A MAS metamodel-driven approach to process composition

M. Cossentino[1,2], S. Gaglio[1,3], S. Galland[2], N. Gaud[2], V. Hilaire[2], A. Koukam[2], and V. Seidita[3]

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo, Italy
cossentino@pa.icar.cnr.it
[2] Systems and Transport Laboratory (SeT) - Belfort, France
stephane.galland,nicolas.gaud,vincent.hilaire,abder.koukam@utbm.fr
[3] DINFO - Università degli studi di Palermo, Palermo, Italy
gaglio,seidita@dinfo.unipa.it

Abstract. The construction of ad-hoc design processes is more and more required today. In this paper we present our approach for the construction of a new design process following the Situational Method Engineering paradigm. We mainly focus of the selection and assembly activities on the base of what we consider a key element in agent design processes: the MAS metamodel and its elements.

1 Introduction

Multi-Agent systems metamodels (MMMs henceafter) and the composition of new design process achieved greater attention in the last years in the agent community for different reasons. As regards MMMs it is evident that their study enabled a deeper understanding of underlying design processes while, at the same time, the growing importance of Model Driven Engineering approaches required a great effort in the study and modelling of systems on the basis of their metamodels. Besides, the need for a continuous evolution of design processes in order to meet new requirements, support new developing platforms and paradigms increased the effort spent on the study of techniques that could ensure an affordable way for the production of the right new design process in a specific situation and development context for solving a specific class of problems. In this field, Situational Method Engineering (SME) [1], provides means for constructing ad-hoc Software Engineering Processes (SEP) following an approach based on the reuse of portions of existing design processes (often called method fragments). Our work is mainly focused on the use of SME [2–4] for the construction of customized multi-agent oriented design processes with the significant difference that we use the MMM for defining the structure of the system we will build by adopting the new SEP.

In this paper, we mainly focus on the importance of the definition of the MMM in the selection of method fragments that will constitute the new SEP.

This article introduces our approach to SME application and report an experiment of creation of a new process (called ASPECS[4]) this is not a classical toy problem but rather we are dealing with the construction of a large processes for the design of agent-oriented systems. The scope of this process will be the design of dynamic hierarchical societies of agents; we aim at using the process and the related implementation platform for realizing open, dynamic holonic systems (Janus [5]]) and solving complex problems requiring a huge number of agents. The MAS metamodel of this new process [5] is mainly composed by elements coming from the PASSI [6] and CRIO [7] existing design processes and supports Janus as an implementation platforms of holonic agents.

The paper is organized as follows: the next section gives a brief description of the proposed approach. Section 3 lists the requirements from which we started for developing the new process and quickly overview the resulting ASPECS design process. Section 4 reports the experiment done. And finally some conclusion statements are provided in section 5.

2 The Proposed Approach

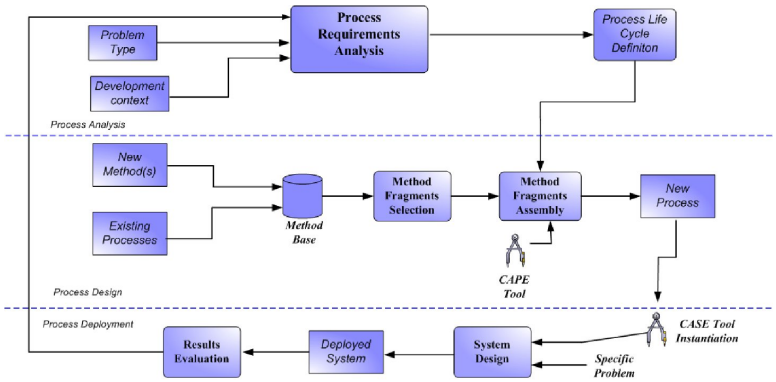

Fig. 1. The proposed design process composition approach.

The proposed design process composition approach shown in Figure 1 is organized in three main phases: process analysis, process design and process deployment.

Process Analysis deals with requirements elicitation and analysis of the process to be developed. It produces a set of elements mainly a portion of the MMM affecting the Method Fragments Selection and Assembly activities. Finally in the Process Deployment phase the new SEP is instantiated, used to solve a problem and then evaluated. The results of the evaluation are useful for defining

[4] ASPECS: Agent-oriented Software Process for Engineering Complex Systems.

new requirements for the next SEP (if any) or improving the designed one. It is worth to note that we consider the process of defining a new design process as an iterative and incremental one.
Process Requirements Analysis is the first activity a method designer undertakes in his work. It has inputs coming from the type of problem to solve. The new process has in fact to be tuned for a specific solution strategy to a class of problem, and the development context that is a description of the available resources such as human, tools, languages, available skills, and competencies that are available in the SEP enactment group.
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) used for selecting the method fragments from the Method repository.
The metamodel contains all the concepts that can be used to design and describe the system under study. 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. Each concept of the metamodel must be designed/defined 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 *method fragments selection* from the repository [8][9]. For each activity, the designer firstly selects the concepts of the metamodel to be designed (for instance scenarios or functional requirements) and then he uses this information in order to retrieve the most useful method fragments for the assembly activity. The method fragments are extracted from existing design processes or created from scratch.
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. This activity is still one of the most important unsolved points in the SME field and some proposal have been done in [10][11]. 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.
During *Process Deployment* the system designer adopts the new process with the help of a CASE tool for solving a specific problem. After that the designed system is used and experimented, an evaluation activity occurs in order to measure and evaluate the new process and the gathered information can be used as a new process requirement for a next iteration.
In the construction of the ASPECS methodology we applied the process that has been described before with a specific study on the procedure adopted for the definition of the MAS metamodel and the method fragments selection. We regard this part of the work as one of the main scientific contributions of our approach and therefore we are going to detail it. In this paper the main focus is on the early construction of a core part of the MAS metamodel and then on its use as a guide towards the selection and assembly of process fragments. The pro-

cedure we defined (Figure 2) starts from the identification of the core part of the MAS metamodel and is done by evaluating the contributions that could come from existing design processes or development platforms (in our case they were PASSI, CRIO, JANUS). In fact it is logical to expect that people already skilled with the concepts related to some existing process or platform prefer to reuse them rather than to build everything from scratch. Parts of those metamodels have been reused in order to satisfy the new process requirements that will be described in the experimental part of the paper (section 4), by now, just to exemplify, it is sufficient to consider that we reused part of the PASSI metamodel because we aimed at obtaining a FIPA-compliant communication structure. In the following subsections we detail the most important steps of this process: the new MAS metamodel construction and the new process design phase where fragments are retrieved from repository and assembled.

2.1 Core MAS metamodel definition

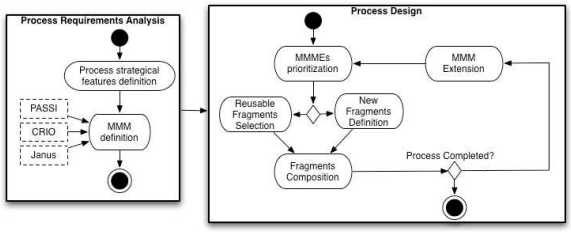

Fig. 2. Details of the process requirements analysis and process design phases presented in Figure 1.

As already said, in this work we composed the new metamodel on the basis of portions of metamodels coming from PASSI, CRIO and Janus. In so doing we are aware that defining the core MAS metamodel means defining a relevant part of the 'philosophy' that will be behind the new design process. For this reason we performed this activity during meetings involving stakeholders. We tried to deduct the list of elements by the portions of the cited processes that could satisfy the new process requirements. Of course this was not sufficient and it was therefore necessary to add new concepts for dealing with the specific case. For instance a lot of work has been done (while building the ASPECS process) in the definition of the agent societies organizational structure as well as on the specification of possible roles that could be played by agents in an holon (Head, Representative, Part/Multipart and StandAlone). These are crucial choices that conditioned the entire process and they have been largely debated before adoption.

The work for designing the new process based on the defined core metamodel can be represented as a cycle composed of three subphases as illustrated in Figure

2: (i) prioritization of MAS Metamodel Elements (MMMEs); (ii) identification and assembly of method fragments defining the MMMEs; (iii) extension of the metamodel until the complete process is defined. The process is detailed in the following algorithm:

```
//Subphase 1: MAS metamodel elements prioritization
1.  Select a metamodel domain (in the order: problem, agency, solution) and consider the
      resulting metamodel as a graph with nodes (MMMEs) and directed links (relationships)
2.  Define List_elements as a list of MMMEs and associated priority p: List_elements (MMME, p
     );
       a.  p<-1;
       b.  List_elements <- null;
3.  Produce a linearization of the MMMEs nodes according to a topological sort in
      List_elements, p is the index of each node in the list
// Subphase 2: Assembly of fragments related to the core MAS metamodel
4.  Select/Build fragments for defining (i.e. instantiating) the selected MMME(s) by doing:
       a.  i<-1;
       b.  Selected_el<-List_elements.select(i);
       c.  if Selected_el.count>1 then select one element according to the easiest
             identifiability of reusable fragment or new fragment creation
       d.  select/build fragment for element Selected_el.
5.  Assembly the fragment in the new process (eventually modify it if required)
6.  Select the next metamodel domain (if any) and repeat from 2
//Subphase 3: MAS Metamodel Extension
7.  If the process is not completed (i.e. not all design activities from requirements
     elicitation to coding, testing and deployment have been defined)
       a.  Introduce new MMMEs according to criteria discussed below
       b.  Repeat from 1.
```

It is worth to note that the previously defined algorithm is based on the following assumptions:

- MMMEs are organized in three domains: problem, agency, solution. In the first domain we put elements belonging to the model of the problem in terms of requirements, in the agency domain we collect elements defining an agent-based solution to the problem defined in the previous domain, in the solution domain we list elements related to the implementation of the solution in one or more available platforms.
- In each method fragment, four different actions can be done on (one or more) MMMEs: new MMME definition (instantiation), creation of new relationships among MMMEs, existing MMME quotation, existing relationship quotation.
- Each method fragment has a concrete, tangible output in form of one (or more) workproduct(s) belonging to the same work product type such as structural/behavioral diagram, text document, composite document.

The extension of the core MAS metamodel towards the completion of the process obtained by composing fragments, is a crucial activity that should be strongly affected by the awareness of the new process requirements and the relationships among requirements and MMMEs. In extending the initial core metamodel some other criteria should be considered: First, opportunity of reusing some existing fragments could lead to the introduction of MMMEs related to them. This is a kind of bottom-up criterion that privileges the reuse of well known and tested fragments. Second, as a consequence of adopting a Model Driven Engineering

(MDE) approach, we agreed that: (i) the three identified MAS metamodel domains should have been regarded as the three different MDE models; (ii) elements belonging to a domain should have a correspondent element in the following one (correspondence is 1-to-1 and it is realized by the transformation from one model to the other). The second rule can have some exceptions related to specific cases when an element is regarded as a design abstraction useful at one specific level but it is not forwarded to the next one.

3 The ASPECS process: requirements and results

This section presents the requirement under which the ASPECS process has been developed and the resulting process itself with the aim of short-circuiting the beginning and the end of the process we discussed so far.

3.1 Requirements for the construction of ASPECS

The design of the ASPECS methodology has been constrained by a set of requirements that according to the inputs of the process requirements analysis phase presented in Figure 1, can be classified as follows:
(i) *Problem Type*: the scope of the new design process was defined to be the development of very large MASs for the solution of complex problems suitable for an hierarchical decomposition.
(ii) *Development context*: the development of the ASPECS methodology can be seen as a joint work of people coming from two different experiences: people working at the SET laboratory who had a strong background in the design and implementation of holonic systems with a strong accent on organizational aspects of MASs (CRIO process) and one new lab member who was the main author of a process (PASSI, [6]) for the design of MASs where agents were mostly peers and whose important features were: the use of ontologies, a requirements-driven agent identification, the adoption of patterns and tools for supporting design/-coding activities. Participants to this project soon agreed to preserve some key elements of their backgrounds and skills in order to enable an easier transition to the new design process. As regards agents implementation, in the SET lab, the development of a new coding platform Janus was undergoing and its adoption in the new design process was, of course, highly desirable.
(iii) *Organization maturity*: several experiments of development of holonic systems have been previously performed in the lab but each single project adopted a different implementation solution or design strategy so that a unique consolidated design process was not available (CRIO only covered a part of the process). Conversely, as regards the experiences coming from PASSI, a complete documentation of the process was available, a large number of projects have been already developed and a large experience of usage of the process and the related guidelines/tools was available.

These requirements concurred to the definition of the process we describe in the next subsection.

3.2 The resulting design process

ASPECS combines an organizational approach with an holonic perspective. Its target scope can be found in complex systems and especially hierarchical complex systems. The principle of ASPECS consists in exploiting the intrinsic hierarchical structure of complex systems to analyze and decompose them. The process of ASPECS consists in four phases that are briefly described below.

The *Analysis* phase is based on the identification of a hierarchy of organizations, which the global behavior may represent the system under the chosen perspective. It starts with requirements analysis and requirements are identified using classical techniques such as use cases. Domain knowledge and vocabulary associated to the target application are then collected and explicitly described in the problem ontology. Each requirement is then associated to an organization, that represents a global behavior able to fulfill the associated requirements. The context of each organization is defined by a set of concepts of the problem ontology. The organization identification gives raise to a first hierarchy of organizations, that will then be extended and updated during the iterative process to obtain the global organization hierarchy representing the system structure and behaviors. The identified organizations are decomposed into a set of interacting sub-behaviors modeled by roles. The goal of a role is to contribute to the fulfillment of (a part of) the requirements of the organization within which it is defined. In order to design modular and reusable organization models, roles should be specified without making any assumptions on the architecture of the agent that may play them. To meet this objective, the concept of capacity was introduced. A capacity is an abstract description of a know-how, a competence of an agent or a group of agents. The role requires certain skills to define its behavior, which are modeled by capacity. The capacity can then be invoked in one of the tasks that comprise the behavior of the role. In return, an entity that wants to access a role, should provide a concrete realization for each of capacities that the role requires. The capacity also allows, in the modeling process, to make the interface between two adjacent levels of abstraction in the organizational hierarchy of the system. A role at level n requires a capacity that is in turn provides by an organization at level $n-1$. The analysis phase ends with the capacity identification activity that aims at determining if a role requires a capacity and then adapting its behavior description. At this step a new iteration may possibly start. If all capacities required by roles at the lowest level of the hierarchy are considered to be manageable by atomic easy-to-implement entities, the process may stop.

The *Agent Society Design* phase aims at designing a society of agents, whose global behaviour is able to provide an effective solution to the problem described in the previous phase and of satisfying associated requirements. The objective is, now, to provide a model of the agent society involved in the solution in terms of social interactions and dependencies among entities (holons and/or agents). Previously identified elements such as ontology, roles and interactions, are refined. At the end of the design phase, the hierarchical organization structure is

mapped to a holarchy (hierarchy of holons) in charge of its execution. Each of the previously identified organizations is instantiated in forms of groups. Corresponding roles are then associated to holons or agents. In multiagent systems, the vision of holons is much closer to the one that MAS researchers have of Recursive or Composed agents. This last activity also focuses on composed holons and aims at describing their two main aspects : their government and the set of application-dependant organizations they contain. The objective consists in describing the various rules used to take decisions inside each composed holon. Rules governing holons' dynamics in the system are also described to support a dynamic evolution of the system holarchy. All of these elements are finally merged to obtain the complete set of holons (composed or not) involved in the solution. In this way, the complete holarchy of the solution is described.

The *Implementation* phase aims at implementing the agent-oriented solution designed in the previous phase by adapting it to the chosen implementation platform, in our case, Janus. Based on Janus, the implementation phase details activities that allow the description of the solution architecture and the production of associated source code and tests. It also deals with the reuse of previously developed solutions. A set of organizational patterns may have been used during the two previous phases. The code reuse activity aims at integrating the code of these patterns and adapting the source code of previous applications inside the currently designed application.

The *Deployment* phase is the final one and aims at detailing how to deploy an application over various Janus kernels. This phase starts with the description of the deployment configuration and details how the previously developed application will be concretely deployed; this includes studying distribution aspects, holons physical location(s) and their relationships with external devices and resources and tests. This activity also describes how to perform the integration of parts of the application that have been designed and developed with other modelling approaches (i.e. object-oriented ones) with parts designed with ASPECS.

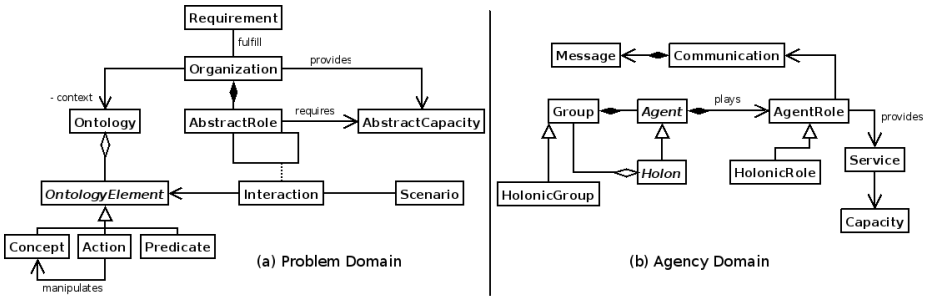

Fig. 3. A part of the ASPECS Problem and Agency domains core metamodel.

4 Building ASPECS

In this section we describe the actual process we adopted for building ASPECS. We report the initially created core metamodel, the definition of the precedence order of the metamodel elements, the selection/assembly of method fragments and the extension of the metamodel with the consequent selection of new fragments in an iterative process. This process is the instantiation of the general process described in section 2 and complements the theoretical part of this paper with the experiment we did in composing ASPECS.

4.1 The core metamodel

A part of the initial core metamodel defined for the ASPECS process can be seen in Figures 3(a) and 3(b). It has been composed by reusing the following elements from the above-described metamodels:

- From PASSI: Requirement, Scenario, Ontology, Ontology Element, Concept, Predicate, Action, Agent, Role (renamed AgentRole in Figure 3(b)), Communication and Message.
- From CRIO: Capacity (renamed Abstract Capacity in Figure 3(b)), Role (renamed AbstractRole), Interaction and Organization
- From Janus: elements from Janus only belong to the implementation model that has not been reported here for the sake of conciseness.

Some interesting issues raised from the composition of these elements in the new metamodel are explained below:

- Elements coming from CRIO have been integrated in the new model with only minor changes in their definitions as can be seen in section 4.3 for Organization and Interaction definitions.
- Two different concepts (Role in PASSI and CRIO) had the same name but different definitions. Essentially, the CRIO Role concept is an analysis level concept while the PASSI Role is mostly a design abstraction. We found a solution to the problem by positioning each of the two elements in the domain it belonged to in its original approach and introducing a transformation relationship between them (i.e. the CRIO AbstractRole is transformed in the PASSI AgentRole when moving from Problem to Agency domain).
- PASSI Requirement is usually related to the agent concept. This represents the fact that in PASSI agents are responsible for satisfying requirements. In the ASPECS process this responsibility is given to the organization as it comes from the CRIO process. Therefore the two concepts have been related and their definitions have been consequently modified.
- The PASSI Agent element as already discussed is no more related to Requirements but it becomes an abstract entity used to give AgentRoles a individuality (shared knowledge and capacity). Holon realizes agents in a concrete way and it is the Solution domain abstraction corresponding to the JHolon implementation class supported by Janus.

- Ontology has the same structure of the PASSI corresponding concept but it is now positioned in the Problem domain. This is the consequence of a precise choice: adopting ontological exploration of the problem domain as a tool for deepening the understanding of the problem to be solved.
- Capacity has been introduced as an agency domain abstraction for representing what the role is capable to do. It is related to Service that is one possible realization of the Capacity. This means that several different services can implement the same Capacity. This structure required a change in the original PASSI specification of Service although maintaining its general meaning.

From these and other similar considerations we built the core metamodel for the ASPECS process. It has not been an easy and short activity but rather it has been performed during several meetings, involving debates with other people not directly belonging to the team of ASPECS developers but involved in previous experiences of usage of agent-oriented methodologies and related platforms.

In the next subsection we discuss the prioritization of the MMMEs that is the order we expect to instantiate these elements in the fragments that will compose the new design process.

4.2 Prioritization of MAS metamodel elements

The priority order of the MMMEs has been defined by applying the already discussed heuristics; the resulting list is: (i) Requirement, (ii) Ontology and all the related elements: Concept, Action, Predicate, (iii) Organization, (iv) AbstractRole and Interaction, (v) Scenario, (vi) RolePlan and RoleTask (not reported in Figure 3(b) because of space concerns) (vii) AbstractCapacity.

This list covers all the elements of the Problem domain and the choice done reflects both the ASPECS design process requirements and some new decisions: Requirement, Ontology and Scenario were the sinks of the graph. Requirement is the first element because of the idea of following a PASSI-like requirements identification phase. Ontology soon follows since we aimed at using the ontological exploration of the domain as a tool for deepening the understanding of the problem. Scenario is positioned later because of its specific meaning: we suppose that text descriptions of user-system interaction stories (sometimes known as scenarios) are provided as an input to our design process. When we talk about scenario here we mean a formalization of this textual descriptions including a detailed list of roles (AbstractRoles) involved in the Interaction(s). Obviously this new and formal description can be done only after the definition of AbstractRole and Interaction. Organization is positioned early in the list since we aim at maintaining the PASSI philosophy of an initial agentification of requirements. AbstractRole and Interaction are positioned soon after Organization. RolePlan and RoleTask will not be discussed here because of space concerns but their positioning in the list is again a consequence of the adopted heuristics. Finally AbstractCapacity is introduced since its inputs are now satisfied. Similarly we

obtained a priority order list for the MMMEs elements of the following domains (Agency and Solution).

After this step we are ready to start with the selection of fragments from the repository or the construction of new ones in order to define the elements according to the prescribed order. This process will be discussed in the next subsection.

4.3 Definition of an initial draft of the process

In performing the fragments selection activity, we refer to our repository of fragments [8] that includes fragments extracted from PASSI, Agile PASSI, TROPOS, and Adelfe. Since several MMMEs required by this novel approach (for instance Holon) are not present in the repository, we expect to produce several new method fragments, hoping of reusing and modifying some existing ones when possible.

According to the previous discussed list of MMMEs, the first method fragment of the process is supposed to draw a model of system requirements by starting from text usage scenarios. This is exactly what the first fragment (Domain Requirements Description) of PASSI does and it was thus reused. The definition of the Ontology is again done in an existing PASSI fragment and it has been reused as well. The next MMME to be defined is Organization. In this fragment we aim at creating a relationship between each organization and the requirements it is responsible for. This is very near to the work done in the PASSI Agent Identification fragment that can therefore be easily adapted to cope with this new situation. The resulting fragment will be labelled Organization Identification. The next fragment is devoted to define instances of AbstracRole and Interaction. These two elements are defined in a static view and then used to depict the dynamics of the system in the following view (where Scenario is designed). The resulting fragments will be Interaction and Role Identification (newly defined fragment adopting a class diagram where elements are instantiated), Scenario description (reused PASSI Role Identification fragment, a series of sequence diagrams describing roles interaction during scenarios). RolePlan and RoleTask are defined in a RolePlan fragment that is obtained by reusing the PASSI Task Specification diagram. Finally Capacity Identification is reused from the CRIO process and it adopts a static view to define capacities and relate them to organizations and roles.

In a similar way we defined the remaining part of the process. In this discussion we omitted the details of each fragment and the difficulties found in defining the new ones as well as modifying the reused ones while adapting them to cope with the new specific issues. This choice draws from the specific scope of this paper that is concerned in demonstrating the role of the MAS metamodel elements as a guide towards the definition of the new design process rather than other aspects of this whole work. In the next section we discuss some examples of extension of the initial core MAS meta-model done in order to refine the initial sketch of the process.

4.4 Completion of the process and extension of the core metamodel

We view the construction of a new design process as an iterative-incremental activity that can be decomposed in the following steps: (i) Construction of a process stub (including several fragments, for instance up to reach the phase size). (ii) Test of the process portion. (iii) Evaluation of results. (iv) Next iteration planning in terms of new process requirements to be addressed, changes to be done in the existing process stub, and new parts of the metamodel to be included in the process.

In the case of the ASPECS methodology, we performed the first significant test activity after completing the System Requirements phase. This test consisted in using the new process stub for designing a couple of simple applications. This allowed us to familiarize with the process and to appreciate its qualities. We only proposed one minor change: the explicit introduction of non functional requirements in the early stages of the process (this implied an extension of the metamodel). After that, according to the 4-steps process discussed at the beginning of this subsection, we designed a new portion of the metamodel, more specifically, the core part of the Agency domain metamodel (see Figure 3(b)). We are not going to detail the work done on this part of the process, we will only discuss one interesting point: the extension of the initially defined core metamodel represented in Figure 3(b) to cope with some new process requirements identified during the iteration. After completing this portion of metamodel, the corresponding process stub and included method fragments, we started performing some tests and during them we realized that the new process had some limits: it was not possible to represent not FIPA-compliant agent interactions (for instance environment mediated). They had not been initially listed among the new process requirements but they were already supported by the Janus platform and sometimes used in previous projects developed in the lab. Another issue arise from the consideration that it was not possible to design simple (non holonic) agents like the conventional PASSI ones. This has been seen as a limit since it limits the possibility of integrating in the same design complex holonic hierarchies with simple agents devoted to deal with minor parts of the problem. In order to solve these issues we changed and extended the core metamodel by including a Conversation and an AtomicAgent MMMEs.

The extended metamodel has been fully realized by a set of fragments and then the process stub tested and evaluated as already described. The work continued in an iterative way until the complete process was defined and thoroughly tested[5].

5 Conclusion

Based on the Situational Method Engineering, this paper has reported an experiment of creation of a new process called ASPECS. The proposed approach

[5] A complete description of the ASPECS process can be found at: http://set.utbm.fr/index.php?pge=352&lang=fr

starts from the identification of the new process requirements in terms of development context, problem type. The requirements are used for defining an initial core version of the MAS metamodel. The elements of this metamodel are then ordered in a precedence list and in this order the fragments that are able to deal with, are retrieved from the repository and assembled in the new process. The resulting MAS metamodel of ASPECS [5] is mainly composed by elements coming from the PASSI [6] and CRIO [7] existing design processes and supports Janus as an implementation platforms of holonic agents. In previous works applying SME, the method engineer usually selects a set of method fragments that he considers as the best for fitting a particular situation and then modify or adapt them the most reusable part according to his own experience. In contrary to these approaches, the approach described in this paper aims at being as much free as possible from the designer skills providing a set of reusable guidelines for fragments selection and assembly.

References

1. Harmsen, A., Ernst, M., Twente, U.: Situational Method Engineering. Moret Ernst & Young Management Consultants (1997)
2. Brinkkemper, S., Welke, R., Lyytinen, K.: Method Engineering: Principles of Method Construction and Tool Support. Springer (1996)
3. Ralyté, J.: Towards situational methods for information systems development: engineering reusable method chunks. Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education (2004) 271–282
4. Henderson-Sellers, B.: Method engineering: Theory and practice. In: ISTA. (2006) 13–23
5. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: A holonic metamodel for agent-oriented analysis and design. In: 3rd Int. Conf. on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS'07. Number 4659 in LNAI, Regensburg, Germany (September 2007) 237–246
6. Cossentino, M.: From requirements to code with the PASSI methodology. In: Agent Oriented Methodologies. Idea Group Publishing, Hershey, USA (2005) 79–106
7. Hilaire, V., Koukam, A., Gruer, P., Müller, J.: Formal specification and prototyping of multi-agent systems. In: Engineering Societies in the Agents' World. Number 1972 in LNAI (2000)
8. Seidita, V., Cossentino, M., Gaglio, S.: A repository of fragments for agent systems design. Proc. Of the Workshop on Objects and Agents (WOA06) (2006)
9. Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. International Journal of Agent-Oriented Software Engineering (IJAOSE) **1**(1) (2007) 91–121
10. Mirbel, I., Ralyté, J.: Situational method engineering: combining assembly-based and roadmap-driven approaches. Requirements Engineering **11**(1) (2006) 58–78
11. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-modelling based assembly techniques for situational method engineering. Information Systems **24** (1999)