Adapting PASSI to Support a Goal Oriented Approach: a Situational Method Engineering Experiment

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

[1] Dipartimento di Ingegneria Informatica - University of Palermo, Italy
{seidita,gaglio}@dinfo.unipa.it
[2] Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo, Italy
cossentino@pa.icar.cnr.it

Abstract. The construction of ad-hoc design processes is more and more required today. Situational Method Engineering (SME) provides an approach with a set of activities to be accomplished during the construction of a new process. In this paper we present our approach for adapting SME for the construction of multi-agent systems design processes mainly focussing on the method fragments selection and assembly phases. The MAS metamodel provides the driving concepts of these two phases; in fact, we use MAS metamodel elements for enabling the retrieval of fragments from the repository and for reasoning about how to assemble them or eventually to modify them. An experiment is reported here we construct a new design process that includes a goal oriented requirements analysis phase starting from the existing PASSI process.

1 Introduction

Situational Method Engineering (SME) [14], provides means for constructing ad-hoc Software Engineering Processes[3] (SEP) following an approach basing on reuse of components (often called method fragments) coming from existing design processes. Our work is mainly focussed on the use of SME for the construction of customized multi-agent oriented design processes. *Method Fragment* is the core concept of SME; different well known approaches [5][11][15] present different definitions and descriptions of method fragment but all of them start from the assumption that whatever design process can be decomposed into (or it is composed of, if we use a bottom-up point of view) self contained components. One of the main principles of SME is that a method designer must have at his disposal a rich repository of components (or method fragments or simply fragments), coming from existing design processes, from it he can retrieve the best fragments for his own needs when he is creating a new customized design

[3] In this context we assume that the term Software Engineering Process is synonym of: methodology, design process or simply process, we will use all these words indifferently.

process. We principally ground our work on SME rationale and because our target is about multi-agent systems we needed to adapt/specialize this approach, however some differences exist between SME and our approach: one of the most important difference is that we use the Multi Agent System (MAS) metamodel for defining the structure of the system we will build by adopting the new SEP. The metamodel elements constitute a determinant input for the selection and the assembly of fragments extracted from our method base and above all they let the method designer identify the semantic differences among the same elements coming from different fragments. These differences inevitably imply the need for adapting/modifying the fragments for the assembly phase; in this paper we will principally focus on this aspect. Another important difference is that we start our design process construction from the assumption that in a given company some kind of SEP is always present, sometimes it is not documented, and not well-defined artefacts are produced so we use the CMMI practices [8] in order to classify the actual design process and to identify the specific goals and practices to be pursued in order to reach a specific maturity level; this results in a specific set of process requirements. In this paper we will present our experiment on the creation of a goal oriented design process following the proposed approach, specifically we decided to modify PASSI [7] and to make it suitable for a goal oriented requirements analysis. Our main requirements were to have an agent design methodology including the use of ontologies and communications with a FIPA compliant structure, so we decided to focus our work on PASSI, and to have a requirements analysis in a goal driven fashion. For this aims we chose to use some method fragments coming from Tropos [2] that we already stored in our repository and to merge them with those from PASSI in order to fulfill our requirements. We will point out how we based the selection of method fragments on the new design process metamodel and how we constructed it. The paper is organized as follows: in the next section we will describe our approach, in section 3 a detailed description of the way for building a new agent oriented design process is given and in section 4 we show how we applied it to a specific system pointing out the results of the assembly phase; finally we will provide some conclusions.

2 Situational Method Engineering for Agent Oriented Methodologies Construction

According to [10] a method engineering process is composed of three main phases: the *requirements specification* of a project specific method, the *selection* of method fragments from the repository basing on the results of the previous phase and the *assembly* of the selected fragments. These three phases are the core of our approach that is reported in Figure 1, here we can identify three main phases: *process analysis, process design and process deployment.*
Process Requirement Analysis deals with requirements elicitation and analysis, it produces a set of elements affecting the *Method Fragments Selection* and *Assembly* activities. Finally in the *Process Deployment* phase the new SEP is

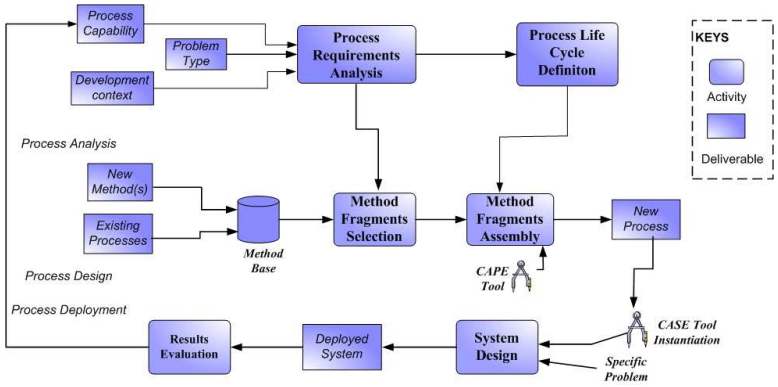

Fig. 1. The proposed approach for Agent Oriented Methodologies Construction

instantiated, used to solve a specific problem and then evaluated. The results of the evaluation are useful for defining new requirements for the next SEP (if any).

It is worth to note that we consider the process of defining a new design process as an iterative and incremental one. In the following subsections we will provide a detailed description of the adopted agent oriented methodology construction approach.

Process Analysis. *Process Requirements Analysis* is the first activity a method designer undertakes in his work. It has inputs coming from the maturity level of the organization, the development context (tools, languages, available skills, etc.) and the type of problem to solve. These inputs are used to define the process life cycle (that establishes the structure the designer has to follow during method fragments assembly activity), the system metamodel concepts and the other process elements (available stakeholders, required activities or work products) that are used for selecting the method fragments from the repository (*Method Base*).

Process Capability: it is the concept defined in the SEI Process Capability Maturity Model Integration (CMMI) for Development [8]: "Software process capability describes the range of expected results that can be achieved by following a software process".

The software process capability of an organization provides a means for predicting the most likely outcomes to be expected from the next software project the organization undertakes. In this way it is well defined how to work for achieving fixed objectives. In our work the identification of these activities results in a well defined set of requirements on method fragments to be selected or on specific stakeholders to be involved in the process.

Problem Type: the new process has to be tuned for a specific solution strategy to a class of problems.

It is possible that, in a big company, different groups produce software for totally different areas (for instance business administration, and biological systems sim-

ulation). In this situation it should be expected that each of these groups adopts a different SEP giving the right importance to the aspects that are more sensible in its target implementation domain.
Development Context: it is a description of the available resources (both human and non human) and competencies that are available in the SEP enactment group. The development context is usually a sensitive aspect to be considered also because if the group is composed of people skilled with some specific approach or standard practice (for instance the use of UML in modelling the system), it is highly desirable to capitalize such an experience in order to lower training costs that always follow the introduction of a new SEP. In the development context we also enumerate possible constraints that could come from available developing tools.
Process Requirements Analysis: from the aforementioned issues, the method designer achieves the necessary inputs for defining some fundamental requirements about the new SEP to be built. These requirements define the domain of interest that the new SEP should take into account.
For example, if the problem type deals with transportation of human beings and if someone in the development group has formal methods practice then some safety properties of the system may be proved.
The elicited requirements also provide some elements useful for the selection of method fragments: they are the system metamodel and process elements.
The metamodel contains all the concepts that can be used to design and describe the system to be: it defines domain-specific concepts, solution concepts and all the concepts that specifically address the characteristic of the particular system a designer is developing, together with all of their relationships.
For instance in the case of a MAS (Multi Agent System), a metamodel represents concepts such as agent, role, communication, agent task, and so on. Each concept of the metamodel must be designed/defined (that means instantiated) at least in one fragment of the process (whereas it can be refined or cited in several other fragments). In this way we can use the list of metamodel concepts for the selection of method fragments from the repository [16][5]; the designer, for each activity (like requirements analysis or detailed design), firstly selects the concepts of the metamodel to be designed (for instance scenarios or functional requirements) and then he uses this information, and other elements that we will explain later, in order to retrieve the most useful method fragments for the assembly activity.
A design process defines when and how someone does something in order to reach a specific objective [13], so the process requirements analysis also results in a list of elements composing the new SEP. These elements can be activities (the work to be done), process roles (particular stakeholder performing the work) and work products (artefacts resulting from some activities) and they too can be used for the retrieval of method fragments [16]. *Process Life Cycle Definition*: it is concerned with the decision about the process model (or life-cycle) to be adopted; this decision is influenced by several factors (for instance contract constraints imposed by the customer/commissioner on that). According to

some studies it seems that the process life-cycle is not affected by the adoption of the agent paradigm and therefore classical life-cycles (waterfall, spiral, iterative/incremental, etc.) can be used for designing agents too [4].
Process Design. This phase presents two activities: the *Selection* and the *Assembly* of method fragments taken from a repository. The *method fragment* is the building block of process design; it is extracted from existing design processes, or created from scratch, and stored in a repository, called method base from which it is selected basing on the results of requirements analysis.
Figure 1 shows that a method fragment can be extracted from existing design processes or created/modified to meet a specific requirement of the new process. The configuration of our actual fragments repository and how to use it for selecting the fragments have been already discussed in [16].
The *method fragments assembly* activity results in the new SEP. This activity consists in putting together the selected method fragments following the structure of the identified process life cycle on the base of specific assembly techniques. This activity is still one of the most important unresolved points in the SME field and some proposal have been done in [15][3], it is a very complex work where the method designer has to collate all the elements gathered in the previous activities and to merge them using his experience and skills. We think that in some cases a set of method fragment can be directly associated each other, in other cases they need modifications of one (or more) constituting elements. For instance if two method fragments adopt different semantics for the produced work products, it is necessary to change (or adapt) the elements of one work product kind to the other to allow a right assembly, or if the process part of two different method fragment overlap then one, or both of them, must be modified in order to create a unique consistent process.
Process Deployment The system designer adopts the new process with the aid of a CASE tool for solving a specific problem. After that the designed system is used and experimented, a results evaluation activity occurs in order to measure and evaluate the new process, according to the CMMI model. Gathered information can be used as a new process requirement for a next iteration (if necessary).

3 Constructing a Goal Oriented Design Process

In this section we will illustrate how we modified/adapted an existing agent design methodology, PASSI [7], in order to create a customized agent-oriented methodology meeting a specific requirement: providing designers with a methodology supporting goal oriented requirements analysis and maintaining the skills and background of all the designer that have been PASSI user since now.
This experiment was carried out by following the approach depicted in section 2, here one of the key point is the construction of the new design process metamodel to which elements we refer for the selection and the assembly of method fragments from existing methodologies [16][5].
Different agent oriented design processes present metamodels where we can often

find the same element, different elements with the same meaning and vice versa same elements with different meanings; the same situation can be found among the metamodel elements of the method fragments stored in the repository [16][5]. This may be a great limit when a designer wants to construct the new design process because the semantic differences among metamodel elements from different methodologies hardly affects the assembly activity. That is why the construction of the new metamodel has to be carried out in conjunction with a punctual definition of each metamodel elements.
Since in this work we focussed on the adaptation of PASSI, we started the construction of the new metamodel from the PASSI one; the process requirements analysis pointed out that first of all we had to remove the elements designed during the requirements analysis phase and to replace them with the new ones introduced for supporting a goal oriented requirements analysis.
The PASSI metamodel can be found in several works [6], in this paper (Figure 2) we only show the portion of metamodel subjected to modifications. In PASSI the requirements analysis phase leads to the agents identification which some system functionalities are assigned to, so the elements that have to be taken into account (Figure 2) are: *Actor*, *Requirements* and *Scenario* concerning the system's features and the entity exchanging information with it, *Agent, Role, Task* and *Plan* representing the entities that perform a set of actions in order to reach a system objective, *Resource*, what the agent uses and *Dependency* allowing to create the society of agents pursuing the same aim.
In goal-oriented requirements engineering field the system under construction (the *system-to-be*) is usually composed of the software and its environment, the latest being composed of humans, devices and other software systems; these are considered active components that can decide on their behaviors and they are called *agent*[4]. Agents depend each other for the goals they want to achieve, the tasks they perform to achieve the goals and the available resources they can use, besides an agent may use plans to achieve a goal. A goal is the objective a system has to achieve in cooperation with the agent in the system to be and in the environment; goal, agent and scenario are strictly related [18] and all contribute to the precise specification of software behavior.
What we have just said corresponds to a summary of the process requirements elicitation phase (see section 2) we carried out at the beginning of our work; from this analysis we realized that the elements we need in our new design process and that should be introduced in the new metamodel are: goal, agent, environment, resource, services, task and plan.
For the reasons we said before we decided to reuse fragments coming from Tropos [9]; Tropos was conceived for providing a deep understanding of the environment where the system will operate and for modeling agent oriented software architectures using the concept of agent, goal and plan.
We explored the Tropos metamodel in order to find which elements are designed during requirements analysis and we found that the following elements have rela-

[4] Note that in this context agent is not only a software entity and its significance is much more complex than the PASSI agent one.

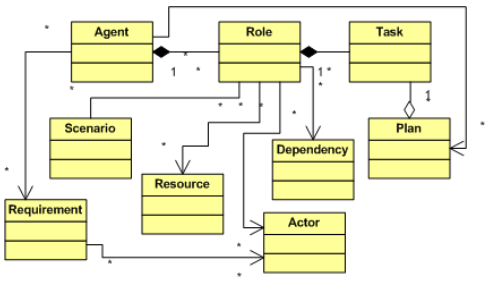

Fig. 2. PASSI Metamodel-Requirements View

tionships with those identified during the previous process requirements analysis activity: actor, goal, position, plan, task, dependencies and resource.
Referring to the PASSI metamodel (Figure 2) we can see that some of these

Table 1. Requirements Analysis Elements from PASSI and Tropos.

Deleted from PASSI	To be modified	Inserted from Tropos
Requirement, Scenario.	Actor, Agent, Role, Task, Plan, Resource, Dependency.	Goal, Position.

elements are not present and we had to introduce them whereas for others we had to check, both in PASSI and in Tropos, their definitions in order to point out similarities and differences.
Table 1 resumes which elements we added ex novo, which elements we deleted from the PASSI metamodel and which one we had to examine for eventually changing their definitions and relationships.
In PASSI, *Requirement* represents a feature that the system to be must exhibit and *Scenario* is the concrete, informal description of a single feature of the system, they had to be deleted because now the system has to be seen according to an "objectives to be pursued" point of view.
In Tropos [2] a *Goal* is a strategic interest of an *Actor* that is an intentional entity, it can be a *Role*, a *Position* or an *Agent*, besides an *Actor* depends on another to accomplish a *Goal*, deliver a *Resource*, or execute a *Task* that is a particular set of actions used to satisfy a goal.
The different definitions of the metamodel elements of the two design processes are shown in Table 2; these led us to the conclusion that since in Tropos the concept of agent is strictly related to the environment the system to be has to work in and it can also be an active entity then it can be assimilated to the agent in PASSI; the same we can say for Resource, Actor and Role, but with different

Table 2. Corresponding PASSI and Tropos elements definition.

Elements	PASSI	Tropos
Actor	External entity exchanging information with the system.	Intentional entity: role, position, agent.
Agent	It is an entity responsible for accomplishing some functionalities.	A specialization of actor.
Role	It is a portion of the social behavior of an agent.	A specialization of actor.
Task	A task is an entity that aims to reach a sub-goal (for instance dealing with a communication); an agent uses tasks to execute its plan(s).	Particular course of action that can be executed in order to satisfy a goal.
Plan	The set of tasks an agent performs to reach an objective.	A set of actions.
Resource	A concrete, tangible entity that can be acquired, shared, or produced by agents.	Physical or informational entity (without intentionality); it can be used to achieve goals and/or to perform tasks and can be shared by agents.
Dependency	A relation between two roles of different agents, which indicates that one agent depends, for some reason, on the other in order to attain some goal or deliver a resource.	A relation between two actors, which indicates that one actor depends, for some reason, on the other in order to attain some goal, execute some plan, or deliver a resource.

relationships among them coming from the Tropos metamodel.
We had to make some reasoning about Task and Plan; in fact a Task in PASSI is related to a particular behavior and the agent performs his action following a set of plans whereas in Tropos task and plan someway have the same meaning both being the actions an agent makes to pursue a goal. Besides the task modelling activity, in the Tropos requirements analysis phase, represents a deeper view of the goal analisys; so it is a concept more related to problem domain rather then to agent domain. We had to separate these two concepts, maintain the concepts of task and plan as they are in the PASSI metamodel and insert a new concept named *Problem Task* to represent the Tropos concept of Task so facing this particularity.
The same we made for Dependency, we had to separate the two domains it belongs to in each design process, the problem domain and the agency domain; we maintained the PASSI vision of Dependency and inserted a new concept, Actor Dependency, for representing Tropos Dependency.
All this affected the use of some selected fragments, as we will illustrate later, some of them needed some modifications and adaptations for fully reflect the

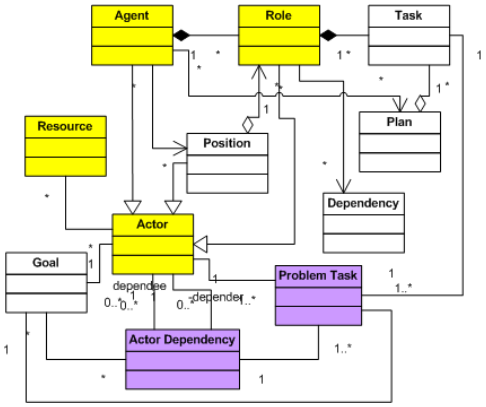

Fig. 3. The New Design Process Metamodel

changes made in the definitions.
We obtained a new metamodel where each element has a precise meaning related to the context the new design process has to be applied to; apart of the resulting metamodel is shown in Figure 3. Once we established which elements had to be inserted in the new metamodel we looked at the Tropos fragments for selecting the more appropriate ones for our aims; from *Early Requirements Phase* we identified:

- the *Domain Description fragments*, here the designer analyzes the problem statement from which he identifies a set of actors involved in the system under construction and the related goals; then the goals are decomposed through AND/OR-decomposition. The resulting work product of this fragment is the Actor Diagram where all actors and their main goals are depicted;
- the *Domain Analysis fragment*, where the Actor Diagram is extended in order to identify all the tasks (plans) each actor has to perform to pursue a specific goal, and then through the means-ends-analysis each task is relate to at least a goal. The resulting work product is the Goal Diagram describing the set of actors, their goals and tasks and the dependencies among actors.

from *Late Requirements Phase*:

- the *Identify System fragment*, where, starting from the previously produced Actor Diagram and Goal Diagram, the actor System-to-be is identified with his goal and tasks (in the same way as in the previous two fragments) and more goals and tasks are detailed. The result is a Actor Diagram named System Actor Diagram.
- *Describe Environment fragment*, here the System Actor Diagram and the Goal Diagram are used to find all the system's actors and goals that can be assigned to the System-to-be actor identifying, in so doing, the dependencies between the System actor and all the other actors.

from *Architectural Design Phase*

- the *Identify Architecture fragment* where the System actor is decomposed into sub-actors which goals are assigned to; this fragment results in an Actor Diagram named sub-Actor Diagram from which agents can be identified; generally each sub-actor becomes an agent.
- the *Define Agent Society fragment* where one or more capabilities, "the ability of an actor of defining and executing a plan for the fulfillment of a goal", are assigned to each identified agent; the resulting work product is a UML activity diagram.

Table 3 summarizes which element of the metamodel is defined in each selected fragment and which input/output the framents need for their assembly. All these fragments constitute the System Requirements Analysis phase, see Figure 4; the first five fragments were obviously assembled (it was a re-assembly activity in this specific case, since they all come from Tropos) without problems and cover all the new metamodel elements except: Task, Plan, Dependencies and Role.
Let us now consider Task and Plan; in PASSI the Task (of an agent) is defined in the *Task Specification* fragment where, focusing on agent behavior, a plan is conceived also describing the dependencies with other agents; so tasks are identified after the identification of agents. We reused from Tropos the *Identify Architecture* fragment resulting in an actor diagram where agents are identified with their goals and the tasks they use to achieve them, but at this stage we still have the actors' tasks that in the new metamodel we named "Problem Task". In the *Define Agent Society* fragment the identified capabilities are the set of actions an agent undertakes to follow the plan it is performing in order to pursue a specific goal; looking at the definitions of Table **??** in this fragment this fragment we can identify the link between those elements (*Task* and *Problem Task*) we had to specialize and distinguish between problem and agency domains. Thus this fragment was modified and adapted in order to relate the two elements; specifically the adaptation consisted in producing the fragment result in the form of a PASSI task specification diagram, that is itself a UML activity diagram, without focussing on the dependencies among agents, only an agent at time is considered.
We made a kind of merging between the PASSI *Task Specification* fragment [1] and Tropos *Define Agent Capability* fragment taking into account the definitions of the two metamodel elements they define and the kind of work products they produce. The resulting work product of the new fragment was a UML activity diagram where for each agent task a plan is depicted; here we can see that the *Problem Task* we have defined in the *Identify System* fragment is refined and linked (as we can see from metamodel in Figure3) to the task of the agent.
As regard defining the Role and Dependency elements we decided to select the *Roles Identification* fragment from PASSI because there the roles each agent plays is defined and also we can identify the dependencies among agents, but, as we can see from Figures 2 and 3, in PASSI the concept of *Role* is related to that of *Scenario* whereas in the new metamodel this latter concept is not present, and since it is the input for *Roles Identification* fragment, this needed

to be modified. We derived only the rationale underpinning the concept of role and we merged it with the description of Tropos agent interaction, the result was a UML sequence diagram where the interaction (the Dependency) among agents is represented pointing out the different role they play when performing a set of actions.
In the next section we will detail this part of assembly activity through an example. In Figure 4 the new design process is shown; the lower part shows, through a SPEM [17] activity diagram, the new fragments (each of them is represented by an activity) we assembled and details for each of them the related work product.

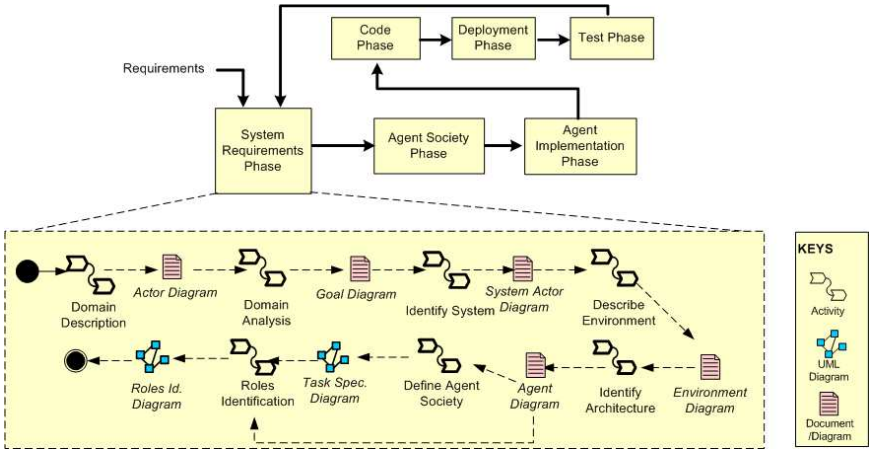

Fig. 4. The new Design Process

4 Results Evaluation

In order to empirically evaluate the resulting design process, we used it for the development of an agent based platform for the simulation of collaborative models in the field of automotive industry. The aim of this platform is to support the planning phase of engineering activities in a collaborative fashion thus enhancing the quickness and the effectiveness of automotive development process through an intense involvement of the suppliers.
The platform was conceived for the distributed collaborative new product development and it is composed of three sub-systems: *i)* First Level Suppliers mapping, *ii)*Collaboration Choice management and *iii)* Collaboration mode management.
In the rest of this section we will illustrate some details of the second sub-system design, and the work products resulting from the application of some fragments in order to exemplify the result of using the metamodel elements in the assembly

Table 3. Fragments Related to the New Metamodel Elements.

Name	Objective	Input	Output	MMM
Domain Description	To identify the actors involved in the system and their goals.	A verbal description of the problem to be faced.	Actor Diagram (Tropos Diagram).	Actor, Goal.
Domain Analysis	To identify the tasks of each actor and to apply means-and-analysis.	Actor Diagram.	Goal Diagram (Tropos Diagram).	Problem Task, Actor Dependency, Resource.
Identify System	To Identify the System-to-be actor.	Goal Diagram.	System Actor Diagram (Tropos Diagram).	Actor, Goal, Problem Task.
Describe Environment	To assign all the system actors' goals to the System-to-be.	System Actor Diagram.	Environment Diagram (Tropos Diagram).	Actor Dependencies.
Identify Architecture	To decompose the System-to-be into sub-actors and to identify agents.	Environment Diagram.	Agent Diagram (Tropos Diagram).	Agent.
Define Agent Society	To identify a set of capability for each agent in order to establish which plans they have to follow.	Agent Diagram.	Task Specification Diagram (UML Activity Diagram)	Task, Plan
Roles Identification	To identify the roles each agent plays and the dependencies with other agents.	Task Spec. Diagram.	Roles Identification Diagram (UMl Sequence diagram).	Role, Dependency.

phase.

In Figure 5 we show, for space concerns, a reduced view of the work product resulting from the *Domain Analysis* fragment, it is an Actor Diagram where we identified the actors involved in the system (the main contractor MC and the Supplier) and some of the goals and the tasks (problem tasks in this case) they have to achieve. After this activity we proceeded with the system identification which resulted in the identification of the Platform system actor with its dependencies with other actors.

Once the System actor was identified we could decompose its goals and assign each of them to an agent (*Identify Architecture fragment* whose resulting work product is shown in Figure 6, the yellow ball with the horizontal line is an actor); then for each agent and for each goal we detailed, through a means-end-analysis all the tasks an agent has to perform. Let us consider the *Request Clustering* (see Figure 6) without focussing on the inner meaning of the task itself, it represents the capability of the *WBS Manager* agent to pursue the *Start Clustering* goal and it is composed of the activities shown in Figure .

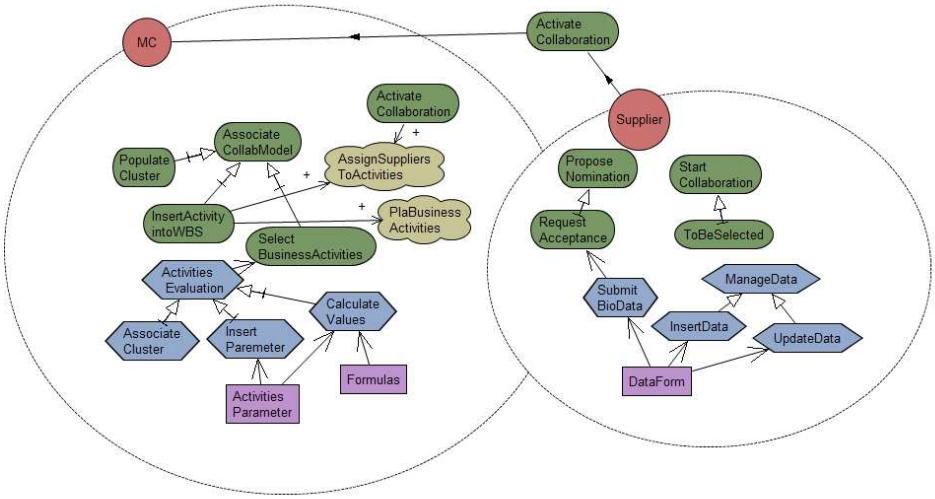

Fig. 5. The Goal Diagram

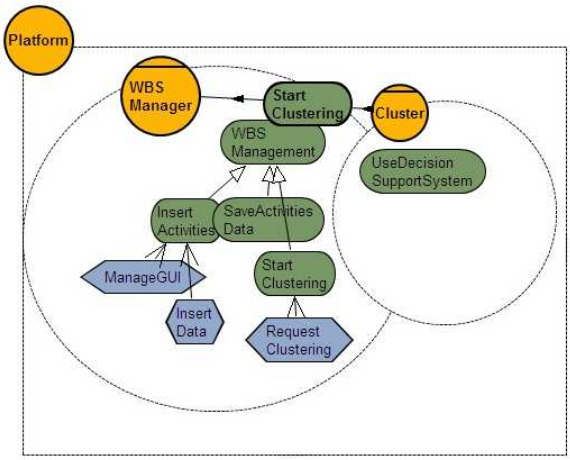

Fig. 6. The Agent Diagram

Figures 7.a) and 7.b) respectively represent the task specification diagram and the agent interaction diagram resulting from the *Define Agent Society* fragment that is the one we merged with PASSI *Task Specification* and *Role Identification*; they only differ, from the work product produced in the original fragments, in the fact that now in the task specification diagram we do not show that interaction with other agents, so as in the *task specification* one, that is however inferred from the specification diagram that presents the same information of

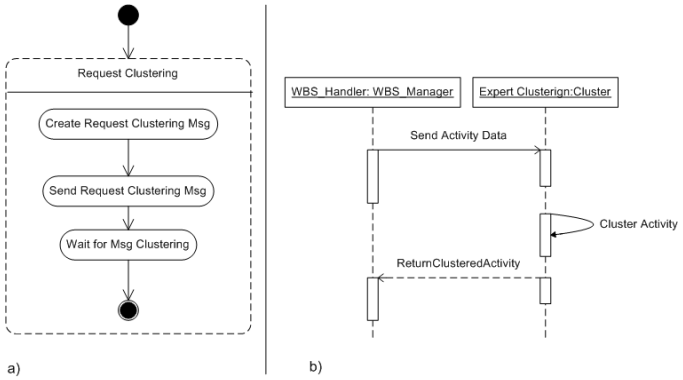

Fig. 7. Define Agent Society and Roles Identification Work Products

*Role Identification*diagram. These two figures clearly show that the creation of the new design process using the approach based on metamodel was in this case succesful because we could integrate two great portions of two methodologies maintaining the traceability of the definition of metamodel elements along the whole process even using different CASE tools, for the first fragments in fact we used TAOM4E tool, the agent oriented modelling environment providing support for Tropos and for the rest of design process we used Rational Rose.

5 Conclusion

In this paper we presented an experiment of using SME for the construction of ad-hoc agent oriented design processes; the core concept of this approach is the MAS metamodel of the constructing design process whose elements are used for the right retrieval of method fragments from the repository and for their right assembly. We used our approach for experimenting the construction of a design process basing on specific requirements: a goal oriented agent design process based on the main structure of PASSI. In the work done we adapted PASSI merging it with some fragments coming from Tropos.
The new process was then used for the design of a real system letting us to validate our choices and to point out that the key point during fragments selection and assembly activities is the specific semantic of the metamodel element that each fragment underpin; a deep knowledge on the meaning of the metamodel elements let the method designer choose the most useful fragments and adapt/modify some of them for a precise need.
This is the second experiment we made in design process composition (the first was Agile PASSI [6]) and we are planning to do another experiment on the creation of a new design process, this time creating from scratch its metamodel. We trust in this way to learn more and more about assembly techniques and to be able to specify guidelines and to create a support for automatic composition

of fragments; until now in fact the selection and assembly technique are mostly based on the skills and the knowledge a specific designer has on the fragments repository he uses.

References

1. M. Cossentino an L. Sabatucci and V. Seidita. Method fragments from the passi process. *Rapporto tecnico ICAR-CNR*, (21-03), 2003.
2. Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Tropos: An agent-oriented software development methodology. *Autonomous Agent and Multi-Agent Systems (8)*, 3:203–236, 2004.
3. S. Brinkkemper, M. Saeki, and F. Harmsen. Meta-modelling based assembly techniques for situational method engineering. *Information Systems, Vol. 24*, 24, 1999.
4. L. Cernuzzi, M. Cossentino, and F. Zambonelli. Process models for agent-based development. *Engineering Applications of Artificial Intelligence*, 18(2):205–222, 2005.
5. M. Cossentino, S. Gaglio, A. Garro, and V. Seidita. Method fragments for agent design methodologies: from standardisation to research. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 1(1):91–121, 2007.
6. M. Cossentino, S. Gaglio, L. Sabatucci, and V. Seidita. The passi and agile passi mas meta-models compared with a unifying proposal. In *In proc. of the CEEMAS'05 Conference*, pages 183–192, Budapest, Hungary, Sept. 2005.
7. Massimo Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies* [12], chapter IV, pages 79–106.
8. SEI. CMMI for Development Version 1.2. *Technical Report of the Software Engineering Institute of the Carnagie Mellon University.*, August 2006.
9. Paolo Giorgini, Manuel Kolp, John Mylopoulos, and Jaelson Castro. Tropos: A requirements-driven methodology for agent-oriented software. [12], chapter II, pages 20–45.
10. D. Gupta and N. Prakash. Engineering Methods from Method Requirements Specifications. *Requirements Engineering*, 6(3):135–160, 2001.
11. Brian Henderson-Sellers. Method engineering: Theory and practice. In *ISTA*, pages 13–23, 2006.
12. Brian Henderson-Sellers and Paolo Giorgini. *Agent Oriented Methodologies.* Idea Group Publishing, Hershey, PA, USA, June 2005.
13. I. Jacobson, G. Booch, and J. Rumbaugh. *The unified software development process.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.
14. K. Kumar and R.J. Welke. Methodology engineering: a proposal for situation-specific methodology construction. *Challenges and Strategies for Research in Systems Development*, pages 257–269, 1992.
15. I. Mirbel and J. Ralyté. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11(1):58–78, 2006.
16. V. Seidita, M. Cossentino, and S. Gaglio. A repository of fragments for agent systems design. *Proc. Of the Workshop on Objects and Agents (WOA06)*, 2006.
17. SPEM Specification. http://www.omg.org.
18. A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Roundtrip from Research to Practice. *Proc. 12th IEEE Intl Requirements Engg Conference, Kyoto*, pages 4–8, 2004.