Different Perspectives in Designing Multi-Agent Systems

Massimo Cossentino

ICAR/CNR – Istituto di Calcolo e Reti ad Alte Prestazioni/ Consiglio Nazionale delle Ricerche
c/o CUC, Viale delle Scienze, 90128 Palermo, Italy
cossentino@cere.pa.cnr.it

Abstract. The process of designing a multi-agent system (MAS) naturally involves many different concepts. The system should reach some goals and provide specific requirements. The designer should produce an accurate model for the domain ontology, the agents' knowledge and interactions. The structural and behavioral description of the system will descend from these elements and will also generate some constraints in the other parts of the design.
From these considerations and our experiences in designing MAS with a CASE tool supported methodology (PASSI) we deducted the importance of exploring the design process of these systems from several different perspectives that explicitly address the architectural, social, knowledge, resource and physical aspects of them.

1. INTRODUCTION

During the design of a MAS (multi-agent system), the designer passes through several different levels of abstraction looking at the problem from many different points of view. The result is a series of models that reflect this multi-faced structure.

As a natural consequence, the process used in affording the different models should not be planar but multi-dimensional. Several different perspectives can therefore be identified in this process. They address different aspects of the agent solution that is more complex than a traditional object oriented one because of its social and ontology aspects, of the necessity of surveillance on the rules observance, and of trust problems connected to a potentially opened system.

In testing the PASSI (Process for Agent Societies Specification and Implementation) [7] design methodology, we observed that the mental process of the designer does not only follow the sequence of steps prescribed, it browses around analyzing different aspects of the design (we will refer to the resulting point of view as a perspective) and different connections among the elements (we will refer to them as influences); the sequence of steps is only the sequential order in which he will perform the formal acts and will produce the diagrams.

The PASSI methodology takes great advantage from the use of PTK (PASSI ToolKit), a dedicated design tool (integrated in Rational Rose) that we have realized.

Several UML design tools exist and some attempts have been done in order to support the designer not only in the pure action of designing (like a simple graphic tool can do) but also in order to introduce some level of cognitive support. Regarding this experiences we have the advantage of focusing our work on a specific design process and therefore the possible activities of the designer are known and can be studied. We looked at the methodology from very different points of view trying to explore the possible relationships between the different elements of it in terms of chronological order of production, flow of information, focus of attention (decomposition of the functionalities, social aspects, implementation of the solution, …).

From this study we learnt some lessons that will be useful in increasing the functionalities of our Rational Rose add-in and better the productivity of the whole process.

2. The PASSI design methodology

In this section, we will briefly introduce the elements of the PASSI methodology [7]. It is composed of five models (Figure 1) and twelve phases that address different design concerns in the process of building a model. Although these numbers could seem high, they reflect a natural vocation of the process for a multi-facet approach while the support offered by PTK (the Rational Rose add-in) drastically lowers the effort of the designer automatically producing several parts of the whole model.

In PASSI we use UML as a modeling language and in order to support some specific issues related to the nature of the MAS we also use AUML [8]. We chose of restricting the implementation environment for our multi-agent systems to FIPA-based platforms and from this choice we obtained the possibility of fully supporting the most detailed, coding-related steps of the design process.

The process is iterative and requirements-to-code; the twelve phases (each of them producing one or more UML diagrams) are:

- *Domain Description (D.D.)*. A functional description of the system using conventional use-case diagrams. We describe the requirements with use-case diagrams that are obtained either through common requirements elicitation typical

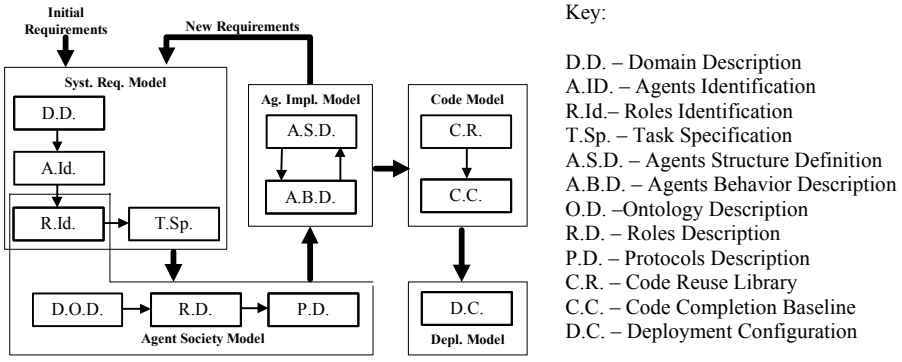

Fig. 1. The models and phases of the PASSI methodology

of the object-oriented methods [10] or through the informal application of a scenario-based teleological method such as GBRAM [11] or ScenIC [12].
The result is a functional description of the system through a hierarchical series of use-case diagrams.

- *Agent Identification (A.Id.)*. Agent identification starts from the use-case diagrams of the previous step. According to our definition of agents, it is possible to see an agent as a use case or package of use cases in the most detailed diagram of the D.D. phase.
- *Role Identification (R.Id.)*. Role identification produces a set of sequence diagrams that specify the most important scenarios from the agents' identification use case diagram. These diagrams are used to identify the roles played by the agents to achieve their goals.
- *Task Specification (T.Sp.)*. In the Task Specification phase we look at one agent at a time, drawing an activity diagram for each agent. In this diagram, each activity node represents a task that the agent can perform. Therefore T.Sp. diagram resumes the behavior of the agent also including its interactions with the others. It completes the requirements model of the system.
- *Ontology Description (O.D.)*. We use class diagrams and OCL constraints to describe the ontology of the domain, the knowledge ascribed to individual agents and the pragmatics of their interactions. In order to better describe the details of these elements, we often use two different diagrams, the *Domain Ontology Description* and the *Communication Ontology Description,* to describe the overall ontology and its use as part of the agent's knowledge and communications.
- *Role Description (R.D.)*. With a class diagram we show the distinct roles played by agents, the tasks involved, the communication capabilities and the inter-agent dependencies (service, resource, … [15]). The scope of this phase is modeling the lifecycle of each agent, looking at the roles it can play, at the collaboration that it needs, and the communications in which it participates. The R.D. diagram is also where we introduce all the rules of the society (organizational rules, [13]), laws of the society and the domain in which the agent operates (e.g. trade laws) and the behavioral laws considered by Jennings in his "social level" [3].
- *Protocol Description (P.D.)*. Sequence diagrams specify the grammar of each pragmatic communication protocol in terms of speech-act performatives. We adopt the representation of the AIP (Agent Interaction Protocol) proposed by Odell as an agent oriented extension (AUML) to the standard of UML [8].
- *Agent Structure Definition (A.S.D.)*. This phase is composed of several class diagrams. The diagrams logically belong to two different levels of detail in the design: the multi-agent and the single-agent. In the first we focus on the general architecture of the system and therefore in it we can find agents and their tasks. In the second level we look at each agent internal structure, showing all the attributes and methods of the agent class and of its internal task classes.
- *Agent Behavior Description (A.B.D.)*. Activity/state diagrams describe the behavior of individual agents. As in the case of the A.S.D. phase we have two different levels of abstraction in this phase. In the multi-agent diagrams we represent the flow of events, the invoked methods and the messages exchanged between agents. In the single-agent diagrams we detail the implementation of the formerly used methods.

- *Code Reuse (C.R.)*. In this phase we try to reuse existing patterns of agents and tasks. It is not correct to talk of patterns of code only, because the process of reuse takes place in the design of the system. Our patterns therefore are not only pieces of code. They are also pieces of design (of agents and tasks) that can be reused to implement new systems. Our add-in for Rational Rose provides an extended support for this phase and thanks to a quickly growing up repository the designer can often reuse large parts of the design/code in a fast and affordable process.
- *Code Completion (C.C.)*. The programmer completes the code of the application starting from the design, the skeleton produced and the patterns reused.
- *Deployment Configuration (D.C.)*. We use deployment diagrams to describe the allocation of agents to the available processing units and any constraints on migration and mobility. Some extensions to the standard UML notation are provided in order to support agents' mobility.

The PASSI methodology is general and can be applied in many different agent implementation environments. We extensively tested it using a standard FIPA architecture [9] in designing informative [16] and robotics systems [17].

3. The multidimensional perspective in the MAS design process

We identified 5 different perspectives in analyzing the system design. Two of them (knowledge and computer) come from a Newell's classification [1] (indeed, our computer perspective, is a little different) later expanded by Jennings [3] with the social level that we adopt as the third perspective, the other two (architectural and resource) come from classical software engineering concepts.

We think that looking at a MAS from different perspective does not result in a series of different not-related subsystems or partial descriptions of the whole system. The real outcome is a more detailed description of it in terms of a well defined aspect.

When the designer looks at the system in order to study some specific problem he thinks about it as 'something' of specific. He can be concerned about the distribution of the software in the available hardware platforms in order to optimize the performances or can be interested in defining the rules of interaction of the agent society. For this reason we intentionally call these different points of view 'perspectives' and not 'levels' or 'abstractions' because we want to stress the concept that the perspective is the representation of the system when the spectator is interested only in a specific conceptual area of the multi-faced agent system.

The architectural perspective looks at the software as a set of functionalities to be implemented in a classical, software engineering approach. It is the more abstract perspective. Its elements exist in the mind of the designer, they are abstractions of the system representing the functionalities and their logical implementation.

The social perspective is characteristics of the multi-agent systems. Many authors use some kind of social description in their approaches. It is interested in the society of agents that interact to reach the goals imposed by the designer.

The knowledge perspective is a high detailed point of view. The single agent, its functional and behavioral details (that induce some specific implementation) are the focus of this perspective.

The resource perspective is oriented towards the reuse of patterns of agents and their tasks. It represents some kind of mental bottom-up process, the process of recycling an existing agent and eventually adapting it to match the necessity of a new problem.

The computer perspective is the more physical, touchable point of view. It relates to the spreading of the files that constitute the software in the available hardware platforms; it therefore consider the problems arising from the hardware and their reflections on the software and deployment aspects of the system.

A classification of our perspective can be done using a biological metaphor: the resource and computer perspectives can be thought as physical representations of the system, the knowledge and social perspectives reflects psychological aspects of the MAS, the architectural perspective is some kind of theological justification for the others, it can be seen as the glue and motivation for the remaining four.

In the following sections we will discuss the details of the different perspectives. Each of them will be discussed using the approach of Newell in [1]. He used the concepts of: *medium* (what is to be processed), *components* (elementary parts of the level that provide primitive processing), *laws of behavior* (how system behavior depends on the component behavior and the structure of the system) and *laws of composition* (how components can be assembled) as aspects of the levels that compose a computer system.

In our work we refer to the concept of perspective instead of level because we want to emphasize the union of the system thought as a representation of the problem-solution couple that evolves from the early stages of the requirements elicitation to the final coding and deployment activities. The system is represented in the sequence of the models and phases of the PASSI methodology with their resulting artifacts. Looking at this unit with different scopes we obtain a perspective of it that shows some elements (under one of their possible facades) hiding what is out of the particular focus.

In the following we will describe each perspective in its general terms and then in relation to the PASSI methodology. In this latter step we will examine the relationships between the diagrams of PASSI arising from the application of each perspective. Then we will introduce these relationships in the schema of the methodology shown in fig. 1 in order to permit a simple comparison of the diagrams and relationships involved in the different perspectives.

3.1 The Architectural Perspective

It is the more conventional, less artificial intelligence perspective. It looks at the system only as a software. Its aim is to produce a good architecture in order to prepare a simple, well-detailed coding activity. This kind of attention is present in papers on multi-agent system design coming mainly from object-oriented experiences like that of DeLoach [2]. The functionalities of the system are fragmented in pieces that are small enough to be explored, captured in their meaning and relationships and therefore implemented in the different parts of the system.

3.1.1 Elements of the perspective

The focus of this perspective is the requirements reflection in the architecture of the system. The main components of this perspective are the functionalities required from the system and the agents that will satisfy these requirements. As a direct consequence of the specific focus, also the agents' roles and tasks (together with their structural/behavioral description) are seen in this perspective.

If we look at the components listed above, we can see that some compositional laws derive from the capabilities of the elements; for example, different roles can be played by one agent (providing a specific service in a moment and asking for another later) or the same role can be played by different agents, (the bidder role can be played by different types of agents in an auction). Other compositional laws come from the requirements of the system. Among these we have the interactions with the external world (that could require agents with sensing/actuating capabilities to be present in the interface between the system and the real world) or the design strategies (the designer could choose if adopting many different agents to perform a few tasks or a few agents performing many tasks).

The behaviors laws are derived from the necessity of satisfying the requirements. The components of this perspective share the goal of realizing all the functionalities as prescribed by these laws in order to obtain the desired external system behavior.

3.1.2 The architectural perspective in PASSI

The first phase that we can find in this perspective is the Domain Description that is composed of use case diagrams and therefore is oriented towards the functionalities. The next one is the A.Id. in which we bind some functionalities with a specific agent and, because of the agent presence in our scope, we should also consider the Role identification diagrams where we illustrate the scenarios in which the required functionalities will be satisfied and the Task Specification diagram that is the classification of the elementary pieces of behavior used to accomplish the requirements assigned to each agent. From the previous phases we obtain the information needed to design the (M)ASD diagram (a structural description of the agents) and the (M)ABD diagram (an activity diagram that is a picture of the way agents will act to reach their objectives and to accomplish the assigned functionalities interacting each other when necessary).

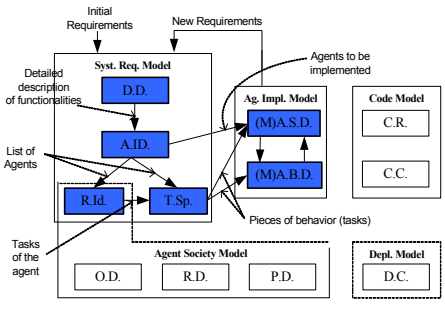

Fig. 2. The architectural perspective in the PASSI methodology

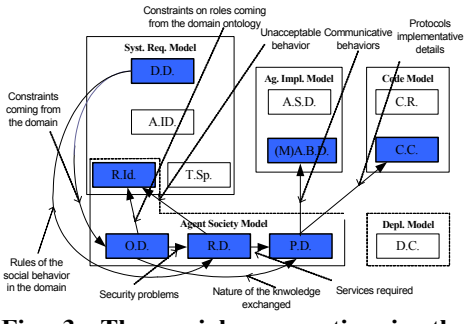

Fig. 3. The social perspective in the PASSI methodology

The listed diagrams are related in several ways (see Figure 2): from the D.D. phase the details of the functionalities arrive to the A.ID. phase where they are assigned to specific agents. This list of agents is used in the R.Id. diagram where the agents description are refined with the attribution of their roles. From the description of roles (performed with the sequence diagram of the R.Id. phase) we deduct the information needed to draw the task specification. The list of identified agents and of the tasks are part of the (M)A.S.D. phase where each agent is shown with its tasks. The same list of tasks determine the number of swimlanes of the activity diagram of the (M)A.B.D. phase and the scenarios of the R.Id. phase are used as guidelines to complete this diagram.

3.2 The Social Perspective

The social perspective is focused on the role of the agent society in achieving the solution to the requirements (through the collaboration among the agents) .

From the domain description we can obtain implications for the roles of the system; this approach is present in many MAS design methodologies [4][2].

3.2.1 Elements of the perspective

The components of this perspective (as expressed by Jennings in [3]) are the agents that constitute the organization, the communication channels and the organizational relationships needed to make them collaborate. The importance of these relationships has also been reported by Zambonelli et al. that in [13] deepened the organizational aspects of the Gaia methodology [4].

The organizational rules coming from the structure of the specific domain and from the agents' interactions, roles and role changes can be regarded as the compositional laws; in the behaviors laws we should consider the society actual behavior, the social responsibilities of the agents and the possibility for them of infringe rules. In this context the medium is the way that the agents can use to influence other's behavior (negotiation techniques, cooperation protocols, request of role changes, …)

3.2.2 The social perspective in PASSI

The agent society model is entirely part of this perspective. The ontology diagrams (with the domain ontology and communication ontology aspect), together with the phases related to the roles (R.Id. and R.D.) and the description of the protocols used in the communications (P.D. phase) describe the system from its social point of view. Important constraints on the specification present in these diagrams come from the analysis of the domain in which the system will operate and therefore from the D.D. phase (see Figure 3). In fact, from the analysis of the domain the designer obtains information that are useful to introduce constraints in the O.D. phase (for example one agent, exposed to external attacks, should not store a specific information because of privacy concerns) and in the R.D. phase (an agent who has win an auction should pay for the goods). These specifications, often put in form of a constraint, have effect also in the R.Id. phase sometimes changing the supposed interaction between the agents.

The P.D. phase is conditioned by the O.D. and the R.D. for the influence that the nature of data exchanged and the service required have on the choice of the protocols.

3.3 The Knowledge Perspective

The focus of this perspective is the agent seen as an asocial problem solver [1], this means that an agent is conceived in terms of the goals it has to achieve and the elementary actions that it can perform in their pursuit.

3.3.1 Elements of the perspective

This asocial agent is the main component of the knowledge perspective, together with its goals and elementary actions. The agent can divide or group its goals in order to decompose the problems and can compose its elementary actions to adapt its behavior to the resulting situation.

The obvious behavior law is the principle of rationality that simply states that if an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action (see Newell [1]). The mediums used by the agent to solve the problem are the knowledge and the elementary actions that can be assembled to give its behavior.

3.3.2 The knowledge perspective in PASSI

In this perspective the agent has some goals to achieve using the repository of its available behaviors and its knowledge. The description of the agent's capability is present in the Task Specification diagram; this information can be used to describe the behavior of the agents and to identify its roles.

In the Ontology Description (O.D.) diagrams we have a description of the knowledge of the agent and in the R.Id. and R.D. phase we have a picture of how the agent assemble its tasks to achieve the goals

The A.S.D. and the A.B.D. phases in this perspective assume a significance that is different from the architectural one. They are now scoped on describing the 'asocial agent' and therefore they are single agent diagrams: (S)A.S.D. and (S)A.B.D.

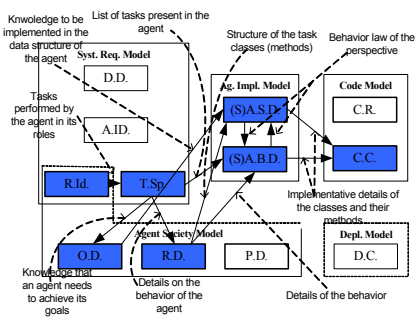

Fig. 4. The knowledge perspective in the PASSI methodology

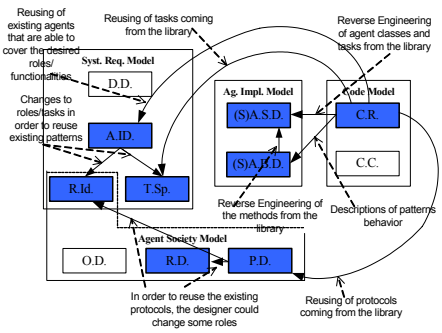

Fig. 5. The resource perspective in the PASSI methodology

The C.C. phase is where the ideas about the agent behavior and structure are concretized into actual code.

Several influences relate the phases of this perspective (see Figure 4). The list of the tasks needed to perform the interactions described in the R.Id. diagrams is necessary to carry on the T.Sp. phase; the tasks description gives a picture of the agent in terms of knowledge required and behaviors available and this information is useful in drawing the O.D. and R.D. diagrams.

The knowledge of the agent is described in the Ontology Description (O.D.) diagrams therefore we can draw a connection starting from it and ending in the A.S.D. diagram where each agent is described also in terms of its data structures. From the R.D. phase several information are derived for the structure of the agent ((S)A.S.D. phase) and the details of its behavior ((S)A.B.D. phase).

The iteration between the (S)A.S.D. diagram and the (S)A.B.D. diagram descends from the behavior law of this perspective. In fact if an agent is implemented in such a way that it has the knowledge that it is convenient to do something then it will actuate the consequent behavior (arrow from A.S.D. to A.B.D). This opportunity depends on the knowledge (awareness of the situation) and the structural behavior description (ability to act in that situation, arrow from ABD to ASD). The details of the classes and of their methods coming from these two last diagrams are then used to actually code them.

3.4 The Resource Perspective

This perspective is focused on the mental process of the designer who tries to reuse existing patterns of agents and tasks. It is useful to underline that in our view, a pattern is not only a piece of code but it is described using some diagrams of the same type of those used in the design of the MAS. As a consequence it influences several aspects of the design process.

We think that patterns will produce even more interesting results in the agent-oriented systems than they did in the object-oriented ones because of the higher encapsulation and decoupling of MAS.

In our opinion patterns should be essentially thought as pieces of structural and behavioral diagrams that can be introduced in the developing system design in order to complete it. The code becomes therefore only a consequence of the design and not the justification of it.

3.4.1 Elements of the perspective

The patterns (composed of code and description diagrams) are the obvious components in this perspective. They are composed using the interfaces and interactions specified. For example in a pattern of a task devoted to deal with an incoming communication using a specified protocol, the initiator of the conversation interacts with the pattern in some ways that specified in the pattern's diagrams and are therefore well defined.

The behaviors laws come from the specification of the behavior of the patterns. In our approach we detail each pattern with a class diagram representing its structure (the (S)A.S.D.) and an activity diagram (the (M)A.B.D.) that describes the behavior.

The mediums used are events and common methods invocation. Usually in fact the elements of the patterns interact using methods invocation within the same agent while events (for example arising from the dispatch of a message) characterize external agents interactions

3.4.2 The resource perspective in PASSI

The center of the resource perspective in PASSI is the C.R. phase. In this step designers identify some patterns that can be profitably applied to their system. These have a direct effect on the implementation model of the agent at the single agent level of detail (phases (S)A.S.D. and (S)A.B.D.) where the specification about the agent's behavior and knowledge are implemented. The A.Id. phase (figure 5) appears in this perspective because it specifies the functionalities bound to an agent that can be taken out from the repository. Eventually, changes in the functionalities of the agent can reflect on its behavior (R.Id., T.Sp., R.D. phases) and communications (P.D. phase).

If the pattern involves some communication the protocols described in the P.D. phase can be effected and these changes could propagate to the R.D. and R.Id. phases.

PTK, our Rational Rose add-in, includes a pattern repository that allows a simple reuse of existing pieces of design and related code. This implementation is fully in accordance with the above arguments and is one of the key features of PASSI.

3.5 The Computer Perspective

The real, tangible implementation of the system is the focus of this perspective. It is interested in how the files that contain the code are related and how the agents are deployed and move in the existing platforms.

3.5.1 Elements of the perspective

This perspective is composed of the software components (executables, libraries, …) and the hardware configuration. These elements are composed through the dependency relationships that exist among the elements of the software components (file dependencies) and hardware requirements (for example hardware connection availability that permits communications among agents).

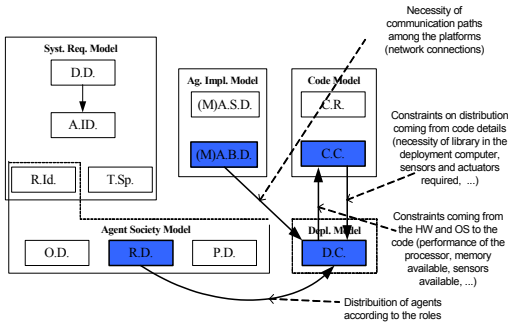

Fig. 6. The Computer perspective in the PASSI methodology

The presence of specific hardware devices (sensors, actuators) can also effects the aggregative behavior of these components (an agent who needs a camera cannot move to a computer without this device connected). The communication links among units of the system as expressed in the D.C. diagram and the executable environment (hardware, software, operating system) configuration are the mediums of this perspective.

3.5.2 The computer perspective in PASSI

The elements of PASSI that appear in this perspective are: the D.C. diagram for its obvious role in describing the position of the components in the execution environment, the C.C. phase where many code-related dependencies (inheritance, visibility, …) are introduced, the (M)A.B.D. phase where the mutual interaction between the agents are related to the agent's implementation and the R.D. phase where some issues about the spreading of the agents in the world could arise.

The deployment of the agents (D.C. phase, Figure 6) is strictly related to the code of the application and therefore to the C.C. phase: code constrains the distribution and presence of some files in the executing units (arrow from C.C. to D.C. in Figure 6) while the hardware configuration (performance of the processor, memory available, sensor devices, …) and operating system affect the design of the code (arrow from D.C. to C.C.). Other influences to the deployment come from the (M)A.B.D. phase in terms of requirements of communication links and from the R.D. phase about the location of the agents (for example, a scooter agent could need to visit a specific server in order to mine information from it).

4. Conclusions and future works

Several studies exist in literature about the possibility of approaching the design of a software system from different point of views. Starting from these researches, some UML CASE tools introduce various representations of the system model (for example in Poseidon we have several views centered on different aspects: diagram-centric, package-centric, …). In designing our MAS using PASSI and a specific design tool that we have produced, we felt the importance of the availability of a multi-perspective support. The first step in introducing these functionalities in our tool, is described in this paper and it consists of the identification and description (with regards to PASSI) of the most important possible views for a MAS model. We identified five useful representations (we refer to them as perspectives): Architectural (a structural representation of the software), Social (a look at the agent society involved in the solution), Knowledge (the analysis of each single agent), Computer (a representation of the physical solution), Resource (the study for reusing existing elements).

We are now working on the introduction of these five perspectives in the add-in for Rational Rose (PTK - Passi ToolKit) that we use to design with PASSI. These phase implicates the further exploration of the flow of information in the design process and its representation in the tool.

Some interesting results could come from this work like, for example, the introduction in PTK of a specific support for analyzing the side-effects of modifications in the design. If a designer changes something in a diagram (that is part of a specific perspective) then he needs to evaluate the consequences of this change in the other parts of the work. These effects are described by the relationships between the diagrams in the specific perspective. We think that some of these consistency checks and related changes can be automatically performed or at least a strong support can be introduced for them.

REFERENCES

1. Newell, A. The knowledge level, Artificial Intelligence, 18 (1982) 87–127.
2. DeLoach, S.A., Wood, M.F., and Sparkman, C.H. Multiagent Systems Engineering. International Journal on Software Engineering and Knowledge Engineering 11, 3, 231-258.
3. Jennings N.R., On agent-based software engineering, Artificial Intelligence 117 (2000), 277-296
4. Wooldridge, M., Jennings, N.R., and Kinny, D. The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems. 3,3 (2000), 285-312.
5. Aridor, Y., and Lange, D. B. Agent Design Patterns: Elements of Agent Application Design. In Proc. of the Second International Conference on Autonomous Agents (Minneapolis, May 1998), 108–115.
6. Kendall, E. A., Krishna, P. V. M., Pathak C. V. and Suresh, C. B. Patterns of intelligent and mobile agents. In Proc. Of the Second International Conference on Autonomous Agents, (Minneapolis, May 1998), 92–99.
7. Cossentino, M., Potts, C.: A CASE tool supported methodology for the design of multi-agent systems. Proc. of the 2002 International Conference on Software Engineering Research and Practice (SERP'02). Las Vegas, NV, USA, June 2002
8. Odell, J., Van Dyke Parunak, H., and Bauer, B. Extending UML for Agents. AOIS Workshop at AAAI 2000 (Austin, Texas, July 2000).
9. O'Brien P., and Nicol R. FIPA - Towards a Standard for Software Agents. BT Technology Journal, 16,3(1998),51-59.
10. Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley (1992).
11. Antón, A.I., and Potts, C. The Use of Goals to Surface Requirements for Evolving Systems, in proc. of International Conference on Software Engineering (ICSE '98), (Kyoto, Japan, April 1998), 157-166
12. Potts, C. ScenIC: A Strategy for Inquiry-Driven Requirements Determination in proc. of IEEE Fourth International Symposium on Requirements Engineering (RE'99), (Limerick, Ireland, June 1999), 58-65.
13. F. Zambonelli, N. Jennings, M. Wooldridge. Organizational Rules as an Abstraction for the Analysis and Design of Multi-agent Systems. Journal of Knowledge and Software Engineering, 2001, 11, 3, 303-328.
14. J.Odell, H. Van Dyke Parunak, B. Bauer. Representing Agent Interaction Protocols in UML, Agent-Oriented Software Engineering, P. Ciancarini and M. Wooldridge eds., Springer-Verlag, Berlin (2001), 121–140.

15. Yu, E., Liu, L. Modelling Trust in the i* Strategic Actors Framework. Proc. of the 3rd Workshop on Deception, Fraud and Trust in Agent Societies at Agents2000 (Barcelona, Catalonia, Spain, June 2000).
16. Burrafato, P., Cossentino, M.: Designing a multi-agent solution for a bookstore with the PASSI methodology. Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002). May 2002, Toronto, Ontario, Canada at CAiSE'02
17. Chella, A., Cossentino, M., Pirrone, R., Ruisi, A.: Modeling Ontologies for Robotic Environments. Proc. of the Fourteenth International Conference on Software Engineering and Knowledge Engineering. Ischia, Italy, July 2002