How to Extract Fragments from Agent Oriented Design Processes

Valeria Seidita[1], Massimo Cossentino[2], and Antonio Chella[1]

[1] Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica,
University of Palermo, Italy
{valeria.seidita, antonio.chella}@unipa.it
[2] Istituto di Reti e Calcolo ad Alte Prestazioni,
Consiglio Nazionale delle Ricerche, Palermo, Italy
cossentino@pa.icar.cnr.it

Abstract. Using Method Engineering for creating agent oriented design processes is a challenging task because of the lack of a fragment repository defined and filled starting from a shared and unique definition of fragment. The creation of a repository implies the fragmentation of existing agent design processes. In this paper we propose a set of guidelines for extracting fragments from agent design processes. The work is based on a precise definition of fragment and it aims to establish a method for fragmenting processes and obtaining homogeneous fragments regardless of how the starting design processes are defined and described.

1 Introduction

When method designer is about to create a new agent oriented design process using a (Situational) Method Engineering [4] compliant approach (s)he needs a repository of fragments; (s)he needs to inspect and query the repository in order to quickly and easily select the fragments that best fit her/his needs and that (s)he may assemble in order to compose/create the new agent design process. If the fragments are correctly identified the result of the assembly phase would be the design process that best allows to develop the multi-agent system solving a specific class of problems.

Currently there are several very good approaches to Situational Method Engineering (SME) [19][6][20][13], all of them primarily aim to re-use components that come from other existing design processes, hence they provide solutions already used and tested. Nevertheless each approach follows its own logic, based on the specific used definition of fragment, and as a consequence the related repository has very specific features that make it little reusable. A repository is created starting from the fragmentation of existing design processes and currently all the existing ones are created in a naive fashion, often based on the skills and experience of the method designer. There is not a well-specified technique to split existing agent design processes and to extract fragments from them.

The fragmentation process basically prescribes to identify some points in which "to cut" the design process. The problems we are facing are: how to find

these points? How to understand and verify that the identified pieces have a meaning, are consistent, from a methodological point of view? It is to be remembered in fact that the fragment of a process is a process itself, therefore it should be a component with a meaning in the process perspective; in our case in an agent oriented process perspective. All this has not found adequate answers so far because if the method engineer wants to extract fragments (s)he must have a clear idea of what is a process fragment for an agent oriented process and (s)he has to highlight the main elements of the fragment that must be sought in the design process. We can see the fragment as a piece of a puzzle, if we do not know how the piece of a puzzle is done we cannot find it in the puzzle, we would always see the puzzle as a whole.

A first and fairly established definition of fragment already exist [9]; on the basis of this definition several research groups broke down their own agent design process into pieces and extracted several fragments that today form a first prototype of agent fragment repository[3]. Despite that, each fragmentation process is not replicable and repeatable on another agent design process and above all it can be done only by (or with the tight assistance of) the design process experts. Moreover, the fragmentation obtained without the aid of specific guidelines has led to fragments that are different in granularity and in the representation. They are not homogeneous even if they are based on the same definition; this fact makes their assembly very difficult.

What we propose in this work is a mean for answering to the previous reported questions. The first question is met by considering the improved definition of fragment we introduce in this paper; above all the granularity of the fragment and the work product lead to the identification of what we call the "*cutting points*". The granularity and the work product are strictly related to our definition of fragment. The answer to the second question is guaranteed by the use of the multi-agent system metamodel during the extraction process; the multi-agent system metamodel is the core of the fragment definition, it is one of the most important concepts in agent oriented software engineering, it strongly distinguishes our approach from the others by providing the key point for applying our SME approach.

In the remainder of this paper, we first illustrate the background on (situational) method engineering and the motivations of our work (section 2). In section 3 we define the concept of process fragment that is the basis of our approach on SME and in section 4 we introduce the guidelines for fragmentation. Finally in sections 5 and 6 some discussions and conclusions are drawn.

2 Background and Motivation

Method Engineering is the "engineering discipline to design, construct and adapt methods, techniques and tools for the development of systems", this is the def-

[3] In http://www.pa.icar.cnr.it/passi/ fragments from PASSI [7], Tropos [3] and Adelfe [2] can be found. Fragments from INGENIAS [21] can be found in http://grasia.fdi.ucm.es/main/fr/node/241.

inition generally accepted since [4]. Method Engineering (ME) and Situational Method Engineering (SME), which is the part of ME dealing with the creation of method(ologies) for specific situations, is the answer to the historical problem of the lack of a one-size-fits-all methodology [19][29][28].

Several approaches to situational method engineering are present in literature, they all descend from the assumptions made by Brinkkemper in [4] and then by Gupta and Prakash in [12] that a method(ology) engineering process is composed of three main phases: method requirements engineering, method design and method construction. More in details Brinkkemper highlights the following steps: characterization of the project, selection of the method fragments, assembly of the method fragments. The method fragment is a piece of existing method(ology), an optimized method(ology) for specific situation may be constructed by reusing relevant method fragments.

Literature emphasized two different modus operandi for applying situational method engineering; one is called assembly based method engineering, the other may be called method engineering by configuration. The assembly based method engineering mainly focuses on assembling method fragments. Recently Ralyté et al. [23] developed a generic process model for situational method engineering. This is an assembly based process model implying the specification of method requirements, the selection of method fragments and finally the assembly of method fragments. Method engineering by configuration prescribes to adapt one particular method to different situations [18].

In any of the previous cases situational method engineering does not principally focus on obtaining method fragments but all the approaches assume to have an already filled repository from which to select method fragments.

We think that we cannot be exempt from considering the criticality of this part of SME. How might we create a repository of fragments to be used for creating specific agent oriented design processes? Therefore the lack of an agent oriented fragment repository, structured in a way that makes easy reusing fragments and fosters the interoperability of the existing approaches, severely limits and affects the use of the SME. It is in fact still considered too difficult to apply.

Having guidelines for fragmentation is very important because one way to break up can lead to homogeneous fragments in which the interfaces are easily identified; this can make the selection of the fragments and their assembly faster and easier. This problem is difficult to tackle because there are many agent oriented design processes from which we can extract fragments but each of them presents features strongly tied to the type of multiagent system they are devoted to develop. Often there is no proper or standard documentation, sometimes the elements that are highlighted in one design process are not dealt with in another one or have different definitions. The substantial difference between design processes, which reflects the intrinsic difference of the different types of agents treated, makes fragmentation very difficult; especially if this process needs to be done by a person who is not fully aware of every detail of the design process.

The fragmentation method we propose overcomes this limitation because of the use we make of an important element of agent design processes, the multi

agent system metamodel. In our approach the system metamodel is a fundamental part of the fragment definition and it is also the core for fragments selection, for assembling them [27] and, as it will be illustrated later, for extracting fragments from existing design processes.

3 The Adopted Fragment Definition

The most important concept in the Situational Method Engineering approach is the *method fragment*, thus it is mandatory to have a repository of fragments and techniques for selecting and assembling them. Moreover a relevant part of the method designers' work concerning the population of fragments repository notably influences the whole SME process. From the experiences made in constructing agent design processes we learnt that the techniques for selecting and assembling fragments, for extracting them from existing design processes and then for storing in the repository strongly depend on the definition of fragment the method designer uses during his work.

Although the focus of this paper is on the fragmentation guidelines, it is important to introduce the definition of fragment that forms the basis of our approach to SME. We already defined the concept of fragment [9] (from now on it will be referred to as *process fragment* for reasons that will be clear later in the section), and now we give an overview on the process fragment definition and the improvement introduced to the first version. The notions here introduced are necessary for understanding the guidelines, how they were created and why some steps are present.

We agree on the general statement that the process fragment is a reusable portion of a design process. Let us dwell on the terms "reusable" and "design process": reusable, we want to create a mean for the method engineer to quickly and easily select and then reuse process fragments for creating new agent design processes. Design process, Fuggetta in [11] defined the software development design process as *"the coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy and maintain (evolve) a software product"*.

We adopt this definition of design process. In plain terms, we could say that design process includes components devoted to make explicit what is the work to be done for pursuing an objective, hence delivering some kind of artefact, who has to do this work and how. The process fragment may be seen as a complete portion of a design process constructed in such a way it can be profitably (and rather easily) reused in a new design process creation. The name we chose to adopt (process fragment) is a direct consequence of that. Most commonly in literature the reader can find names like method fragment or chunk. As it will be evident later on, the proposed definition of process fragment is near to the chunk concept [1][24][20], while the method fragment definition [19][5][15][16] addresses a smaller item, pragmatically speaking a process fragment can be built by assembling several method fragments. Up to now, it can be said that design

process, and process fragment, ground on three principal components: activities, artefacts or work products and stakeholders or roles.

The principal difference of our approach against the others is that we add to this triad another very important concept that is crucial in the agent oriented software engineering, the *multi-agent system metamodel*. The system metamodel is the representation of constructs needed for creating system models.

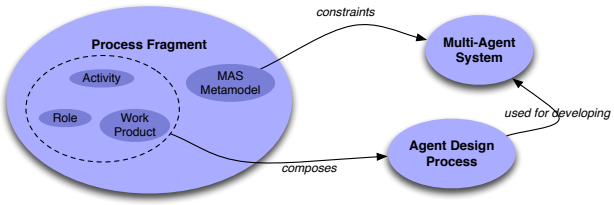

Fig. 1. An Overview on Process Fragment

Figure 1 shows an overview of the process fragment concept and its use in our approach. The process fragments compose an agent oriented design process and are mainly composed of activities, work products, roles, and multi-agent system (MAS) metamodel. The MAS metamodel contains constructs to be instantiated during the design activities and constraints the multi-agent system. In the following a detailed list of all the elements of our process fragment definition is reported:

- specification of the workflow, the work to be done (activities, tasks or steps[4]), the roles performing it and the work products produced;
- a list of constructs of the multi-agent system metamodel to de defined (or refined) through the fragment workflow;
- a description of the fragment goal, for providing the reader with a quick understanding of the design objective pursued by the process fragment workflow; for instance "the aim of this fragment is to identify the agents involved in the system";
- a description of the fragment origin and its granularity. This part lets the method designer have a quick idea on the focus and the domain in which the fragment might work, this can also allow a sort of automatic or semiautomatic selection of the fragments;
- a set of guidelines that are divided in enactment guidelines and reuse guidelines, the first helps the designer while he is using the fragment for producing a portion of the multi-agent system, the second is used by the method designer while creating a new design process for having means for carrying on the selection and the assembly phases;

[4] See SPEM 2.0 specification [22] for a definition of these items.

– a glossary of terms used in the fragment; this prevents misunderstandings if the fragment is reused in a context that is different from the original one.

As regard the granularity, our SME approach does not require to establish a *length* for the portion of process in the process fragment; it is only needed that it can manage some specific elements. Therefore we can have process fragments at three different levels of granularity: phase fragment, composed fragment and atomic fragment as defined below.

Phase Process fragment. A phase process fragment delivers a set of work products belonging to the same design abstraction level of the design flow. An examples of phase-level work product may be a system analysis document; it is composed of several work products (diagrams, text documents, ...) all belonging to the same design abstraction level (system analysis).

Composed Process Fragment. A composed (process) fragment delivers a work product (or a set of instances of the same work product). Composed process fragments may be nested. This is an obvious choice to allow the maximum flexibility for representing whatever size of fragment. For instance a composed fragment delivering a composite work product may be composed by composed fragments delivering non-composite work products (free text, structured text, diagram,...) or even atomic fragments (see below). An example of composed fragment may consists in a portion of a process where the designer models use cases. This fragment delivers a work product (use case diagrams and a description text document) that is part of the System Analysis document produced by the System Analysis phase fragment.

Atomic fragment. An atomic (process) fragment delivers a portion of a work product and/or a set of system model constructs (in terms of their instantiation or refinement). A portion of a work product is here intended never to be a whole work product; in other words, atomic fragments never deliver entire work products. An atomic fragment may also not deliver a portion of work product but rather it may deliver a portion of the system model. An example of atomic fragment may be the identification of actors to be used for modeling use cases by starting from the analysis of some text describing system behavior.

Finally, as regard the MAS metamodel we performed an extended experimentation with some existing agent design processes. We realized that different types of metamodel may also be considered from the point of view of the set of constructs included in them:

Complete System Metamodel. It includes all the system metamodel constructs (elements and relationships) that are managed by the designers using a specific design process. This also includes all the constructs that are accepted as external inputs in the process.

Definable System Metamodel. It includes all the system metamodel constructs that are instantiated during the design process enactment. This is a subset of the complete system metamodel.

Workproduct System Metamodel. It only includes all the complete system metamodel constructs that are reported and drawn in the design process work products.

4 The Proposed Fragmentation Approach

Basing on the previous definition of process fragment and on the assumption that the MAS metamodel has a central role in designing multi-agent systems, we established guidelines for extracting process fragments from existing design processes. Let us suppose to have an agent design process anyway described and documented; it is not important if we have text documentation, produced in a formal way, or an oral representation of the design process. What is important is to have a complete description of the process in form of activities to be performed, work products to be delivered, stakeholders to be involved, the system metamodel to be instantiated and enactment guidelines (illustrating how to perform activities and to produce work products). These latter are often referred to as techniques.

The key idea is to consider the design process as a workflow of activities which bring to some kind of results; this can be done both with a mere sequential design process and an iterative and incremental one. In both the cases we can see the whole process as a temporal line where at each time t_i a portion of work (a set of activities) produces an outcome, a work product, a diagram or an entire model (see Figure 2).

The first step in the fragmentation process consists in analyzing the whole design process. This activity let us identify a set of portions of the process. Each portion of the process is composed of activities that begin at the time t_{n-1} and end at the time t_n, when it can be thought that a specific identifiable portion of the system has been designed; the time t_n is one of the cutting points we may identify.

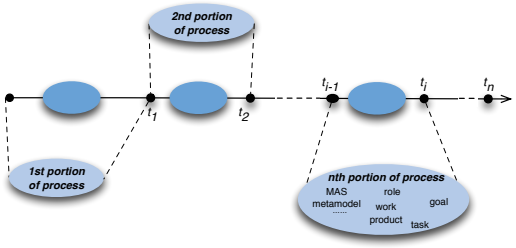

Fig. 2. The Design Process seen as a Temporal Line

This is the general rule for recognized cutting points; going into more details, the cutting points depend on the granularity of process fragment we want to extract, hence three different case may be pointed out:

– while extracting phase process fragment - the time t_i occurs when we can identify a set of work products representing a complete model of the system;

– while extracting composed process fragment - the time t_i occurs when we can identify a complete work product modeling a portion of the system;
– while extracting atomic process fragment - the time t_i occurs when during the hypothetical enactment process we complete the definition or the refinement of one element of the system model or we produce a consistent portion of a work product.

Once the cutting point has been identified the portion of process it delimits has to be investigated in order to identify the main elements that constitutes process fragment. The steps we ought to follow include:

1. identifying the main outcome, likely a work product or a portion of it or the definition of a MAS metamodel construct for atomic fragments.;
2. identifying the portion of the multi-agent system metamodel which elements are instantiated in the portion of work;
3. identifying all the elements (work product and/or MAS metamodel constructs) needed as input for carrying on the selected portion of work;
4. identifying for this portion of work all the elements that will form the process fragment: roles, tasks, goals of the fragment, etc.

4.1 The Details about the Guidelines

Figure 3 sketches the work the method designer does when he uses our guidelines. In order to better illustrate the guidelines let us follow an example for their application. Suppose the method designer wants to extract process fragments from the PASSI design process [7] and that he chooses the composed level of granularity.

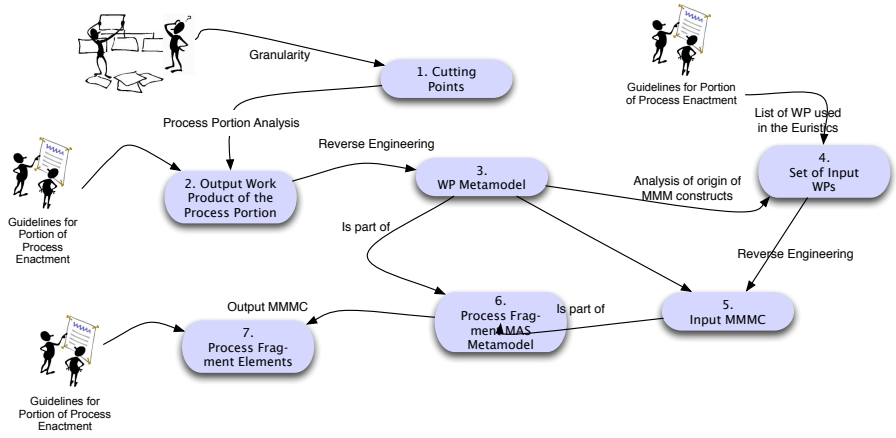

Fig. 3. The Guidelines for the Fragmentation Process

PASSI (Process for Agent Societies Specification and Implementation) [8] is an agent oriented design process for designing peer-agents and covers all the phases from the requirements analysis to the code of the multi-agent system. PASSI includes five phases arranged in an iterative/incremental process mode. Each phase produces a document that is usually composed aggregating UML models, text descriptions, tables, code, and so on produced during the related activities. Each phase is composed of one or more activities each one responsible for designing or refining one or more artefacts that are part of the corresponding model. For instance, the System Requirements model includes an agent identification diagram that is a kind of UML use case diagrams and also some text documents like a glossary and the system user scenarios.

After a first analysis the method designer may identify fifteen cutting points (remember that (s)he is looking for composed process fragments), each of them corresponds to the time t_i when a work product is delivered (see Figure 4).

Starting from the beginning, consider the portion of process before the time t_1, the work product of this first portion is the *Domain Requirements Description (DRD) diagram*, a functional description of the multi-agent system using UML use case diagrams with their textual description.

The method engineer analyzes this work product (Step 2. Output Work Product of the Process Portion in Figure 3) and through a reverse engineering process identifies the MAS metamodel construct (MMMC) here reported.

This activity is very simple when the work product is only in form of diagrams, in fact each graphical element of the diagram corresponds to one construct of the metamodel, the only thing to do is finding the mutual relationships among constructs. This descends from the assumption (section 3) we made about metamodeling: every element in the whole model of the system, in a work product or in a document specifying the system, is an instance of one construct of the MAS metamodel.

In the case of the DRD, when we draw the use case we instantiate a functional requirement, when we draw the “actor" we instantiate an actor and the associations are instance of generalize, include, extend and association relations.

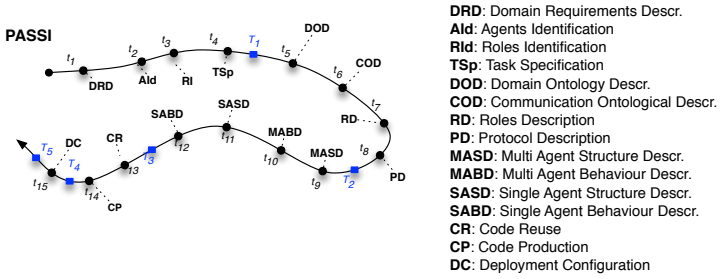

Fig. 4. The Cutting Points for PASSI

Figures 5.a) and 5.b) show an excerpt of the DRD diagram and the related work product MAS metamodel (Step 3. WP Metamodel).

The next step is to identify all the WPs that serve as inputs for carrying on the work in this first portion of process. In performing this activity the method engineer needs to consult the design process enactment guidelines. Roughly speaking, the work done during a process activity aims to define elements that the designer reports in the work product, for doing that (s)he makes some kind of reasoning, above all on other elements of the system domain. For instance, in order to elicit requirements of the system, the analyst often uses textual scenarios for gaining the interaction between the system and the users and the related functional requirements. This is the case of the DRD, by looking at the enactment guidelines the method engineer may deduce that the elements in the DRD work product are designed by managing the concept of *Scenario*.

Hence the set of input WPs (Step 4.Set of Input WPs) includes two text documents, the *Problem Statement* and *Scenarios*; the input MAS metamodel is only composed of the construct *User Scenario* (Step 5. Input MMMC). The constructs of the input MAS metamodel are identified in the same way of the step 3, through a reverse engineering process. Often, part of the constructs of the input MAS metamodel are reported, hence quoted, in the work product of the portion of work.

At this point the method designer has all the elements useful for establishing the complete MAS metamodel (Step 6. Process Fragment MAS Metamodel), indeed it is the sum of the previous two ones. In the case of our example the complete MAS metamodel is shown in Figure 6.

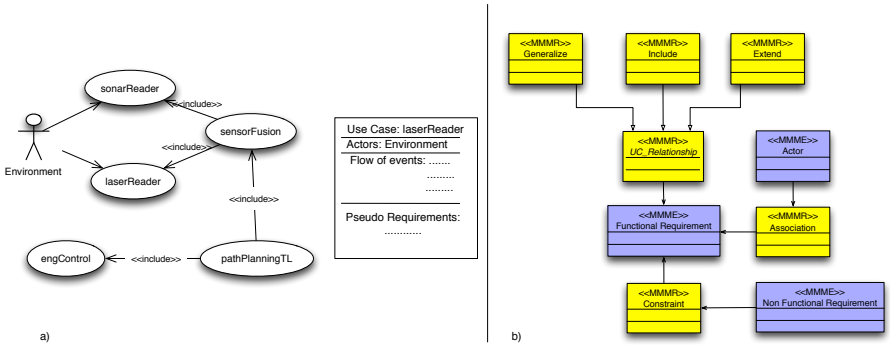

Fig. 5. a) An Excerpt of the DRD Work Product; b) The Work Product MAS Metamodel

After the sixth step, the extraction of the first composed process fragment is complete, the method designer must now (Step 7. Process Fragment Elements) identify the other elements in the fragment; first of all activities, roles

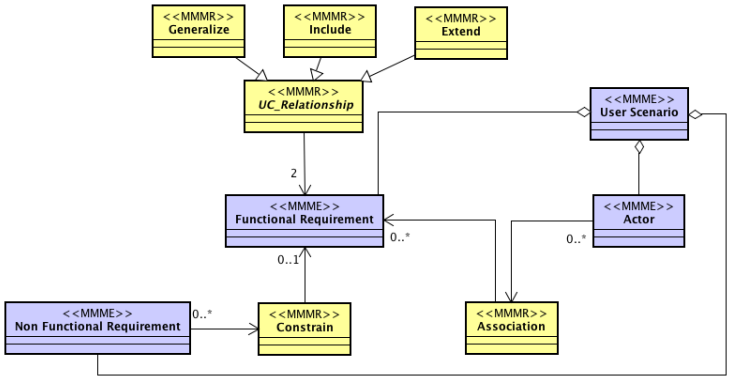

Fig. 6. The Complete MAS Metamodel for the DRD Process Portion

and work products and then all the others such as goals, reuse guidelines, and so on. Now that the first fragment has been identified and extracted the method designer may consider the rest of design process and he may extract all the other composed fragments.

The same guidelines may be applied for extracting phase or atomic fragments; in the case of a phase fragment for PASSI, the time T_1 (see Figure 4) is when the set of work products dealing with the problem domain description has been delivered; from the definition a phase fragment is the one delivering a set of work products that belong to the same design abstraction level.

In the PASSI example the method designer can find five different cutting points, they are illustrated in capital letter in the Figure 4 and correspond to the five main phases PASSI is composed of.

Once the method designer has found the cutting points, he can perform the same activities done for obtaining the composed process fragments. The number of work products to analyze is greater than one in fact one phase fragment can be considered a composition of composed process fragments.

Some differences are present when the method designer wants to extract an atomic fragment. An atomic fragment is different from the other two in the outcome, it has been conceived for representing the smaller piece of work the designer may perform. For instance all those situations in which the designer makes some kind of reasoning, sometimes without producing concrete outcome and following for instance some heuristics, for designing a specific MAS metamodel construct. Other times atomic fragments deliver portions of work products.

The cutting points can be identified following the process workflow and stopping it when a portion of work product is completed, in the sense that one element of the system model is completely defined. For instance, if a work product aims at defining the agents involved in the system and their goals, a part of the work product may be the one listing or representing all the agents, supposing

that the flow of work for producing that work product implies firstly to find all the involved agents and then to identify their goals.

In the PASSI example, the first part of the design process prescribes that the system analyst and the domain expert collaborates in identifying use cases and then they describe them in order to produce the previous said DRD diagram. The first part of the work does not imperatively result in a work product, the analyst and the designer might list the use cases in a document or they might keep them in their mind and then represent them in the diagram (in a following portion of work that will be part of another atomic fragment).
Whatever is the result, the work done aims to instantiate several times the functional requirement construct in order to obtain use cases; this portion of work can represent an atomic fragment and has to be described and represented as the others two.

We claim that the proposed guidelines can be applied for extracting process fragments from agent design processes also described in an informal or not well structured way; the work to be done in this cases could be a little more demanding but it leads anyway to good results. The best result is obtained if the agent design processes were documented using the FIPA IEEE standard template [17].

This specification requires to emphasize the main elements of design process we shown in Figure 1 by means of SPEM [22] activity and class diagrams thus providing dynamic and structural views of the design process parts, metamodels, tables for representing input/output workproduct, input/output MAS metamodel constructs and other elements that allow a quick and a complete view on the overall design process. Using this kind of representation leads to a very quick and easy application of the guidelines since the cutting points can be rapidly identified and then all the information is visible and identifiable[5].

It is worth to note that basing on our experience the best way for extracting fragments from a design process is using a top-down approach, hence we suggest starting with the extraction of all the phase process fragments and then the composed process fragments. Once all the composed process fragments have been extracted the atomic fragments may be extracted, indeed once we have the composed fragments we also have all the work product MAS metamodels and the complete MAS metamodel constructs, so we can decide which construct we are interested to and then extract the portion of work dealing with it. In this case there is only a little difference in the use of guidelines for atomic fragments that allow saving lot of time being the atomic the most detailed, and full of information, process fragment.

5 Discussions

The guidelines for extracting process fragments we illustrated in the previous section are only a part of the whole approach to Situational Method Engineering we developed; it includes the creation of the repository [26], the formalization of

[5] The specification can be found in
http://fipa.org/specs/fipa00097/index.html.

a SME process phases (selection, retrieval and assembly of fragments) [27] and, the most important, the definition of the process fragment, what it is composed of and the best way to document it [9][25].

These guidelines have been created in a way that lets the method engineer have all the elements for documenting the process fragments and respect the improved definition we gave in this paper and the template we proposed in [25]. Documenting the process fragment using this template aims to mainly put to evidence the process and the product part of the fragment.

One of the major advantages introduced by the proposed method for fragmenting agent design processes, and then extracting process fragments from them, is simplifying the part of situational method engineering devoted to the creation of fragment repositories. Until now, this topic has not been widely considered, indeed commonly the work of the method designer is thought to start from an already filled repository.

It is worth to note that the proposed method fragment definition gives the possibility of creating an agent fragments repository where all the fragments are homogeneous because they share the same definition and above all they are structured basing on a core concept, the multi-agent system metamodel for which in [10] we illustrated how it is constructed and all the rules for extracting knowledge on the process.

With this work we, now, have all the ingredients for increasing the fragment reusability, hence selection and then assembly and for establishing the basis of a formal approach to SME for agent design process creation. Besides having a well specified set of data regarding the process fragment (the metamodel, the activities, the role and the work products, all represented in a well defined fashion) gives the possibility of automatically supporting the overall SME approach and making some kind of reasoning that might lead, for instance, to automatic selection or assembly of fragments.

6 Conclusion and Future Works

In this paper we propose guidelines for fragmenting and then extracting fragments from agent oriented design processes. The guidelines are based on a definition of process fragment that has been improved from the previous one illustrated in [9].

Our aim is to handle a part of the Situational Method Engineering that, in all the existing approaches, has not been treated with the right details. Indeed, in most cases, applying a SME approach means to start from an already filled repository; how to construct the repository has not been sufficiently investigated yet. Every existing SME approach implies a selection phase from a repository, we claim that if this one is not realized in such a way that the fragments were homogeneous and with easily identifiable interfaces, it would be very difficult to select and above all to assemble the right fragments meeting the method engineer needs.

It is anyway difficult to extract process fragments from existing agent oriented design processes because of the lack of a proper documentation highlighting the elements characterizing the process fragment, they are: activities, roles and work product, hence the work to be done for obtaining a design result and the stakeholder performing the work. A fourth element, very important in our fragment definition, is the multi-agent system metamodel; by using it we are able to overcome all the limitations said before and to provide guidelines for extracting fragments thus obtaining process fragments of the same granularity.

In the future we plan to develop a tool for aiding the method designer in automatically (or semi-automatically) selecting and assembling process fragments; this will be done exploiting the structure of the fragments and the fact that a lot of important information inside the process can be inferred by means of the system metamodel.

Acknowledgments. This research has been partially supported by the EU project FP7-Humanobs and by the FRASI project managed by MIUR (D.M. n593).

References

1. P. Backlund, J. Ralyté, M. Jeusfeld, H. Kühn, N. Arni-Bloch, J. Goossenaerts, and F. Lillehagen. An interoperability classification framework for method chunk repositories. *Advances in Information Systems Development*, pages 153–166, 2007.
2. C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard. Engineering adaptive multi-agent systems: the adelfe methodology. In *Agent Oriented Methodologies*, chapter VII, pages 172–202. Idea Group Publishing, 2005.
3. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An agent-oriented software development methodology. *Autonomous Agent and Multi-Agent Systems (8)*, 3:203–236, 2004.
4. S. Brinkkemper. Method engineering: engineering the information systems development methods and tools. *Information and Software Technology*, 37(11), 1996.
5. S. Brinkkemper, K. Lyytinen, and R. Welke. Method engineering: Principles of method construction and tool support. *International Federational for Information Processing 65*, 65:336, 1996.
6. S. Brinkkemper, R. Welke, and K. Lyytinen. *Method Engineering: Principles of Method Construction and Tool Support.* Springer, 1996.
7. M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies* [14], chapter IV, pages 79–106.
8. M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies* [14], chapter IV, pages 79–106.
9. 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.
10. M. Cossentino and V. Seidita. Metamodeling: Representing and modeling system knowledge in design processes. Technical Report 11-02, Technical Report ICAR-CNR, 29 July 2011.
11. A. Fuggetta. Softaware process: a roadmap. In *In Proceedings of the Conference on the Future of Software Engineering.ACM Press, New York (USA)*, pages 25–34, Limerick (Ireland), June 4-11 2000.

12. D. Gupta and N. Prakash. Engineering Methods from Method Requirements Specifications. *Requirements Engineering*, 6(3):135–160, 2001.
13. B. Henderson-Sellers. Method engineering: Theory and practice. In D. Karagiannis and e. Mayr, H. C., editors, *Information Systems Technology and its Applications.*, pages 13–23, 2006.
14. B. Henderson-Sellers and P. Giorgini. *Agent Oriented Methodologies.* Idea Group Publishing, Hershey, PA, USA, June 2005.
15. B. Henderson-Sellers, C. Gonzalez-Perez, and J. Ralyté. Comparison of method chunks and method fragments for situational method engineering. *Software Engineering Conference, Australian*, 0:479–488, 2008.
16. B. Henderson-Sellers and J. Ralyté. Situational method engineering: State-of-the-art review. *J. UCS*, 16(3):424–478, 2010.
17. IEEE Foundation for Intelligent Physical Agents. *Design Process Documentation Template, Document number XC00097A-Experimental*, 2011.
18. F. Karlsson and P. Agerfalk. Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, 46(9):619–633, 2004.
19. K. Kumar and R. Welke. Methodology engineering: a proposal for situation-specific methodology construction. *Challenges and Strategies for Research in Systems Development*, pages 257–269, 1992.
20. I. Mirbel and J. Ralyté. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11(1):58–78, 2006.
21. J. Pavòn, J. J. Gòmez-Sanz, and R. Fuentes. The INGENIAS methodology and tools. In *Agent Oriented Methodologies* [14], chapter IX, pages 236–276.
22. S. process engineering metamodel. Version 2.0. Final Adopted Specification ptc/07 03-03. Object management group (omg), march 2007.
23. J. Ralyté, R. Deneckère, and C. Rolland. Towards a generic model for situational method engineering. In *Advanced information systems engineering: 15th international conference, CAiSE 2003, Klagenfurt/Velden, Austria, June 16-20, 2003*, volume 15, pages 95–110. Springer Verlag, 2003.
24. J. RalytÈ and C. Rolland. An approach for method reengineering. *Lecture Notes in Computer Science*, pages 27–30, 2001.
25. V. Seidita, M. Cossentino, and A. Chella. A proposal of process fragment denition and documentation. *Proceedings of the nineth European Workshop on Multi-Agent Systems (EUMAS'11)*, 2011.
26. 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.
27. V. Seidita, M. Cossentino, V. Hilaire, N. Gaud, S. Galland, A. Koukam, and S. Gaglio. The metamodel: a starting point for design processes construction. *International Journal of Software Engineering and Knowledge Engineering.*, 20(4):575–608, 2010.
28. K. Slooten and S. Brinkkemper. A method engineering approach to information systems development. In *Proceedings of the IFIP WG8. 1 Working Conference on Information System Development Process*, pages 167–186. North-Holland Publishing Co., 1993.
29. ter Hofstede A.H.M. and V. T.F. On the feasibility of situational method engineering. *Information Systems.*, 22(6/7):401–422, 1997.