Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies

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

[1] Dipartimento di Ingegneria Informatica, University of Palermo,
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. Situational Method Engineering for constructing ad-hoc agent oriented design process is grounded on a well defined set of phases that are principally based on reuse of components coming from existing agent design processes that have to be stored in a repository. The identification and extraction of these components could take large advantage from the existence of a standardized representation of the design processes they come from. In this paper we illustrate our solution based on SPEM 2.0 specifications for modelling agent design processes and extending them when necessary to meet specific needs we faced in our experiments.

1 Introduction

Our research is focussed on the field of Situational Method Engineering (SME) [7][1][6][8] for the construction of ad-hoc multi agent systems design processes. Applying Situational Method Engineering requires executing a well defined set of phases [5][4][11]: Process Requirements Specification, Process Fragments Selection and Process Fragments Assembly.
SME is based on the reuse of components coming from existing design processes, the so called method fragments or simply fragments; the request for reusable fragments leads to the need for a repository containing standardized fragments that could be easily selected and assembled in new design processes.
Since the repository is composed of fragments coming from existing design processes, its construction cannot be done without considering: the knowledge of a set of existing processes, their standard description through a standard notation and a precise definition of the fragment notion and of the process itself.

We decided to use SPEM (Software Process Engineering Metamodel) 2.0 [9], both for design process and fragments representation, because it is an OMG standard and because it is based on a metamodel (we already adopt metamodels in our work) containing the main elements a design process is composed of: activity, work product and process role [3].
Besides in the agent oriented context, according to our view, a process is devoted to design a MAS model whose elements are represented in the work products

resulting from the enactment of a specific activity. A MAS model element is an instance of a MAS metamodel element; the MAS metamodel represents the structure of the system that is being built with the specific design process.
The MAS metamodel is one of the most important factors of our approach, since it is not present in the SPEM specifications we decided to extend these specification and in this paper we present how we use SPEM, its elements and diagrams, for representing a design process and how we extended it by adding elements and diagrams, in order to meet our needs.

The paper is structured as follows: in section 2 an overview on the Situational Method Engineering approach for creating new agent oriented design processes is given; in section 3 the main SPEM elements we use are illustrated, the needed extensions are justified and an example on applying SPEM is provided; finally in section 4 some conclusions are drawn.

2 The Formal Description of a Design Process

The construction of a new design process following Situational Method Engineering principles is based on three main phases [4]: Method Requirements Engineering, Method Design and Method Construction. Our approach for applying SME in the agent oriented context is based on these three phases too but it is specialized for the agent context, as shown in Figure 1; here we sketch the whole process a method designer has to carry out in order to construct a new agent oriented design process (i.e a methodology). The first step a method designer must undertake is to create a repository of fragments starting from those extracted from a set of existing design processes and/or constructing new ones from scratch. During the Process Requirements Analysis activity, the method Designer considers inputs coming from the development context (tools, languages, available skills, etc.) and the type of problem he has to solve. These inputs are used to define the process life cycle (that establishes the structure the designer has to follow during process fragments assembly activity), the system metamodel concepts and the other process elements (available stakeholders, required activities or work products) that are used for selecting fragments from the repository and for constructing the MAS metamodel (MMM) of the system to be. It is worth to note the fundamental difference between the MAS metamodel, that will be instantiated in the actual agent system during design, and the process (or method fragment) metamodel, that will be used to define the design process.

The output of the process requirements analysis contributes to the selection of process fragments and to their assembly; once the process is created, it can be enacted and then evaluated for eventually iterating all of its construction process.
In this work we want to point out our attention to the method base construction, for which we need two elements: a set of existing design processes used to extract fragments to be stored and a specific definition of design process and therefore of method fragment that enables the method engineer to correctly describe the existing design process in order to extract from them a set of fragments.

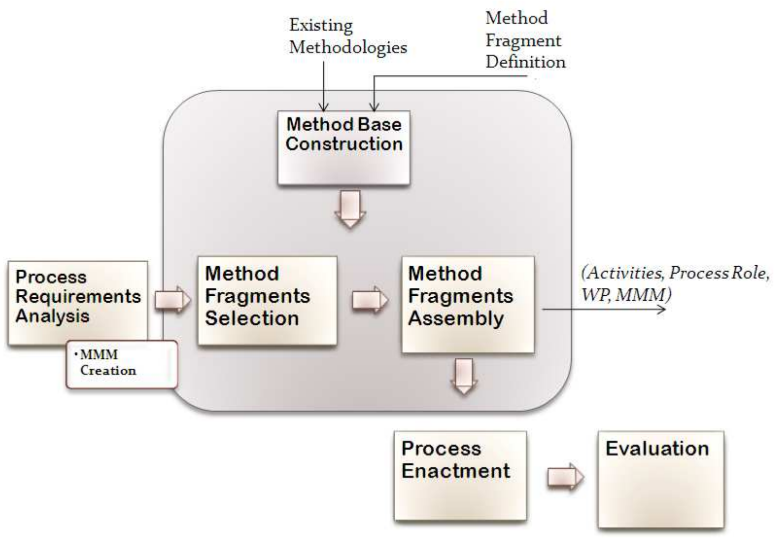
Existing Methodologies
Method Fragment Definition
Method Base Construction
Process Requirements Analysis
•MMM Creation
Method Fragments Selection
Method Fragments Assembly
(Activities, Process Role, WP, MMM)
Process Enactment
Evaluation

Fig. 1. Our Approach for Creating an Agent Design Process

As regards the definition of design process and process fragment we consider a design process as the set of activities to be performed in order to produce an output, the way of performing some activities (guidelines or techniques), and the resources and constraints this requires. In [3] we gave a definition of multi-agent system design process; in Figure 2 we represent the main elements we use for describing the agent design processes for their fragmentation.

A design process is composed of activities, each activity is performed by a process role that is responsible for one or more work products that are structured by a work product kind representing a specific category, for instance, text document, code and so on. A design process is devoted to design a MAS Model that is composed of MAS model elements each of which can be represented in one or more work products; a MAS model element is an instance of a MAS metamodel element so in each work product there is a correspondence with one or more MAS metamodel elements. A process can be decomposed into (process) fragments that are self-contained pieces of the whole process, with all the elements characterizing the process itself (activity, process role, work product and MAS metamodel element) and that instantiate one or more MAS metamodel elements described in the work product(s) resulting for the fragment itself. In our approach we assume that each fragment has to deliver at least one work product, thus, basing on this hypothesis and on the proposed fragment definition, we can say that our fragment extraction activity is work product driven.

We decided to follow a top-down approach for a clear description and an easy retrieval of the main elements of a process: starting from highest level activities,

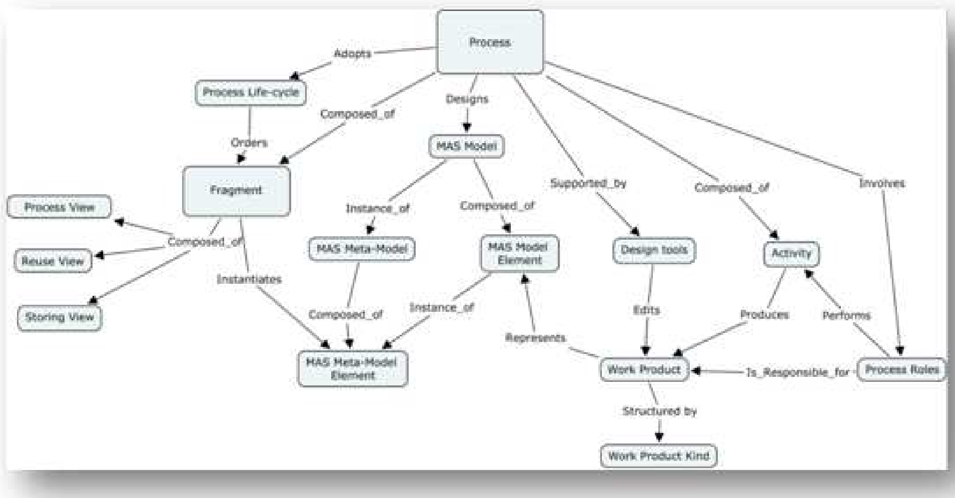

Fig. 2. The Agent Oriented Design Process Definition

we decomposed them (and the corresponding fragment) down to the atomic steps that compose the work to be done; at each level of detail, we report the produced work products (with their work product kind) and the specific process role that perform/assist the work and above all the description of the MMM element that is defined/refined/quoted in each work product.

Because of our formalization needs we decided to adopt SPEM 2.0 (Software Process Engineering Metamodel) Specification; in the next section we will describe which elements and diagrams we use of this specification and for each of them we will illustrate the related definition (from [9]).

3 Using SPEM for Representing Agent Oriented Design Process

Software Process Engineering Metamodel (SPEM) [9] is a meta-modelling language used for the description of development design processes and their components.
The SPEM is based on the idea that *"a software development process is a collaboration between abstract entity called process role that performs operations called activities on tangible entities called work products"* [9]. This well defined conceptual model allows to represent every kind of process lifecycle (iterative, incremental, waterfall and so on), SPEM in fact is composed of a breakdown structure allowing to represent all design processes.

The SPEM 2.0 presents a metamodel structured in seven main packages; only three of them will be illustrated in this paper in order to justify their specific use, they are: *Process Structure*, *Method Content* and *Process With Method* Packages. Each of them contains all metamodel classes and abstractions for describing and modelling a view of a design process.
It is worth to note that in SPEM 2.0 the concepts of "*Method*" and "*Process*" have a specific meaning, later detailed, that allows their use respectively for representing and modelling the fragment and the design process.
The *Method Content* package contains all the elements for creating a set of reusable methods, independently from a specific project, in which techniques and concrete realizations of best practices are specified; the aim of this package is to illustrate which are the goals that a method has to reach, which resources are used and which roles are involved.
The *Process Package* is composed of the main elements for modelling a process: Activities, nested in a breakdown structure where the performing Role classes and the input and output Work Product classes for each activity are listed. These elements are used to represent a high-level process that when instantiated on a specific project takes the method content elements and relates them into partially-ordered sequences that are customized to specific types of projects. Finally, the *Process With Method* Package contains all the elements for integrating the method content in the process.

To sum up a Process provides a complete breakdown structure for a specific development situation, whereas a Process with Methods relates each method content elements (such as Tasks Definition, Work Products Definition, etc.) to an ordered sequence of work steps customized for specific types of projects.
Therefore at a first level of abstraction a process, through the Process Package elements, can be represented in its general structure without any reference to a specific project and without detailing the inner content description of each activity.

SPEM 2.0 presents two levels for process representation, both of them aim at modelling the portion of work to be done with the required input/output and involved roles, but whereas a Method is considered at an higher abstraction level, where there is no reference to a specific project (and above all it is considered as an auto consistent portion of process), a Process is the concrete representation of a specific development situation.

For these reasons SPEM 2.0 is suitable both for our top-down decomposition/representation of a design process and for the representation of each fragment; in fact we can use the *Method Content* Package elements to represent a (process) fragment and the *Process With Method* Package elements to represent an existing design process; in fact a development process, in our approach, is composed of process fragments (see section 2). From these two packages definitions it is clear that SPEM 2.0 well supports our representation of a process as composed of fragments.

In the *Process with Method* Package the central element is the BreakdownElement; in SPEM 2.0 processes are represented with a breakdown structure of Ac-

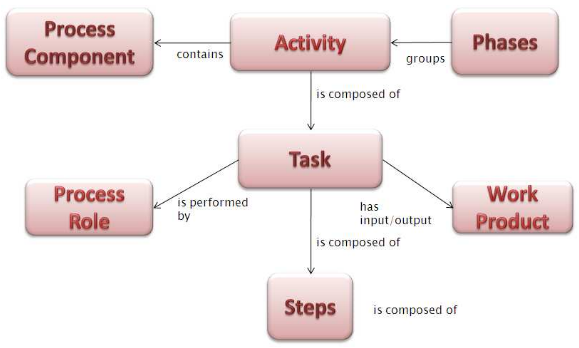

Fig. 3. The Used SPEM Elements

tivities that nest BreakDown Elements.
The BreakdownElement definition from Process Package, that is not shown here [9], is "*the abstract generalization for any type of Process Element that is part of a breakdown structure. Any of its concrete subclasses can be placed inside an Activity that represents a grouping of nested BreakDownElements such as other Activity instances, Task Uses and Role Uses*". Thus Activity represents a basic unit of work within a Process as well as a Process itself.
Role Uses represent a "*Role in the context of a specific Activity, it is the performer of a Task Uses and defines a set of related skills, competencies and responsibilities of a set of individuals*" and *Task Uses* define the unit of work that is performed by Roles; a Task Use has a clear purpose in which the performing roles achieve a well defined goal. Task Uses provide complete step-by-step explanations for doing all the work that needs to be done to achieve a goal. *Work Product Uses* are the artifacts, produced, consumed or modified by Task Uses[3]; Roles use Work Products to perform Tasks and produce Work Products in the course of performing Tasks. As it could be noted the *Process With Method* Package contains the same elements we considered in our process metamodel (see section 2), the Activity can be related/mapped to Task Use, The Process Role to Role Use and obviously the Work Product to Work Product Use.
These elements together with the rationale of the work breakdown structure, upon which SPEM is based, are sufficient enough for describing a complete design process for our purposes following a top-down approach, from the higher level definition of the work to be done until the details of each task with the roles performing it and the artifacts produced.
Besides SPEM provides other two useful elements, not explicitly included in our process metamodel, for grouping a set of activities under a common application theme, they are the *Process Component*: that is a "*special Process Package that applies the principles of encapsulation. A Process Component contains exactly*

[3] From now on, for the sake of brevity, Task and Task Use, Role and Role Use, Work Product and Work Product Use will be indifferently used, with the same meaning.

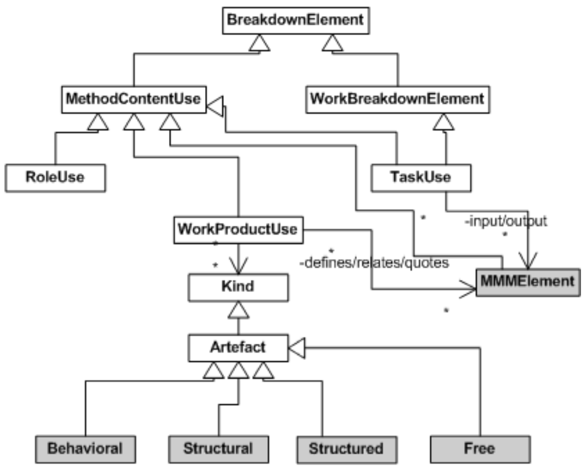

Fig. 4. The Proposed Extension to Process With Method Package Metamodel

one Process represented by an Activity and defines a set of Work Product Ports that define the inputs and outputs for a Process Component", and the *Phase* that represents "*a significant period in a project, ending with major management checkpoint, milestone or set of Deliverables. It is included in the model as a predefined special Activity, because of its significance in defining breakdowns.*"

Figure 3 resumes which SPEM elements and related relationships we use for representing a design process, an example will be provided in the following section. A whole process can be divided into process components that groups the activities under a common theme, set of activities are also grouped into phases that impose specific milestones to the work to be done. Techniques, methods and guidelines for each activity are given for tasks that are performed by roles and consumes/produces (has input/output) work products.

3.1 Extending SPEM specifications

In the previous section we saw which elements of SPEM metamodel packages we use for our purposes of providing a standard representation for agent oriented design methodologies, they are summarized in Figure 3; however we needed some extensions on SPEM Packages elements due to the presence of two specific elements in our process metamodel that are not present in SPEM Packages: the MAS Metamodel element and the Work Product Kind.
Regarding the first element, its definition is the core of the work done in each portion of process resulting in the delivery of a work product; as section 2 illustrates, using a specific design process for developing an agent system means to instantiate its metamodel, this instantiation results in a precise set of actions that we identified to be done on each MMM element; a designer can in fact *i)* **define** an element (i.e. instantiate it) thus establishing its properties and/or attributes; the resulting instantiated element is, of course, reported in the work

product produced by the fragment the design is executing. Similarly, the designer can *ii)* **relate** a MMM element to other elements or *iii)* simply **quote** it for introducing relationships among work products.

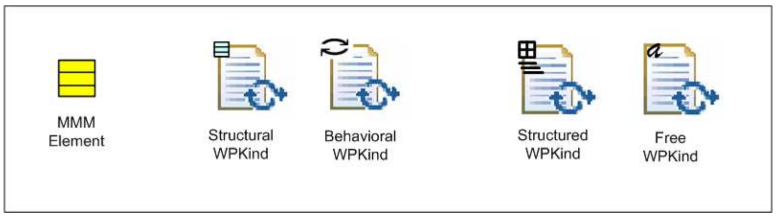

Fig. 5. The Proposed Icons

An important factor in the process representation is also modelling work products dependencies; SPEM specifications provide a way for modelling dependencies through the so called Work Product Dependency Diagram.
The Work Product Dependency diagram supplies way for representing three kinds of dependency relationships among work products: Composition, Aggregation and Dependency, the first expresses that a work product is a part of another one, the second indicates that a work product uses another work product and finally the third indicates that a work product depends on another one. In our case the dependency among a work product and another is due to the relationships among elements in the MAS metamodel; for instance let us consider an hypothetical requirement elicitation phase of a given process, we could think that this portion of process underpins a metamodel where the concept of actor is related to that of scenario; now, let us further suppose that the designer performing this phase begins his work by defining the concept of actor and produces a document listing all the actors he has identified.
When the designer has to define the concept of scenario, since this is related to the concept of actor, he has to look at the last document he created, he cannot proceed without knowing all the defined actors; in so doing all the relationships among metamodel elements are reflected upon a precise dependency among the work products that act upon them. In order to represent such situations, we need a diagram reporting both the dependencies among work products (and this is provided by SPEM) and the correspondence between each work product and the MMM elements.
The SPEM Work Product Kind allows to represent a work product when it: *i)* is an intangible one or it is not formally defined (in this case it is of the kind: Outcome), *ii)* it aggregates other work products (the kind is Deliverable) and *iii)* it defines a tangible work product consumed, produced or modified by a task (the kind is Artifact).
In our work we defined four kind of work products [10] as a result of our need for adequately storing process fragments in our method base; the defined kinds on work products are: Behavioural, Structural, Structured and Free, they can be

considered a specialization of the SPEM Artifact Work Product Kind (Figure 4).

Therefore for applying our design process definition we made some specific extensions to the Process With Method Package by adding some elements; in Figure 4 we show the portion of Process With Method metamodel that we extended; the white elements are all the pre-existing SPEM ones whereas the gray elements are new.
To sum up, we added five elements and for each of them the related icon was created (see Figure 5); the relationship between Work Product Use and MMMElement found its realization in a new diagram that represents the correspondence between each work product and the MMM elements it defines/relates/quotes. Figure 6 shows the artefact produced during the PASSI Agent Society phase, it is composed of work products of three kinds (structural, behavioural, structured) and for each of them all the MMM elements they work on are shown, the letter indicates the specific action that is made on the MMM element: **D** stands for Define, **R** for Relate and **Q** for Quote. As it will be seen in the next subsection Agent Society is one of the Process Component elements of the PASSI process, it also is the phase resulting in the Agent Society model, modelled through a Work Product Use composed by two structural work products, two behavioral ones and structured.

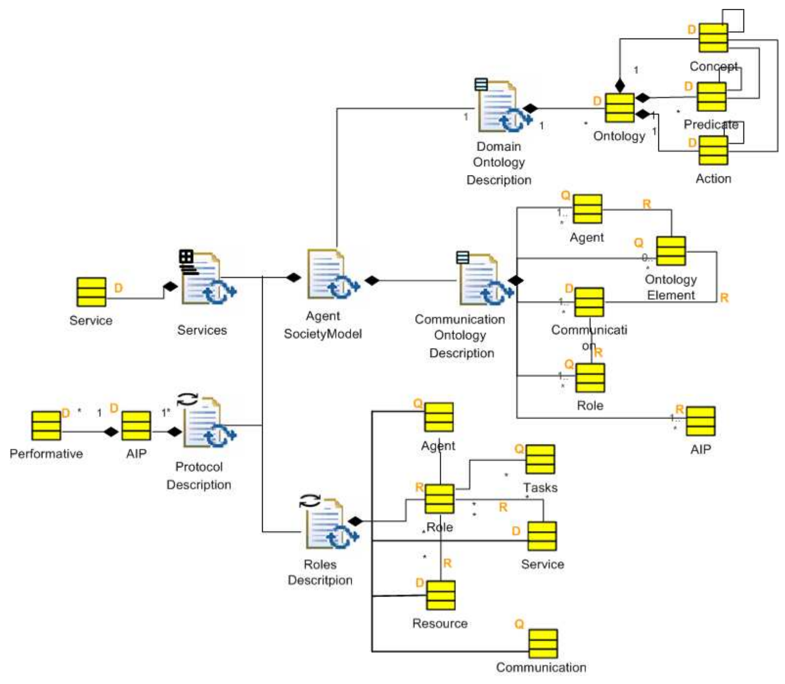

Fig. 6. The Correspondence among Work Product and MMM elements

3.2 Representing PASSI with SPEM 2.0

The standard representation of a design process follows a well specific method, it is carried out in a top-down fashion in order to reach the right level of granularity allowing the extraction of process fragments. Thus a process is represented in a first time as a package of components through the *Process Component Diagram* that allows to represent all the portions of a design process with the needed input and output, this diagram models the design process at a high level of detail. In Figure 7 it can be seen, for instance, the Process Component Diagram related to PASSI [2], it is composed of five components, each of them representing a phase, a portion of work for which a specific outcome and milestones can be identified and represented in this diagram.
The second step is to detail the work done in each component through an Ac-

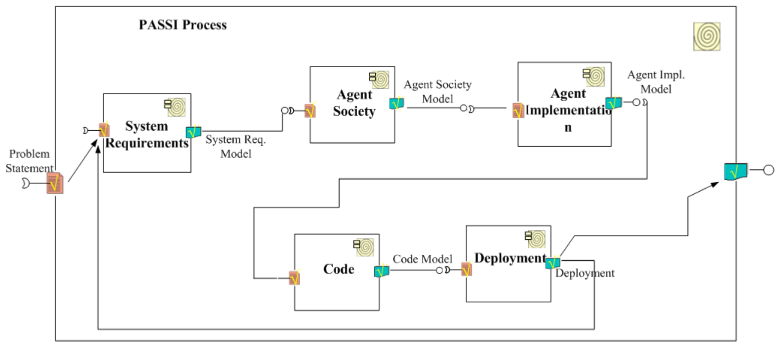

Fig. 7. The PASSI Process Component Diagram

tivity Diagram, Figure 8 shows all the activities nested in the Agent Society component, the activities sequence is indicated through the <<predecessor>> stereotype (the pointed activity is the predecessor of the other one); each activity is composed of tasks, roles performing the task and input/output work products, no tasks sequencing is necessary at this step, it is only useful to give the idea of the set of elements an activity is composed of (in the figure only the Role Description activity is shown for space reasons). When complete, this diagram is full of information that could be accompanied by definition, explanation and so on, but it can result too huge, in alternative another diagram can be produced where only activity and input/output work products are shown.

This level of detail is sufficient in order to identify all the process fragments that can be extracted from a design process; for a detailed documentation of the whole design process other diagrams have to be produced, they allow to model and document all the techniques, the methods and the guidelines involved in each task, i.e. the dynamic part of each portion of process, for space reasons we do not specify this part in this paper. As already said, our process fragment extraction is work product-oriented [3], in the sense that we look at a work

product and extract from the whole process only the portion of work delivering the selected work product.
For instance looking at the Role Description we can see that there are two outputs, Services and Role Descritpion diagram, from Figure 6 (thai is the final digram we draw when we model a design process) we can identify the MMM elements these work products act on and so we could decide to extract a fragment dealing with the concept of role and delivering the Role Description diagram and/or one dealing with the concept of service; of course it is also possible to identify a fragment delivering both the work product.

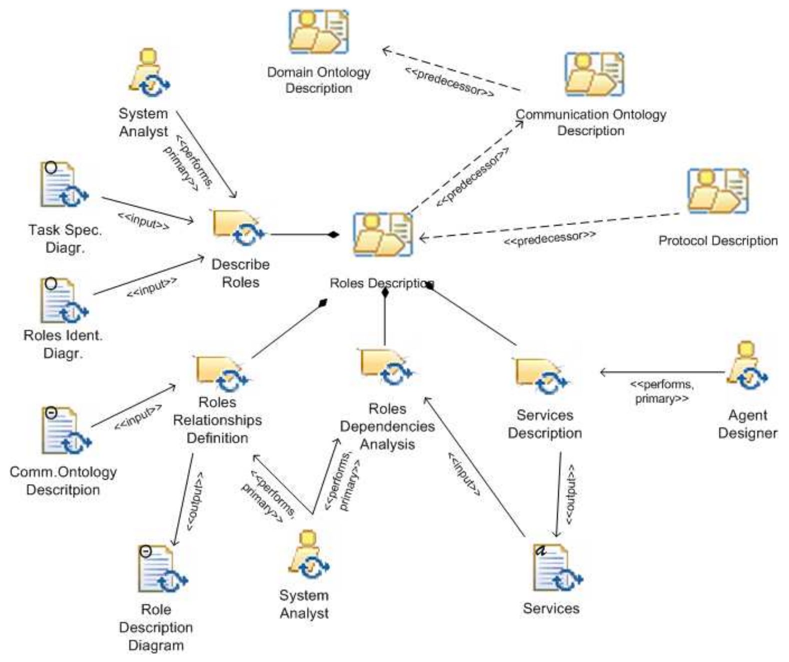

Fig. 8. The PASSI Process Activity Diagram

4 Conclusions

The approach we use for adapting Situational Method Engineering to agent oriented design processes construction is composed of three phases; in this paper we mainly consider the part regarding the (process) fragments documentation in a form that encourages the fragment extraction from an existing process and the repository construction. We started this work from a set of existing design processes that we represented and fragmented in a standard fashion in order to extract a set of fragments coherent with the definition we gave.

The first problem we faced was to find a way for representing a design process coherently with the process metamodel we consider in our approach (activity, process role, work product and MAS metamodel element); we decided to follow a top-down approach for the design process representation using and extending, when necessary, the SPEM 2.0 specifications.
SPEM is well suited to our purposes because of the conceptual metamodel it underpins; the breakdown structure, SPEM is composed of, let us to represent PASSI only using three kinds of diagrams; one of them is the extension we made on the work product dependency diagram provided by SPEM.
In the past we used SPEM 1.1 in our work and we modelled several agent design processes, we found several difficulties that are now over with SPEM 2.0; in the future we are going to experiment and to test SPEM 2.0 for other processes and to formalize a set of guidelines for correctly model an agent design process for the scopes of Situational Method Engineering.

References

1. S. Brinkkemper. Method engineering: engineering the information systems development methods and tools. *Information and Software Technology*, 37(11), 1995.
2. M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies*, chapter IV, pages 79–106. Idea Group Publishing, Hershey, PA, USA, June 2005.
3. 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.
4. D. Gupta and N. Prakash. Engineering Methods from Method Requirements Specifications. *Requirements Engineering*, 6(3):135–160, 2001.
5. A.F. Harmsen, M. Ernst, and U. Twente. *Situational Method Engineering.* Moret Ernst & Young Management Consultants, 1997.
6. Brian Henderson-Sellers. Method engineering: Theory and practice. In *ISTA*, pages 13–23, 2006.
7. 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.
8. I. Mirbel and J. Ralyté. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11(1):58–78, 2006.
9. Software process engineering metamodel. Version 2.0. Final Adopted Specification ptc/07 03-03.
10. 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.
11. V. Seidita, J. Ralyté, B. Henderson-Sellers, M. Cossentino, and N. Arni-Bloch. A comparison of deontic matrices, maps and activity diagrams for the construction of situational methods. In *CAiSE'07 Forum, Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering.*, pages 85–88, Trondheim, Norway, 11-15 June 2007.