agent platform. http://www.agent-software.com/.

Bellifemine, F. (1999). Developing multi-agent systems with jade. In *Proceedings of PAAM 99, London (UK)*, pages 97–108.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

Cabri, G., Ferrari, L., and Leonardi, L. (2003). Enabling mobile agents to dynamically assume roles. In *Proceedings of the ACM Symposium on Applied Computing, Melbourne (USA), March*, pages 56–60.

Cabri, G., Leonardi, L., and Puviani, M. (2008). Methodologies and Infrastructures for Agent Society Simulation: Mapping PASSI and RoleX. In *Proceedings of the 19th EMCSR, Wien, March 2008*.

Cossentino, M., Sabatucci, L., and Chella, A. (2004). Patterns Reuse in the PASSI Methodology. *LNCS*, pages 294–310.

Jennings, N. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41.

Mamei, F. and Zambonelli, F. (2005). Programming stigmergic coordination with the TOTA middleware. In *Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems, New York, USA*, pages 415–422.

Molesini, A., Omicini, A., Denti, E., and Ricci, A. (2006). SODA: A roadmap to artefacts. *Engineering Societies in the Agents World VI*, 3963:49–62.

Object Management Group (2007). SPEM.http://www.omg .org/technology/documents/formal/spem.htm.

Omicini, A. and Zambonelli, F. (1999). Coordination for internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269.

Picard, G. and Gleizes, M. (2004). The ADELFE Methodology–Designing Adaptive Cooperative Multi-Agent Systems. *Methodologies and Software Engineering for Agent Systems. Kluwer Publishing*.

Puviani, M., Barbieri, C., Cabri, G., and Leonardi, L. (2008a). A Case Study for MAR&A. Technical Report DII-AG-2008-1, DII, University of Modena and Reggio Emilia.

Puviani, M., Cabri, G., and Leonardi, L. (2008b). Agent Roles: from Methodologies to Infrastructures. In *Proceedings of CTS 2008, Irvine, USA, May 2008*.

Ricci, A., Viroli, M., and Omicini, A. (2007). CArtAgO: A framework for prototyping artifact-based environments in MAS. In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for MultiAgent Systems*, volume 4389 of *LNAI*, pages 67–86. Springer.

Winikoff, M. and Padgham, L. (2004). Developing Intelligent Agent Systems: A Practical Guide.

Zambonelli, F., Jennings, N., and Wooldridge, M. (2003). Developing multiagent systems: The Gaia methodology. *ACM TOSEM*, 12(3):317–370.