AN AGENT ORIENTED TOOL FOR NEW DESIGN PROCESSES

Massimo Cossentino [b,c] Luca Sabatucci [a] Valeria Seidita [a] Salvatore Gaglio [a,c]

[a] *DINFO - Dipartimento di Ingegneria Informatica, Università degli Studi di Palermo*
Viale delle Scienze, 90128 Palermo, Italy
sabatucci@csai.unipa.it; seidita@csai.unipa.it
[b] *SET - Université de Technologie Belfort-Montbéliard - 90010 Belfort cedex, France*
cossentino@pa.icar.cnr.it
[c] *Istituto di Calcolo delle Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo, Italy*
gaglio@unipa.it

Abstract

In the last years we applied the Method Engineering paradigm to the development of agent-oriented design processes. The main difficulty in our initial experiments was ensuring the support of a customised CASE tool to the new methodology. In this paper we now present a Computer-Aided Process Engineering (CAPE) tool that we developed in order to allow the design of the new process and then its instantiation. The process is executed as a workflow, the designer receives the help of an expert system (for routine works automation and syntax/semantic checks) and can model his/her system using a set of Eclipse plug-ins supporting all UML diagrams. The development of this tool started from the definition of a system meta-model obtained from the initial requirements and then instantiated using open-source and ad hoc developed components; as it could be expected relevant portions of this tool are developed using the agent-oriented paradigm.

1. Introduction

Agent-oriented research in the last decade achieved significant advancements in the application of software engineering techniques to multi-agent systems (MAS). The growing interest for this field brought to the creation of a specific branch of software engineering called Agent-Oriented Software Engineering (AOSE) [2, 18, 26]. Initial works in this field received a relevant influence by artificial intelligence and more attention was given to model reasoning capabilities of single agents rather than defining the system architecture and real implementation possibilities. This has been overcome by the availability of recent approaches that allows the successful application of the agent paradigm to real and complex problems. Complexity is indeed becoming one of the leading arguments in AOSE. Engineering very large complex system is a difficult challenge that can find interesting solutions in agent societies. New works on systems adaptation (also in relation with the relevant distribution rate of enormous number of agents) [11] are arising from the attention of researchers and practitioners for such agent societies and the possibility of facing the problem with a kind of bottom-up approach based on agents self-organization capabilities.
Several design methodologies have been developed in the last years, each one with a specific attention for some categories of problem, development context or design approach [1, 3, 10, 22, 23, 29, 30].

A growing interest for the possibilities offered by the results of (situational) method engineering [12, 13, 21] is a natural consequence of this dynamic situation. The agent community work on this discipline is also characterized by a relevant attention for the system meta-model that is, in the case of MAS, not a well established one (like it is in the object-oriented world) but rather, it is one of the distinguishing elements of

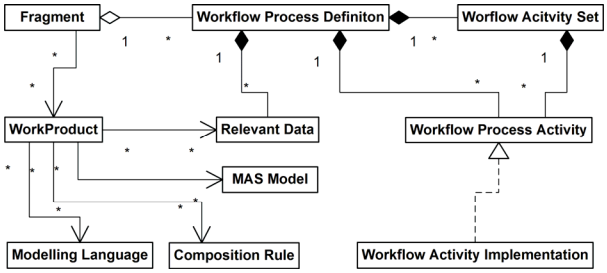

Figure 1 - The fundamental elements of the Metameth meta-model

each existing design process. Just to provide an example of the great spread of MAS meta-models proposed in literature we can consider that even their real core, the concept of agent, is not subject to a commonly agreed definition and some moderately shared properties like autonomy, pro-activeness, social ability and reactivity [27] are not adopted by all the authors.
In this paper we are going to discuss the development of a tool we built in order to support our method engineering experiments. Our aim was to have a tool (we called it Metameth) that may be used to define a new (agent-oriented) design process and can support the design phases in both research activities as well as students' projects, where different stakeholders can participate in the design of an unique system in an asynchronous and distributed way.

The real development process of Metameth started with the collection of the requirements it should fulfill, after an analysis of these requirements we defined the system meta-model that could be used to satisfy all the constraints. This meta-model has been finally implemented in the real application that is essentially composed of 4 main components:

- the process definition component (an open source graphical tool, JaWE [17] by Enhydra that allows the definition of a process in form of a workflow process);
- the process execution (Shark [25], an open source workflow engine, again by Enhydra) that orchestrates the different activities allowing their distributed, asynchronous, and (in a limited way) collaborative execution;
- each designer essentially works in tight relationship with a personal agent that takes care of executing recursive tasks (like starting the automatic composition of some artifacts and performing several syntax/semantic checks during the design phase;
- the designer performs his/her work also with the support of a UML modeling tool we built as an IBM Eclipse plug-in [14]. This tool supports UML 1.5 specifications and also extends them as requested for some diagrams of the PASSI process [10].

Metameth has been developed during the last two years with the involvement of two PhD students and up to twelve master graduating students for a total effort of almost seven man/years.
The paper is organized as follows: in section 2 we discuss the requirements of the Metameth tool and some choices we adopted in its development. In section 3 we present the system meta-model we defined as a consequence of the previous discussed choices and then in section 4 we introduce a brief usage scenario in order to present the resulting prototype. Finally some conclusions are drawn in section 5.

2. The Metameth Project

This work started from a set of requirements arising from our previous experiences. Our first attempt in the development of design tools was the creation of a plug-in for Rational Rose (we called it PTK, PASSI ToolKit) [7] that introduced several checks in the design semantics and the automatic composition of several PASSI diagrams. Code generation was performed with the help of a pattern reuse tool and good results were obtained about that [6].

In 2004 there was an evolution of PASSI in the direction of agile processes. The consequent agile PASSI was the result of an experiment of design process creation using method engineering [8]; it has been supported by a tool based on MetaEdit+ (by MetaCase) whose flexibility in modeling the semantic and notation of single method fragments proved very useful. We extended the MetaEdit+ features including the

automatic composition of some diagrams (mostly based on reverse engineering of code for documentation purpose) and pattern reuse support [5]. The development of such a tool although simple and not totally engineered was a high cost and not easily reproducible for future experiments of design process construction.

In the last year we decided to collect all the experiences done with the use of PASSI and evolve it towards a new release that is still under test in our laboratory. In the meanwhile we faced the need of producing a new tool (Metameth) to support it and decided to enlarge the requirements of this tool towards a more comprehensive system that could support our future experiments in method engineering.

In the following subsections we discuss the requirements we identified for our tool; starting from these requirements we perform the fundamental architectural choices that are finally presented in subsection 1.2.

1.1 Requirements for the Metameth tool

Our work started with the identification of the requirements for the tool to be built. As said before, we had some experience with the production and use of a couple of other tools in the past. It is obvious to expect a real enhancement with the new project without loosing any of the positive features of the old systems; their main limit was the impossibility of easily sharing the different design phases (collaborative work was not supported) and the resulting artifacts. To overcome this limitation we decided to explicitly support distributed design processes (involving several designer working on different phases at the same moment in different locations) (**Req. 1**).

As already discussed our attention was also focused on helping the designer in properly adopting the prescribed flow of activities. We estimate that our design processes will be very frequently iterative/incremental, also because no other specific needs arose in our previous works for the design of MAS on that, see [4]; therefore we decided that the tool should support a design process composed of several activities (eventually organized in some kind of hierarchy) with several different possible iteration paths (**Req. 2**).

We also considered important to plan an easy extensibility of our tool. Being a method engineering tool it should obviously support several different design processes (this means allowing the composition of different processes starting from a repository of method fragments, **Req. 3**) and the related notations (proper design instruments should be available, **Req. 4**) but also different code generation modules in such a way that different coding languages or implementation frameworks could be adopted. As regards the fragments repository we explicitly address an existing proposal coming from the old FIPA Methodology Technical Committee [9, 24] whose general structure will be discussed later.

Finally, the tool requirements include the possibility of automatically composing (portion of) diagrams

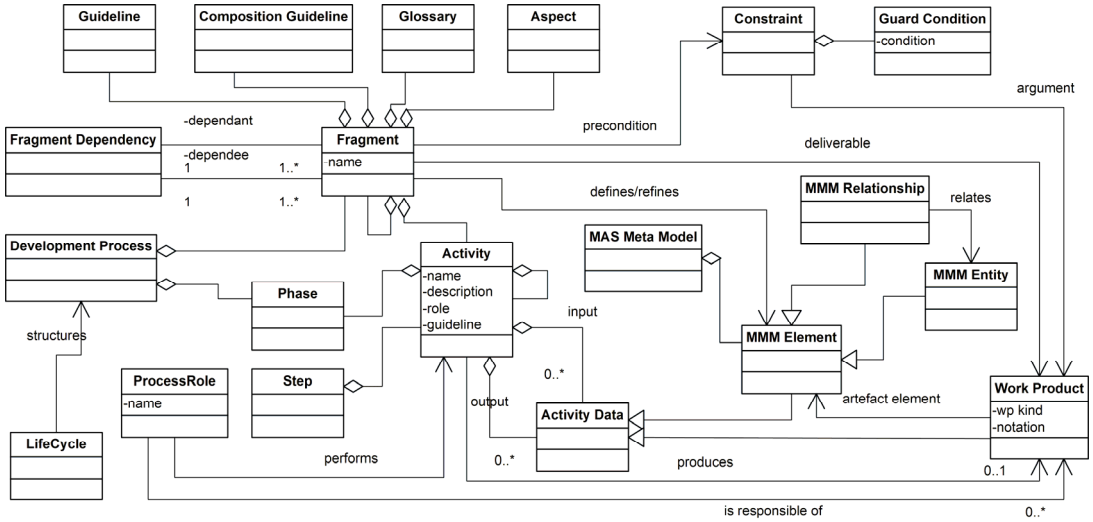

Figure 2 – The Fragment Meta Model

when it is possible (**Req. 5**), the syntax check on notational aspects of the project (**Req. 6**) and the consistency check on some design issues like the correct instantiation of the most important elements of the MAS meta-model (**Req. 7**), pattern reuse (**Req. 8**), code generation (**Req. 9**) and finally the production of an adequate documentation (**Req. 10**) that, starting from the produced artifacts, is able of creating complex documents by merging diagrams, text, tables and so on.

In the next sub-section, from the analysis of the listed requirements we are going to deduce the strategic choices and general architecture for the tool.

1.2 Requirements analysis and strategic choises

It is worth to remember that the tool we are discussing has been developed in the context of a research institute, with the work of PhD and graduating students. In this context and with these involved resources, the fulfillment of Req. 1 (distributed design process) and Req. 2 (support for iterative/incremental processes) was a relevant challenge for our group. We decided to adopt a well-known solution: supporting the software development process as a workflow management process.
For the definition of workflow we refer to the WFMC [28] specifications [16]: "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules".

We think this concept could be profitably applied to the execution of a software development process and therefore we decided to separate the modeling phase of our processes from its enactment as it happens for the design of a WFM process and its execution.

During the process modeling phase we decided to adopt a standard by OMG, the Software Process Engineering Metamodel (SPEM) [20] that we have been using during the last three years. Once the process is modeled in SPEM we use a graphical tool (JaWE) to produce its XPDL translation (XPDL is the process specification language adopted by WFMC). Obviously this requires some compromises that are related to the structure of the system meta-model we define and specifically to the relationships among the workflow elements and the process elements of the method fragments (more details on that will be provided in section 4). The fragment repository (Req. 3) is an extension of an existing repository we built according to the work done within FIPA and it has been extended with the proper set of expert system rules and software components (agents) used to support the specific GUIs required in the fragment; the repository of fragments already exists and we are working for integrating it in the tool. Only the fragments used to implement our own methodologies (classic and agile PASSI process) are already fully integrated in the tool, we are currently working on the software implementation of fragments from other methodologies that are by now described in only a textual form in the repository.

The next group of requirements is inherent to the instantiation of the XPDL produced with JaWE. For this aim we used an open source workflow engine (Shark) integrated in a multi-agent system realizing all the operations required for our tool. The system incorporates an engine for a logical Lisp-like language execution (Jess [19]). This choice allows realizing some kind of reasoning about the information introduced by the designer during his/her work (fulfilling Req. 5-6: artifact auto-composition, syntax and semantic check). In order to support the modeling phases we integrated the system in the Eclipse (IBM Eclipse) infrastructure offering a well known architecture with an easy plug-in support. In this environment we developed a set of UML-compliant diagram editors.

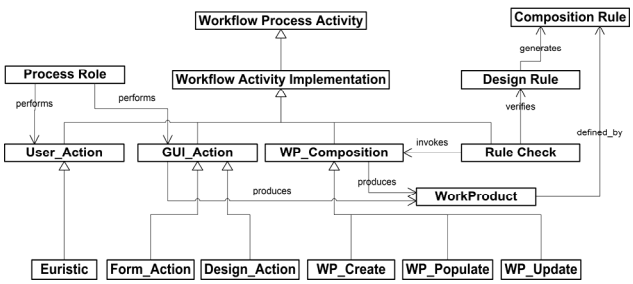

Figure 3 - A detail on the activities of the Metameth meta-model

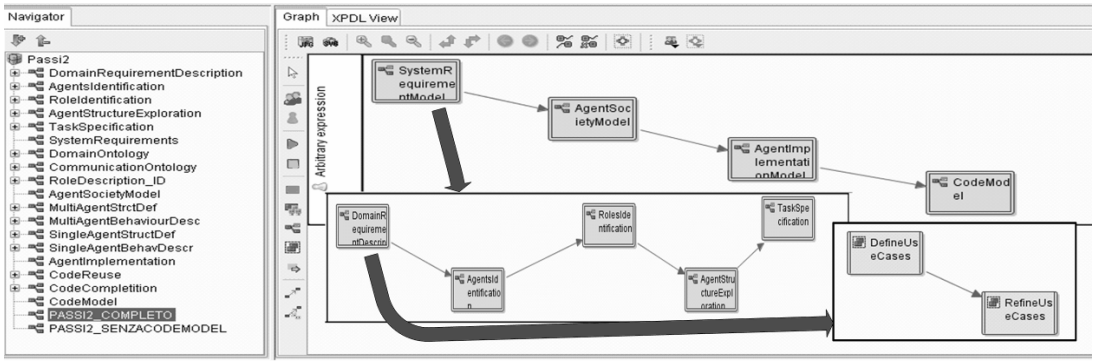

Figure 4 - A screenshot of the JaWE tool used for designing the methodology

3. The system metamodel

In designing our method engineering tool we decided to spend a great attention on the system meta-model. Starting from the results of the requirements elicitation and analysis phases we deduced the fundamental aspects of this meta-model and then we started a more detailed design process. The inputs of our work as already discussed in previous sections were: the general structure of the fragment meta-model, the WFMC workflow process model and the SPEM (software development) process model. Our first step was merging these elements in a unique system meta-model that could profitably capture the best contribution of each of the different standards. The resulting system meta-model also included a specific support for fundamental aspects like system modeling (in form of a plug-in for IBM Eclipse), syntax/semantic check on design artifacts (and resulting instance of the MAS meta-model) and automatic compilation of some artifacts according to their composition rules.

The fundamental elements of the resulting system meta-model are reported in Figure 1. About the method fragment definition, we can say it is composed of (see Figure 2): a process description (in terms of activities and involved actors), produced work products, guidelines, glossary and aspects (regarding specific issues related to fragment reuse). A link to the MAS meta-model element(s) defined/refined in the activities of the specific fragment (and represented in the fragment work products) is also included.

Because of the choice of enacting our design process as a workflow, all process related elements in our system meta-model are substituted by the corresponding elements in the WFMC specifications. For this reason, in Figure 1 the fragment is related to a Workflow Process (also called Process in WFMC specifications [28]) and it is composed of Workflow Activities. Relevant data as specified in the WFMC specifications is: “the data that is created and used within each process instance during process execution”. Because of this meaning we related it to the work product that is used to represent this data. The WorkProduct element is also related to the Modeling Language that is used to draw it and to the MAS Model it represents. A MAS Model is the instance of the MAS meta-model underpinned by the design process. The WorkProduct is also related to the Composition Rules we use to represent all the norms (syntactic/semantic rules, document templates design conventions) defined for the artifact. These rules will automatically be checked by our expert system thus satisfying some of the previous discussed requirements.

Finally we can note that the Workflow Process Activity is realized by the Workflow Activity Implementation element. This is the result of a precise design choice: the workflow engine, as it is common, after triggering some activity, invokes an external application to support the stakeholder in his/her work. In a common business workflow process the application can be a word processor, a spreadsheet or another personal production tool. In our system, the external application is always an agent (we named it Activity Agent) that is responsible for the interaction with the user, the control of design applications (UML design modules built as a plug-in of IBM Eclipse), the collection of new data and the request for syntax/semantic checks (from the expert system).

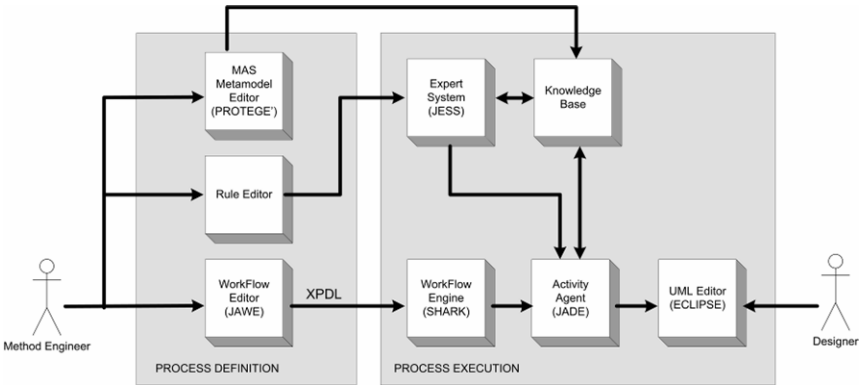

Figure 5 - The abstract architecture of Metameth

In order to better understand some details of the final configuration of our meta-model, we can start from its fundamental element, the Fragment reported in Figure 2. Its main elements have been reported in Figure 1 with the changes needed to include the other parts (mainly workflow support and expert system rules). The original Process Description element has been replaced by the Workflow Process element coming from the WFMC specifications that is composed of Workflow Activities. Differently from the initial schema, these activities are not directly related to the WorkProduct. In fact, in our new approach, each Workflow Activity (here we mean the entity scheduled and managed by the workflow engine) is realized by a Workflow Activity Implementation that corresponds to a real piece of work done by the designer (and in so doing the designer generally interacts with a specific Activity Agent rather than the workflow engine).
The implementation element has been further refined as reported in Figure 3 where we list the categories of design actions supported by our Activity Agents:

- User_Action; it is an action specified in the workflow but not supported at all by the tool (for instance the application of a heuristics).
- GUI_Action: it is an action performed by the designer using a GUI. Two kinds of these actions exist: Form_Action (the designer fills in a form) and Design_Action (the designer composes some diagram, for instance using our UML editor). Gui_Actions are related to WorkProduct since they participate to artifact composition.
- WP_Composition: it is an action performed by the tool to create a new artifact (for instance it corresponds to the instantiation of a new UML diagram) and/or its population with elements already defined in previous steps of the process (this is usually done with the help of the expert system). The update of a work product in order to be consistent with other documents is another kind of possible action (suppose that a design element is renamed elsewhere from someone else, it is necessary to spread the change all over the design process). Obviously WP_composition actions are related to the WorkProduct element too.
- Rule check: it is an action performed by the expert system when it executes the composition and design rules.

The discussed system meta-model will find a realization in the system architecture discussed in the next section.

4. The system architecture

The prototype we developed is based on several technologies, standards and existing tools. The architecture of Metameth is inspired both to the IEEE's recommended practice (see [14]) and the OMG's MDA architecture [20]. We need to adapt these abstract specifications to our specific application context; this caused a resulting architecture where the basic elements are still detectable, but other ones were changed in order to reflect the use of the agent paradigm.

Metameth main sub-systems are reported in Figure 5 and a great part of them has not been developed from scratch; listing the most relevant open source components we reused in our Metameth tool we have:

1. JaWE (by Enhydra) adopted as a graphical workflow editor to design the process (it exports the process using the XPDL format);

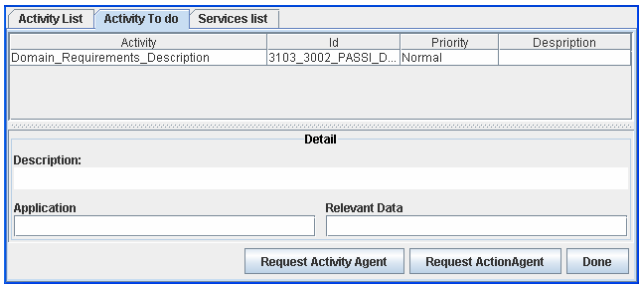

Figure 6 - A screenshot of the Stakeholder agent during the selection of an activity by the designer

2. Shark (again by Enhydra) adopted as a workflow execution engine (it is able of reading XPDL files and also to interact with our Java-based Activity Agents);
3. Jade is the platform we used to develop our Activity Agents; this is the most diffused FIPA-compliant agent development platform.
4. Jess is the Java-based rule engine we used to build our expert system; such a system is the 'intelligent' part of the Metameth tool; a relevant portion of its services are required by the Activity Agents that need reasoning capabilities in order to assist the designer in his/her duty.
5. The MAS meta-model required by the adopted design process is depicted in form of an ontology using the tool Protégé by the Stanford University, California.
6. IBM Eclipse is the IDE we used to develop our UML editors.

Now we are going to illustrate the main steps of the construction of a new design process and its enactment with the Metameth tool.

The scenario starts with the JaWE session used to define the new process; this tool offers a graphical interface to model the process as a flow of sub-processes, and activities; each activity may be atomic or decomposed in sub-activities (Figure 4).

The next step in the definition of the design process is the specification (using Jess rules) of the semantics of the MAS meta-model elements and the (work products) composition rules of the process. This is done by using Protégé for drawing the ontology and a specifically built Rule editor tool for describing Jess rules starting from some templates.

When the design process has been entirely described using the XPDL language (process aspects) and Jess rules (semantics and composition rules), the process administrator may instantiate it using the process execution module. This has been developed using a multi-agent system composed by:

1. A Controller agent (that is interfaced with the workflow execution engine).
2. One or more Stakeholder agents, one for each designer that uses it to accept, start, decline the activities assigned to him from the process definition. After a log-in session, the user may verify his activity list and can start/ refuse or delegate an activity. Figure 6 shows the user interface for a system analyst who is going to accept a *define use case* activity.
3. An Expert agent (used to wrap the Jess expert system).
4. One or more Activity agents. When the designer chooses to perform an activity, an agent becomes responsible for coordinating all the operations related to the specific activity (also in collaboration with the Expert agent and the UML editors). Activity agents offer several services to the designer: i) auto-composition used when a work product can be automatically modified/created or updated; ii) notation interpretation, used to map notational elements (use cases, classes, activities, …) into elements of the MAS meta model (requirements, agents, behaviours, …), iii) semantic validation used to verify the semantic consistence of the whole project.

In Figure 7 the first three work products of the process are reported (domain requirements description, agent identification and role identification).

5. Conclusions and Future Work

Method Engineering received a relevant attention from some members of the agent-oriented community in the last few years. This is the consequence of a well-identified need for a rapid assessment of existing agent-oriented methodologies and possibly the development of a widely accepted standard in the field. The

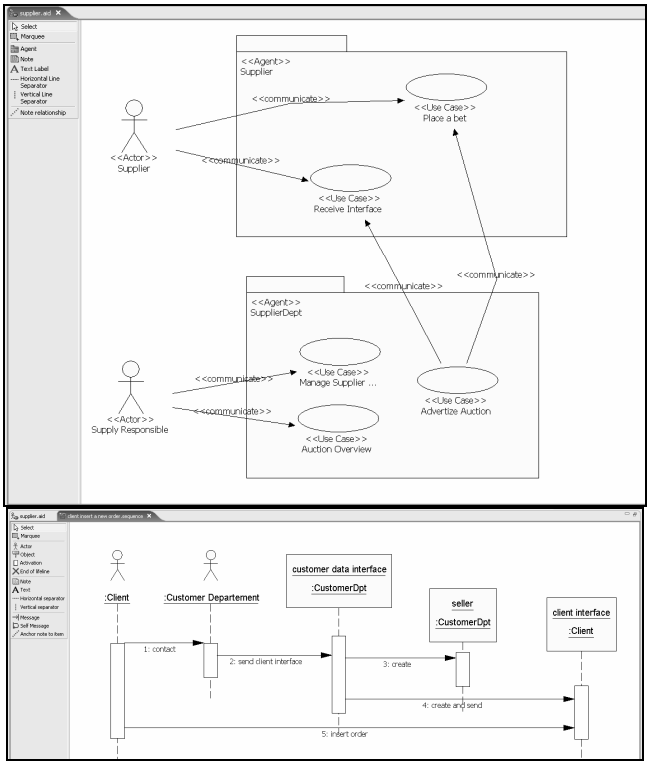

Figure 7 - Some screenshots of the UML editors we developed as a plug-in for Eclipse

novelty brought by some of these works is the focus on the multi-agent system meta-model. This is a crucial element that can be seen as the signature of different approaches; it is a matter of fact that different methodologies in this context adopt models that differ starting from the definition of the same basic notion of agent.

In this paper we present the tool we developed to support our experiments of methodologies composition; it supports the method engineer in the definition of a design process (seen as a workflow process) and the definition of the composition/syntax/semantic rules of the work products; these rules will be used by an expert system to supervise the design activities and provide some automation level.

A workflow engine ensures the design process instantiation and allows the distributed and asynchronous execution of the different activities. The tool is completed by a set of plug-ins for IBM Eclipse we developed in order to support UML diagrams and some specific diagrams of our preferred methodologies.
In the future we are going to complete this tool with more features; specifically we are going to interface it with an agent-oriented pattern reuse tool that allows code generation for one of the most diffused agent development platforms (Jade). The production of an extensive and well-formatted documentation from the design artifact is also scheduled and will be obtained through a society of agents; each one specialized for the composition of one specific kind of document.

References

[1] Bernon, C., Camps V., Gleizes M. P., and Picard G. 2005. Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology, To appear in Agent-Oriented Methodologies edited by B. Henderson-Sellers and P. Giorgini, Idea Group Pub

[2] Bernon, C., Cossentino, M. and Pavon, J. 2005. An Overview of Current Trends in European AOSE Research. Informatica Journal, vol. 29, Number 4.

[3] Bresciani P., Giorgini P., Giunchiglia F., Mylopoulos J., and Perini A. 2004. TROPOS: An Agent-Oriented Software Development Methodology, Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers 8(3), pp. 203-236

[4] Cernuzzi, L.,Cossentino,M., Zambonelli, F., 2005. Process Models for Agent-based Development. *Journal of Engineering Applications of Artificial Intelligence, (EAAI).* Elsevier. Vol. 18, No.2.
[5] Chella A., Cossentino M., Sabatucci L. and Seidita V. 2006. Agile PASSI: An Agile Process for Designing Agents. International Journal of Computer Systems Science & Engineering. Special issue on "Software Engineering for Multi-Agent Systems". May.
[6] Cossentino M., Sabatucci L. and Chella A. 2003a. A Possible Approach to the Development of Robotic Multi-Agent Systems. IEEE/WIC Conf. on Intelligent Agent Technology (IAT'03). Halifax (Canada), October, 13-17, pp 539- 44.
[7] Cossentino M., Sabatucci L. and Chella A. 2003b. Designing JADE systems with the support of CASE tools and patterns in Special Issue on JADE of the Telecom Italia Journal EXP of September
[8] Cossentino M. and Seidita V. 2005a. Composition of a New Process to Meet Agile Needs Using Method Engineering, Lecture Notes in Computer Science, Volume 3390, Feb 2005, Pages 36 – 51
[9] Cossentino M., Garro, A. 2005b. Activity of the FIPA Methodology Technical Committee. ICAR-CNR Technical Report RT-ICAR-PA-05-15. Dec. 2005.
[10] Cossentino M. 2005c. From Requirements to Code with the PASSI Methodology, In Agent-Oriented Methodologies, edited by B. Henderson-Sellers and P. Giorgini, Idea Group Inc., Hershey, PA, USA
[11] Di Marzo Serugendo G., Gleizes M.-P., Karageorgos A. 2006. Self-Organisation and Emergence in Multi-Agent Systems: An Overview - Informatica, An International Journal of Computing and Informatics, Vol. 30, N. 1, pp. 45-54, ISSN 0350-5596.
[12] Fortino, G., Garro, A., and Russo, W. 2004. From Modeling to Simulation of Multi-Agent Systems: an integrated approach and a case study. In Gabriela Lindemann, Jorg Denzinger, Ingo J. Timm, Rainer Unland, editors, Multiagent System Technologies, Lecture Notes in Artificial Intelligence (LNAI) volume 3187, pages 213–227, Springer-Verlag, Berlin Heidelberg, Germany.
[13] Henderson-Sellers, B. 2005. Creating a comprehensive agent-oriented methodology - using method engineering and the OPEN metamodel. In B. Henderson-Sellers and P. Giorgini, editors, Agent-Oriented Methodologies Idea Group.
[14] IEEE Recommended practice for architectural description of software-intensive systems, IEEE Std 1471-2000 , vol., no.pp.i-23, 2000
[15] IBM Eclipse - available at: http://www.eclipse.org/
[16] Management Coalition. Terminology & Glossary. Document WFMC-TC-1011, 3.0. Feb-1999. Online at: http://www.wfmc.org/ standards/docs/TC-1011_term_glossary_v3.pdf
[17] JaWE (the Enhydra Java Workflow Editor) – available on internert at: http://www.enhydra.org/workflow/jawe/index.html
[18] Jennings N. and Wooldridge M. 2000. Agent-Oriented Software Engineering. Handbook of Agent Technology. AAAI/MIT Press, J. Bradshaw, edition.
[19] Jess Rule Engine – available at: http://herzberg.ca.sandia.gov/jess/
[20] OMG, 2002, Software Process Engineering Metamodel Specification, Version 1.0, Object Management Group, formal/02-11-14 (Nov 2002)
[21] Juan, T., Sterling, L. and Winikoff, M. 2002. Assembling Agent Oriented Software Engineering Methodologies from Features. In Proc. of the Third International Workshop on Agent-Oriented Software Engineering, at AAMAS'02.
[22] Padgham L. and Winikoff M. 2003. Prometheus: A Methodology for Developing Intelligent Agents, Lecture Notes in Computer Science, Volume 2585, Jan 2003, Pages 174 - 185

[23] Pavón J., and Gómez-Sanz J. 2003. Agent-Oriented Software Engineering with INGENIAS, In Multi-Agent Systems and Applications III, (CEEMAS 2003) edited by V. Marík, J. Müller and M. Pechoucek, Springer-Verlag, LNCS 2691, pp 394-403
[24] Seidita, V., Cossentino, M., Gaglio, S. 2006. A repository of fragments for agent systems design. Proc. Of the Workshop on Objects and Agents (WOA06), Catania, Italy. September 2006.
[25] Shark (Java Open Source workflow engine based on XPDL) – available on internet at: http://www.enhydra.org/workflow/shark/index.html
[26] Wooldridge M. 1997. Agent-based software engineering, IEE Proc Software Engineering 144, pp. 26-37.
[27] Wooldridge, M. and Jennings, N. R. 1995. Intelligent Agents: Theory and Practice. In Knowledge Engineering Review 10(2)Workflow
[28] Workflow Management Coalition. Workflow Standard Process Definition Interface - XML Process Definition Language. Version 2.00. Document Number WFMC-TC-1025. October 3, 2005. Online at: http://www.wfmc.org/standards/docs.htm#Interface_1
[29] Zambonelli F., Jennings N. and Wooldridge M. 2003. Developing Multiagent Systems: the Gaia Methodology, In ACM Transactions on Software Engineering and Methodology, 12(3), pp. 417-47
[30] Waern A. and Gala, S. 1993. The commonkads agent model. ESPRIT Project P5248 KADS II KADS-II/M4/TR/SICS/004/V.1.0, Swedish Institute of Computer Science.