Designing a Problem Specific Design Process for Multi-Agent Systems

Massimo Cossentino
Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
Italian National Research Council (CNR)
Viale delle Scienze, 90128 - Palermo (Italy)
cossentino@pa.icar.cnr.it

Antonio Chella
Dipartimento di Ingegneria Informatica
University of Palermo
and Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
Italian National Research Council (CNR)
Viale delle Scienze, 90128 - Palermo (Italy)
chella@unipa.it

Abstract

Starting from the point that several design methodologies have been conceived to design multi-agent systems and nonetheless designers often prefer to create a new methodology instead of studying and applying the existing ones, we propose an extension to agents of the method engineering process that has been successfully used in the object-oriented context to easily compose new design process. The work reports a detailed production process that allows the composition of new methodologies by reusing parts of existing ones.

Introduction

Our work starts from the consideration that almost twenty different design methodologies can be found in literature for MASs (Multi-Agent Systems). We think that this is the unquestionable prove that agents designer (just like objects designer) in accomplishing their different tasks, and solving specific problems in distinct production environments, often prefer to setup an own methodology specifically tailored for their needs instead of reuse existing ones. What seems to be widely accepted is that an unique specific methodology cannot be general enough to be useful to everyone without some level of personalization.

In this scenario we can identify two contrasting elements: first, in an interesting paper on object-oriented software (development) processes, Fuggetta (Fuggetta 2000) states that the research in this field is stuck and most technologies developed by the software process community have not been adopted by the industrial world. Second, the AgentLink community, in its roadmap (Luck, McBurney, & Preist 2003) for agent based computing, embodying the feelings of a large part of the agent research and industrial community, has identified the designation of a standard in design methodologies as an essential demand.

In order to accomplish this request without neglecting the important warnings coming from the OO world we want to propose a quite open approach that allows the composition of a very large repository of human experiences (design process is first of all an human process) that could be expressed in terms of a standard notation (we are trying to refer to existing standards from OO and in case extend them).

We believe that agent-based systems are by themselves one of the aspects of the OO crisis solution and we are also strongly persuaded that the future standard in design methodologies will be a significant improvement in the agent research and industrial applications. The benefits will not only be limited to the dimension of the MAS that could be managed (this is not a secondary aspect if we think about the growing dimension of agent societies) but they will also be related to the possibility of integrating existing (even not agent-based) systems with new features on the fly (Zambonelli & Parunak 2002).

We think that the best solution that could be pursued to the discussed situation consists in dismissing the existing stringent design methodologies and allow the designer to rapidly construct a new methodology that could fit their specific needs. This implies that each methodology will be the result of a specific problem, developing environment (involved stakeholders with their skills) and the chosen structure of the system to be built (agents could be intelligent or purely reactive, mobile or node-constrained, deterministic or not and so on)

The approach that we describe in this paper is strongly related to the activity that the authors are carrying on within the FIPA Methodology Technical where several researchers and companies are trying to create common standards for the creation of an unique database of pieces of design methodologies that could be used by everyone to prepare his own design process.

The design process construction

In order to take advantage of the experiences done with existing methodologies we will adopt the method engineering paradigm(Brinkkemper 1995)(Kumar & Welke 1992). According to this approach, the development methodology is built by the developer (or by a method engineer) assembling pieces of the process (method fragments) from a repository of methods built up taking pieces from existing methodologies (Adelfe, AOR, Gaia, MESSAGE, PASSI, Tropos, ...). In this way he/she could obtain the best process for his/her specific needs.

In the last years, the method engineering approach proved successful in developing object oriented information systems(Saeki 1994)(Tolvanen 1998). Its importance in the OO context should be evaluated considering not only the direct

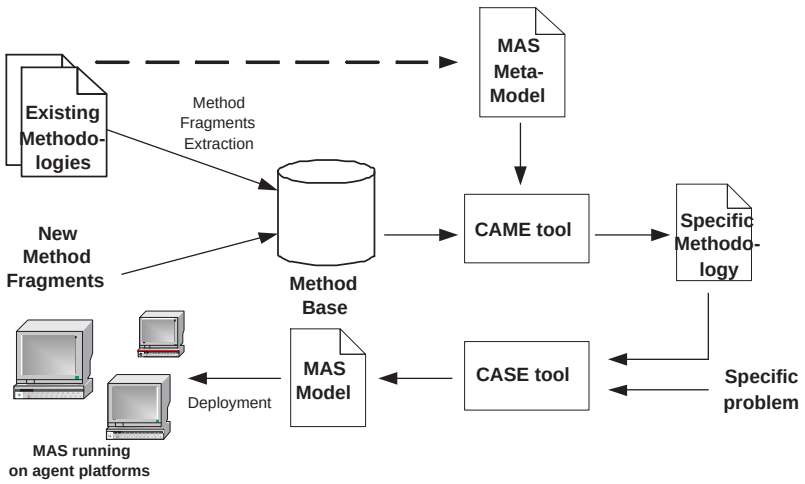

Figure 1: The proposed design methodology construction process

influence (not so much companies and individuals work in this specific way) but the indirect consequence that now, the most important and diffused development processes (for example RUP, the Rational Unified Process) are not rigid but they are a kind of framework within which the single designer can choose his/hers own path.

It could seem that introducing the method engineering paradigm in the AOSE context is a plain operation. But it is not so, because in the OO context the construction of method fragments (pieces of methodology), the assembling of the methodology with them and the execution of the design rely on a common denominator, the universally accepted concept of object and related model of the object oriented system. In the agent context, there is not an universally accepted definition of agent nor it exists any very diffused model of the multi-agent system.

We think that designing a system (object or agent-oriented) consists in instantiating the system meta-model that the designer has in his/hers mind in order to fulfill the specific problem requirements. This meta-model is the critical element in applying the method engineering paradigm to the agents world.

Referring to a MAS meta-model we mean a structural representation of the elements (agent, role, behavior, ontology,...) that compose the actual system with their composing relationships. Sometimes we can see that these concepts, for example the behavior, are used with slightly different meanings or granularity. We will provide an example of MAS meta-model in the following.

In order to allow the composition of pieces coming from different methodologies, it is necessary to express all of them in a common way. The first step of this work consists in the creation of the meta-model that will be used to describe the existing methodologies and the multi-agent system structure (MAS meta-model). An important contribution to this approach comes from an OMG specification, the Software Process Engineering Metamodel (SPEM)(OMG 2002). This is the natural candidate to be the meta-model adopted in this activity for processes since it is already an accepted standard in the OO context (and OO process are not too different from the AO ones). Moreover from the analysis of many existing approaches to these problem, Fuggetta (Fuggetta 2000) says that Process Modeling Languages "must be easy to use, intuitive, and tolerant"; all of these are properties that we can find in SPEM.

We are currently evaluating the possibilities offered by SPEM in the specific agent-oriented context obtaining interesting results in the representation of the different fragments of PASSI (Cossentino, Sabatucci, & Seidita. 2003b)(Cossentino, Sabatucci, & Seidita. 2003a).

From the descriptions of methodology processes we will extract the method fragments. A method fragment is a reusable part of a design process that taking some already designed pieces of the system produces a new part of the design following a precise procedure. In this phase of the work we will again be near to the FIPA Methodology Technical committe work by adopting its specitication of (method) fragment that we therefore consider composed by (see also the FIPA method fragment definition (FIPA 2003)):

1. A portion of process
2. One or more deliverables (artifacts like (A)UML/UML diagrams, text documents and so on). Some preconditions (like the required input data or guard condition)
3. A list of concepts (related to the MAS meta-model) to be defined/designed/refined by executing the specific method fragment.
4. Guideline(s) that illustrates how to apply the fragment and best practices related to that A glossary of terms used in the fragment
5. Other information (composition guidelines, platform to be used, application area and dependency relationships useful to assemble fragments) complete this definition.

These fragments will be collected in a method base. This introduces a new important step of our plan: the study of pos-

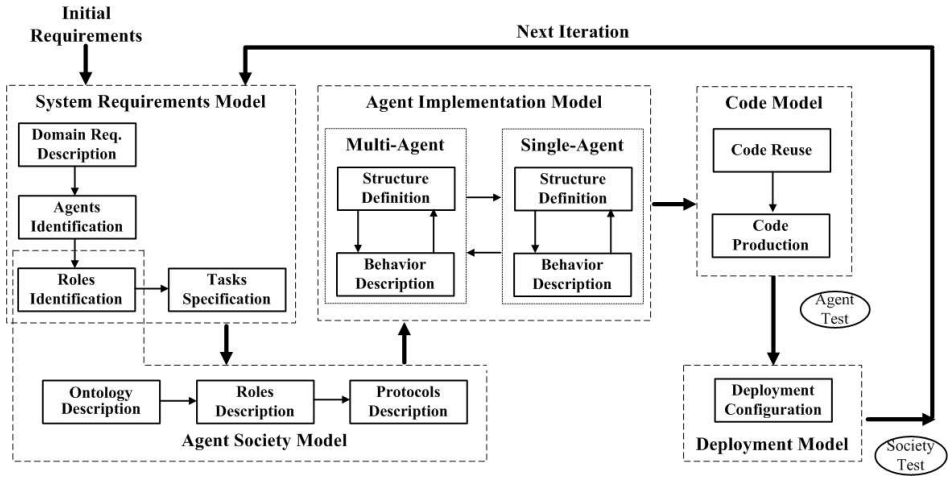

Figure 2: The PASSI process

sible technological solutions for the implementation of this database in order to obtain a representation of the fragments that could be easily supported in a CASE/CAME (Computer Aided Software Engineering/Computer Aided Method Engineering) tool. The last crucial phase of the work will be the study of the method fragments composition strategies. Each method fragment produces an artifact that contributes to the construction of the complete MAS design model. Composing fragments coming from different methodologies implies considering that they may refer to different models of the system. For example they could address the concept of role in a slightly different way and as a consequence, reusing roles defined with one methodology in another context could bring to inconsistent or incomplete models. Anyway, this artifact structure is only one of the aspects of the problem we are dealing with. There is also a procedural point of view. In taking two method fragments from a repository and reusing them, we could find that they do not exactly match. It could be necessary to integrate them with some more activities that should complete the process.

In our approach we will face both the problems giving the right importance to the MAS meta-model and using it as a beacon for orienting the choices. Obviously we will refer to the currently under development FIPA modeling language as the modeling language used to represent all the artifacts produced by our method fragments and this will facilitate their integration and minimize the effort needed to comprehend and evaluate a larger number of possible solution strategies.

When our work will be completed we think that in a real design process (Figure 1), the designer (or better the method engineer), before building his/hers own methodology, has to select the elements that compose the meta-model of the MAS he/she will build. In so doing he/she uses a CAME tool (Computer Aided Method Engineering tool) that offers a specific support for the composition of a methodology from existing fragments or with new ones. The availability of the MAS meta-model will help him/her both at a logical and practical level. First this will be useful in the method fragment selection phase (avoiding the selection of methods referring different elements) and secondly, the same fact of clearly declaring the structure of the system will allow the CASE tools to check for model coherence and to find not completely defined parts. Once composed the methodology the designer will perform the established process obtaining a model of the system that solves his/hers problem. Finally he/she could deploy the agents on the required platforms.

An example of MAS Meta-model

In this subsection we will show an example of MAS meta-model extracted from the PASSI (Process for Agent Societies Specification and Implementation) methodology (Cossentino & Potts 2002) that has been specifically conceived to be supported by a CASE tool that automatically compiles some models that are part of the process, using the inputs provided by the designer. PASSI is a step-by-step requirement-to-code methodology for developing multi-agent software that integrates design models and philosophies from both object-oriented software engineering and MAS using UML notation. We widely applied it in the design of robotics applications (Chella *et al.* 2002) but it also proved successful in designing information systems (Burrafato & Cossentino 2002). In PASSI (Fig. 2), the reuse of existing patterns has a great importance but unlike other authors (Kendall (Kendall *et al.* 1998)), we chose to introduce a pattern definition that is based on a specific architecture (the FIPA one); this simplified the introduction of patterns in the early design phase and their final code implementation with the use of specific design tools (Agent Factory (Chella, Cossentino, & Sabatucci 2003)). During a PASSI design process, designers will use a Rational Rose add-in that we have specifically produced. In this procedure they move gradually from the problem domain (described in the System Requirements Model and Agent Society Model) toward the solution domain (mainly represented by the Agent Implementation Model) and, passing through the coding phase, to the dissemination of the agents in their world. It is in this progress of activities that they can iden-

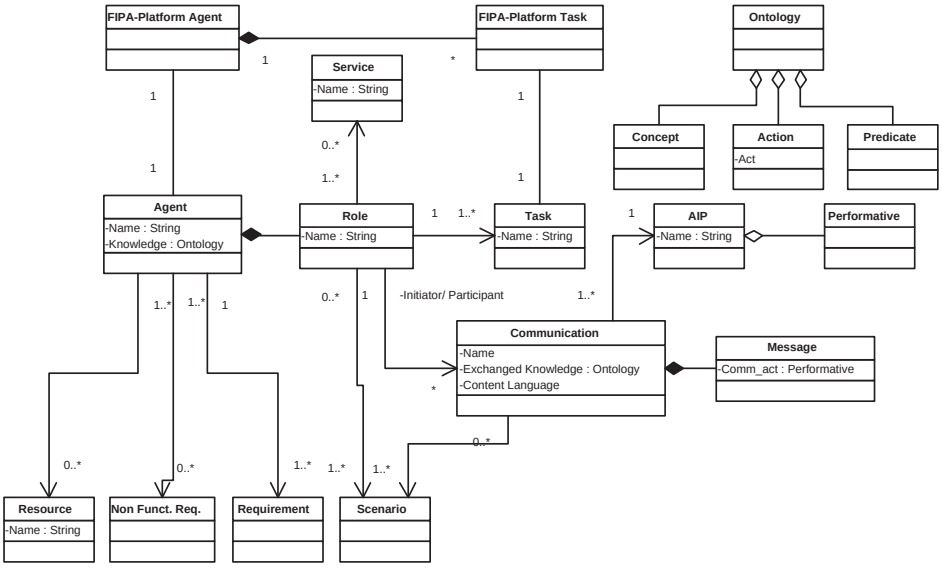

Figure 3: The PASSI MAS Meta-model

tify some problems that could be profitably solved reusing the patterns of our repository. The choice of the implementation platform is postponed to the final steps of the design and in order to support the localization of our patterns in both the most diffused FIPA (O'Brien & Nicol 1998) platforms (FIPA-OS (Poslad, Buckle, & Hadingham 2000) and JADE (Bellifemine, Poggi, & Rimassa 2001)) we represent the models and the code of each pattern using XML from which we obtain the final Java code using some XSL transformations.

In Figure 3 we can see the MAS meta-model related to the PASSI methodology. It describes the user's problem in terms of scenarios, requirements, ontology and resources; scenarios depict a sequence of interactions among actors and the system. These scenarios are initially expressed using a textual description and later designed using conventional UML sequence diagrams. Requirements initially are reported in a textual document and they are later represented with conventional use case diagrams. There is a strong point behind these choices: a lot of highly skilled designers are already present in different companies and can be more easily converted to the use of an agent-oriented approach if they are already confident with some of the key concepts used within it. Analysis related issues (like requirements and scenarios) being situated in the highest abstraction phase are strategic in enabling this skill reuse and allow a smooth entering in the agent solution part of the meta-model. Ontological description of the domain is composed of concepts (entities and 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 elements). This represents the domain in a way that is substantially richer than the classic structural representations produced in the OO analysis phase. As an instance we can consider ontologies devoted to reasoning strategies or problem solving methods whose essence is very difficultly captured in OO structures. Resources can be accessed/shared/manipulated by agents. A resource could be a repository of data (like a relational database), an image/video or also a good to be sold/bought. We prefer to expressly model them since goals of most systems are related to using and capitalizing available resources.

The central part of the model specifically deals with the agent solution that is to be built but yet at an abstract level. It contains the elements of the agent-based solution but none of these elements is directly implemented; they are converted to the correspondent object-oriented entity that constitutes the real code-level implementation. The real center of this part of the model is the concept of agent. Each agent in PASSI is responsible for realizing some functionalities descending from one or more requirements. The direct link between a requirement and the responsible agent, is one of the strategic decisions taken when conceiving PASSI. Sometimes an agent has also access to available resources. This could happen because it accesses the corresponding information (for example stored in a DB) or it can perceive it using its sensors (like in the case of em-bodied robotic agents sensing the environment). Each agent during its life plays some roles. In PASSI, a role is considered as a portion of the social behaviour of an agent characterized by some specificity such as a goal, or a set of attributes (like responsibilities, permissions, activities, and protocols) or providing a functionality/service.

From this definition easily descends that roles could use communications in order to realize their relationships or portions of behaviour (called tasks) to actuate the role proclivity. In PASSI, the term task is used with the significance of atomic part of the overall agent behaviour and, therefore, it describes some agent capabilities; from a different point of view, we can say that the agent can accomplishing its duties differently composing the set of its own tasks. Tasks cannot be shared among agents, but their possibilities could

be offered by the agent to the society as services (often a service is obtained composing more than one task). Obviously according to agent autonomy, each single agent has the possibility of accepting or refusing to provide a service if this does not match its personal attitudes and will. This is totally different from an object-oriented service where an object/component could not refuse a service that is required via the strong relationship of a method invocation instead of the weak connection established by agents' asynchronous communications. A communication (also called conversation in FIPA specifications) is composed of one or more messages expressed in an encoding language (like the FIPA ACL) that is totally transparent to agents. The message content could be expressed in several different content languages (SL, CCL, KIF, RDF); we chose to adopt RDF (Resource Description Framework, a specification by W3C) and the PASSI supporting tool (PTK) offers a concrete aid in generating the RDF code from the design models of ontology. Each communication explicitly refers to a piece of ontology (in the sense that information exchanged are concepts, predicates or actions defined in the ontology) and its flow of messages is ruled by an interaction protocol (AIP). Interaction rules of agent conversations plays an important role in the FIPA Abstract Architecture; these have been formalized primarily through two concepts: the communicative act and the AIP (Agent Interaction Protocol). Each conversation has to respect a protocol and has to be made up of communicative acts. A communicative act is a way to associate a predefined semantic to the content of a message so that it can be univocally understood by agents. Some examples of communicative acts are: request, refuse, agree, inform, and failure. An interaction protocol defines which communicative acts may be used in a conversation and in what order the related messages have to be sent to give the proper meaning to the communication. Therefore, a protocol compels the use of determined messages with a specific semantic according to a specific sequence. When an agent starts a conversation with another agent it has to specify a protocol; a conversation without a protocol is not possible in FIPA-based systems. If a message does not respect the rules of the protocol or violates the prescribed order, then the conversation fails.

The implementation part of the meta-model describes the structure of the code solution in the chosen FIPA-compliant implementation platforms (like FIPA-OS or JADE) and it is essentially composed of three elements: the FIPA-Platform Agent that represents the implementation class for the agent entity represented in the Agency domain; the FIPA-Platform Task that is the implementation structure available for the agent's Task and, finally, the Service element that describes a set of functionalities offered by the agent under a specific name that is registered in the platform service directory and therefore can be required by other agents to reach their goals. This description is also useful to ensure the system openness and the reusability of its components.

Experimental Setup

Robotics is the primary application field for the systems we develop with our methodologies. In order to focus our attention on a specific problem we are now looking at a scenario involving a robotic system devoted to surveillance tasks. More in detail the implemented functionalities comprehend the reconnaissance of the building, the automatic detection of an intruder, the pursuit (and encirclement if more robots are available) of the intruder.

Figure 4: The B21 robot, during a surveillance mission, is calculating its position using the a-priori known position of the marker that it found on its left.

From the hardware point of view, the system is composed of one (but more is possible) B21 mobile robot with a computer and a stereo camera aboard; some fixed cameras are positioned in the environment (a floor of our department) in order to detect the intruder and four fixed workstations are used for agents deployment. The software aspects are characterized by a multi-agent system implemented with a FIPA-compliant platform (JADE). The system is composed of 16 different agents (six of them devoted to implement the robot behavior, the others related to vision tasks). Several of these agents are instantiated more than once at runtime.

A particular effort has been dedicated to the vision subsystem that introduces a multi-level architecture allowing the dynamical introduction of new hardware (fixed cameras that are distributed in the environment in order to detect intruders) (Infantino, Cossentino, & Chella 2002) and services (agents performing different kinds of filtering and images manipulations). One instance of a grabber agent is bound to each camera to capture the images. Several instances of these agents are used and therefore an *HardwareManager* agent is necessary to allow other agents to interact with the best positioned (or more useful) camera for each specific purpose. The captured images are then manipulated by other agents that can perform tracking, motion detection, camera calibration and other operations. A *SelfLocalizator* agent localizes the robot in the environment using two images captured by its stereo camera while looking at a landmark whose position is known a priori (see Figure 4). This information is also used to correct the odometry error. Other

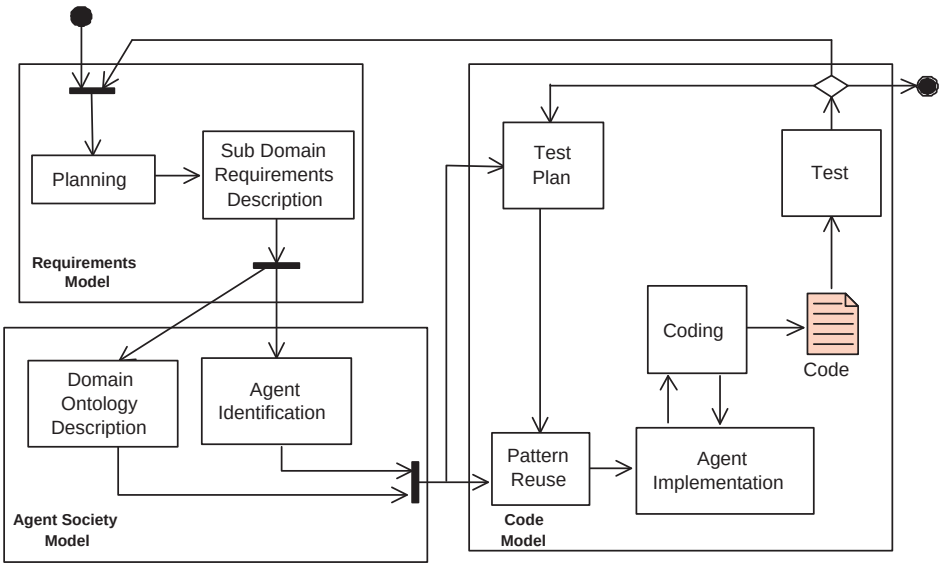

Figure 5: The agile process composed reusing fragments coming from the PASSI methodology

sensors (infra-red, laser range finder and compass) are managed by an unique agent, the *SensorReader*. The dimensions of this project are quite interesting. The total amount of code is about 10.6 thousands of lines, the time spent in developing the MAS and ontology (design, coding and testing) with the conventional PASSI methodology and the use of patterns was about 7 man/months.

Our actual aim (whose related activities are ongoing) consists in rebuilding this whole application with a new methodology that is specifically conceived to move the attention on code and algorithms specification rather than on other aspects (that are instead typical of a conventional methodology) and that could divert the researcher's attention from his main focus that is related to implementing some kind of new robotics solution using agents as the most suitable paradigm to achieve a fast and reliable system. The methodology we are working on (see Figure 5), is an agile process (Agi),(Beck *et al.* 2001) and follows some of the rules prescribed in the XP (Extreme Programming) approach (Wells 2003). According to the previously presented approach the new methodology construction process is based on the reuse of existing method fragments (coming from PASSI) and is supported by a specific design tool. The methodology is composed by three main models: the Requirements Model where the iteration is planned (in terms of identification of a portion of the system to deal with), and requirements are captured; the Agent Society model where agents are identified and domain ontology designed; the Code model where test is planned and reusing pattern the code is produced. The coding phase is supported by a tool that from previous specifications builds some diagrams (agent implementation diagrams inherited from PASSI) and helps the designer in compiling several aspects of the design (for example setting up communication parameters). In case of changes made to the code, design artifacts are updated on the fly thanks to a reverse engineering process. Future experiments will consists in rebuilding the discussed robotic surveillance application with this new process and compare the results with the ones obtained by adopting the PASSI methodology.

Conclusions

Our work is, by now, mainly centered on the study of several MAS meta-models and the description of the PASSI methodology in SPEM (Software Process Engineering Metamodel) in order to extract the method fragments from it. Since other research groups are preparing methodg fragments in a similar way we are confident to be able to reuse the results of their work too at the same time we will contribute with our ones to their effort. This fragment identification activity is complemented by an ongoing study of the technological solutions that could be used to realized the method fragments repository. Another crucial point will be the study of possible guidelines for the methodology creation via the fragments assembling. Several approaches exist in literature about OO systems but in our specific context others could be explored like some ontology related ones.

The discussed approach is undoubtedly long and complex but we think it can be considered very promising also because it is near to the one adopted by the FIPA community and in this context it will find the stage for its application and probably several people operating in that context will have the opportunity to concretely apply it.

As an intermediate goal we are now composing an agile process whose application field is the development of agent-based robotic systems.

References

Agile alliance. http://www.agilealliance.org.

Beck, K.; Beedle, M.; van Bennekum, A.; Cockburn, A.; Cunningham, W.; Fowler, M.; Grenning, J.; Highsmith, J.;

Hunt, A.; Jeffries, R.; Kern, J.; B.Marick; Martin, R.; Mellor, S.; Schwaber, K.; J.Sutherland; and Thomas, D. 2001. Agile manifesto. http://www.agilemanifesto.org.

Bellifemine, F.; Poggi, A.; and Rimassa, G. 2001. Jade - a fipa2000 compliant agent development environment. In *Agents Fifth International Conference on Autonomous Agents (Agents 2001)*.

Brinkkemper, S. 1995. Method engineering: engineering the information systems development methods and tools. *Information and Software Technology* 37(11).

Burrafato, P., and Cossentino, M. 2002. Designing a multi-agent solution for a bookstore with the passi methodology. In *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)*.

Chella, A.; Cossentino, M.; Pirrone, R.; and Ruisi, A. 2002. Modeling ontologies for robotic environments. In *The Fourteenth International Conference on Software Engineering and Knowledge Engineering*.

Chella, A.; Cossentino, M.; and Sabatucci, L. 2003. Designing jade systems with the support of case tools and patterns. *Exp Journal* 3(3):86–95.

Cossentino, M., and Potts, C. 2002. 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.

Cossentino, M.; Sabatucci, L.; and Seidita., V. 2003a. Method fragments from the passi process. *Rapporto tecnico ICAR-CNR* (21-03).

Cossentino, M.; Sabatucci, L.; and Seidita., V. 2003b. Spem description of the passi process. *Rapporto tecnico ICAR-CNR* (20-03).

FIPA. 2003. Method fragment definition. FIPA Document, http://www.fipa.org/activities/methodology.html.

Fuggetta, A. 2000. *Software Process: A Roadmap*. ACM Press. chapter The Future of Software Engineering.

Infantino, I.; Cossentino, M.; and Chella, A. 2002. An agent based multilevel architecture for robotics vision systems. In *The 2002 International Conference on Artificial Intelligence*. Las Vegas (NV), USA: ICAI'02.

Kendall, E. A.; Krishna, P. V. M.; Pathak, C. V.; and Suresh, C. B. 1998. Patterns of intelligent and mobile agents. In Sycara, K. P., and Wooldridge, M., eds., *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, 92–99. New York: ACM Press.

Kumar, K., and Welke, R. 1992. Methodology engineering: a proposal for situation-specific methodology construction. *Challenges and Strategies for Research in Systems Development* 257–269.

Luck, M.; McBurney, P.; and Preist, C. 2003. *Agent Technology: Enabling Next Generation Computing. A Roadmap for Agent Based Computing*. http://www.agentlink.org/roadmap.

O'Brien, P., and Nicol, R. 1998. Fipa - towards a standard for software agents. *BT Technology Journal* 16(3):51–59.

OMG. 2002. Software process engineering metamodel - version 1.0. OMG Document. http://www.omg.org/technology/documents/formal/spem.htm.

Poslad, S.; Buckle, P.; and Hadingham, R. 2000. The fipa-os agent platform: Open source for open standards. In *5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*.

Saeki, M. 1994. Software specification & design methods and method engineering. *International Journal of Software Engineering and Knowledge Engineering*.

Tolvanen, J.-P. 1998. Incremental method engineering with modeling tools: Theoretical principles and empirical evidence (ph.d. thesis). *Jyvskyl Studies in Computer Science* 301.

Wells, D. 2003. Extreme programming. a gentle introduction. http://www.extremeprogramming.org.

Zambonelli, F., and Parunak, H. V. D. 2002. Sign of a revolution in computer science and software engineering. In *3rd International Workshop on Engineering Societies in the Agents' World*. LNAI.