A Proposal of Process Fragment Definition and Documentation

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

[1] Dipartimento di Ingegneria Chimica Gestionale Informatica Meccanica
Università degli Studi di Palermo, Italy
`{seidita,chella}@dinfo.unipa.it`

[2] Istituto di Reti e Calcolo ad Alte Prestazioni, Consiglio Nazionale delle Ricerche - ICAR/CNR Palermo, Italy
`cossentino@pa.icar.cnr.it`

Abstract. This paper focuses on the field of Situational Method Engineering (SME) for the construction of agent-oriented design processes. Whatever SME approach a method designer wants to use, he has to manage two main elements: the (method or process) fragment and the repository where it is stored. Specific fragment definition and documentation are fundamental during these activities, for new process composition, and for the consequent system design activities. This paper aims at illustrating a proposal of fragment definition and documentation. This proposal is aimed to be an input for the IEEE FIPA Design Process Documentation and Fragmentation working group and, as regards our own research work, this is the ideal completion of the methodological practices prescribed in the PRoDe approach for new processes composition.

1 Introduction

The work presented in this paper starts and is based on what we have done during the latest years towards the definition of the best way to create ad-hoc agent oriented design processes. The development of a multi agent system always requires great efforts in learning and using an existing design process.

It has been said and heard several times that it does not exist one design process (or also a methodology or a method) to develop systems able to solve every kind of problems and that there is the need for creating techniques and tools for a designer to develop an ad-hoc design process prior to use it on the base of his own needs [11][20][19].

In order to solve this problem and to give means for one to develop an agent system using the "right" design process, we adopted the (Situational) Method Engineering approach and we started from pointing out what we intend for design process. (Situational) Method Engineering [8][2][11][15] provides tools and techniques for creating design processes by reusing portion of existing ones, called *method fragment*, stored in a repository, the *method base*.

In [4] the main elements of an agent-oriented design process have been identified, they fundamentally ground on three of the main elements a designer always meets during design, they refer to the *stakeholders* that perform *activities* in order to produce *design results* (also labelled *work products or artefacts*).

The key idea of our approach is that this core triad has to be augmented by another important element: the *system metamodel.* This concern, also deduced from the MDE [16] approach, led to the consideration that producing design results, is nothing else but instantiating elements from a (meta-)model.

The system metamodel is the fundamental element to be considered following the (Situational) Method Engineering approach, PRoDe (the Process for designing Design Processes), we have recently created [17]. In PRoDe the creation of design processes can be done by following specific phases from analysis to implementation. The system metamodel contains the set of constructs that will satisfy the system requirements

A lot of existing (Situational) Method Engineering approaches exist [7][14][12][9][1], they are developed around three main phases: the *process requirements analysis*, the *process fragments selection* and the *process fragment assembly.* The principal aim of SME is to manage the *method fragment.*

Nowadays, there are a lot of definitions of method fragment in the research on (Situational) Method Engineering. We claim that none of them can be universally applied. Different (Situational) Method Engineering approaches own different notion of method fragment and as a consequence they use proprietary repositories. Actually, this reason and the lack of a unique fragment interface severely limit the availability of repositories.

In this paper we focus on the process fragment definition and documentation by identifying its main elements and following a twofold aim: reuse, in terms of providing all the information for supporting the selection and assembly phases, and reuse in a more general design point of view, hence providing information useful to designers. During system development, designer needs guidelines on the portion of work described in the fragment and how to produce the related artefacts. So the main notion, we deal with, is the System Metamodel that represents the major improvement to the work proposed in [17].

Our aim is to give a definition aiming at well documenting the process fragment. We pursue a twofold objective: 1) using the fragment at the system design time and 2) reusing in storage and assembly, in so doing we lay the foundations for establishing a standard definition of fragment and the related standard documentation. This work is the natural prosecution of the work done by the authors within the IEEE FIPA Design Process Documentation and Fragmentation working group that already resulted in the standard way of documenting design processes [10].

In the following sections the definition and a template for documentation of process fragment will be shown together with an example of documentation.

2 Background and Motivation

During the latest years the fact that the projects' features and organization specificities greatly influence the software development methods has become evident. Besides the more the software systems become complex the more this fact becomes urgent. The consequence is that it does not exist a unique development method that could fit every kind of needs organizations could present and that can be used for engineering every kind of software systems.

In this scenario it is increasing the need for techniques allowing organizations to create and then to use their own development method(s). Specific development methods could take into account the kind of problems the organization is devoted to solve and the characteristics they present in terms of designers/developers skills, known and used tools and so on.

The discipline of Method Engineering has faced this problem and some important results has been reached. The Method Engineering has been defined by Brinkkemper et al. [2] as "the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems ". Method engineering aims to accomplish two different scopes: the first is to create situation specific methods for meeting organizational features and represent a sort of choice list, the second is to produce the so called method "on the fly". Hence the system development implies and starts with the definition of development methods that fit specific project situations (this is the matter of a sub-area of Method Engineering (ME), the Situational Method Engineering (SME) [11]).

The best and quickest way to develop situation specific methods is reusing existing ones. For these purposes (S)ME prescribes to break down existing methods into "components" that may be stored in a repository. These components may be retrieved (analyzed and then selected) from the repository in order to be composed/assembled in a new method fitting project/organization needs. They can also be used as they are or adapted in order to best fit specific needs or in order to facilitate the composition of the process.

Still open issues are: the definition of the components and their granularity, how they have to be selected from the repository and how they can be assembled.

In the past the authors developed an approach for new design processes composition (PRoDe [17]) that entails the aforementioned "component" namely the *Process Fragment* and also the *System Metamodel*.

> The ***Process Fragment*** is a portion of design process adequately created and structured for being reused during the composition and enactment of new design processes both in the field of agent oriented software engineering and in other ones (model driven engineering-based approaches are preferred fields of application for the proposed definition).

> The ***System Metamodel*** is the definition of constructs needed for creating system models.

It is our belief that during the enactment of the methods one (or more) process role refers, more or less knowingly, to the metamodel in order to produce work products where instances of a set of metamodel constructs are managed (more details about this argument can be found in [10]).

Managing process fragments is the main aim of PRoDe, it covers the three main phases of SME, the process requirement analysis and the definition of process fragment, the selection and then the assembly. Because of our conviction about the importance of the system metamodel in all the design activities, it has a central role in PRoDe.

The first activity in the PRoDe approach entails a set of steps that, starting from the process requirements, are able to produce the system metamodel or in any case a first draft of it. PRoDe is iterative, the new design process, after a first enactment, might be modified/enhanced due to test results and new requirements identification. As regard selection and assembly the PRoDe approach provides a well defined set of activities for identifying and retrieving fragments from repository basing on some considerations made on the system metamodel [17]. The PRoDe activities, as well as other SME approach activities, are also highly grounded on the SME fundamental element, the *Process Fragment* (or method fragment or chunk or simply fragment - however it is named by different researchers), and obviously on the repository aimed at storing it. In order to apply a SME approach in the most fruitful way, a well done definition and documentation of process fragment is useful for properly storing, selecting and assembling new design processes whatever SME approach one wants to follow.

The process fragment definition together with the specific SME process (see for instance [17]) used for retrieving and composing fragments may notably influence how the repository is conceived and constructed. We try to not take this chance by using the definition we propose in the following section.

3 The Process Fragment Definition

Figure 1 represents all the elements composing a Process Fragment. It contains all the elements useful for representing and documenting the fragment under the process, product and reuse point of view; the proposed fragment documentation template, that will be presented in the following section, slavishly follows the proposed representation, its elements and their definitions.

The root element, the *Process Fragment*, has been generally extracted from an existing design process, therefore an important information to be stored in the repository is the *Design Process* the fragment refers to. This serves for the designer to set the application context and the particular features the fragment would exhibit. The *Process Fragment* can be of three different levels of granularity: *phase*, *composed* and *atomic*, each of them is related to the quantity of work to be done and to the complexity of the produced outcome.

- A *phase (process) fragment* delivers a set of work products belonging to the same design abstraction level of the design flow. Such a work product may belong to any of the cited work product types. An example 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).

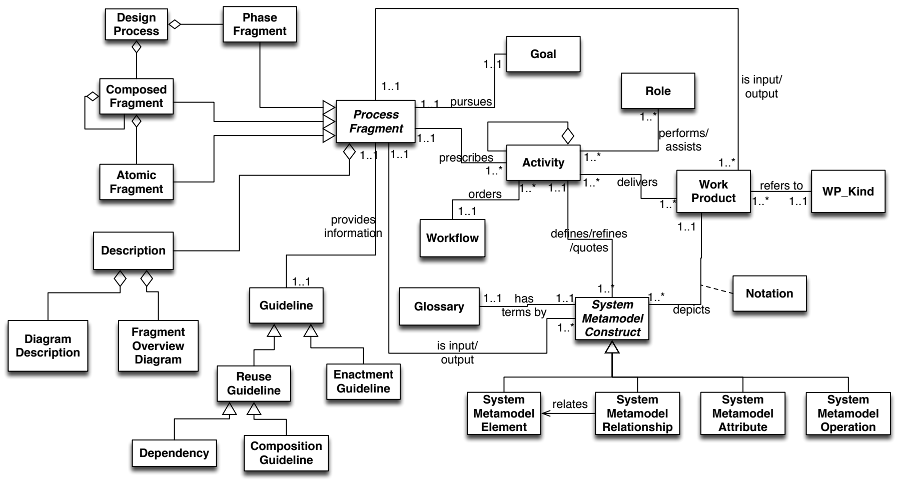


Fig. 1. The Process Fragment View

- A *composed (process) fragment* delivers a work product (or a set of instances of the same work product). Such a work product may belong to any of the cited work product types.
- 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.

The process fragment prescribes some activities to do, each of them is a portion of work that has to be performed by one or more stakeholders (*Roles*).

Activity delivers *Work Products*, where the results of design activities are drawn by using a specific *Notation* and each work product is developed under the responsibility of one role. The notation to be used greatly influences the flow of work to be done for producing a work product and for this reason a fragment has to be supplied with a set of *Guidelines*. As regards the process and product perspective of the fragment the *Enactment Guidelines* provides all the elements, description and so on, for applying the workflow prescribed in the fragment.

It is not mandatory to follow a specific notation, the same kind of diagram (for instance a structural one) may be expressed by using different notations without significant differences in the resulting expressiveness. Moreover, different kinds (*WP_Kind*) of work products can be delivered. We identified two main work product kinds: graphical and textual, the former when an activity results in a diagram, the second when designers produce textual documents. Finally a work product can be of composite kind if it is a composition of the previous said kinds, for instance a document with a diagram and the text explaining it (more details can be found in [18]).

As well as in the design process definition, one of the most important elements in the fragment definition is the *(Multi-Agent) System Metamodel* (Multi-Agent SMM); each fragment is based on a system metamodel that is obviously a part of the metamodel of the design process it comes from. The metamodel contains the set of constructs representing the (portion of) system to be designed using a specific process fragment. We consider System Metamodel composed of constructs that can be elements (*SMME* - the concepts to be designed), relationships among them (*SMMR*), attributes (*SMMA*) and operations (*SMMO*) for respectively representing a particular feature and the behavioral characteristics of an element (see [6] for further details).

The main aim of process fragment is to instantiate one (or more) system metamodel construct(s) (SMMC) and in so doing it may be requested to define relationships among elements or to quote other elements and/or relationships; besides the result of defining an element or a relationship might be the refinement of existing elements or relationships. This fact led to the definition of the kinds of action to be done on a system metamodel construct (see the following section for details). Finally SMMC has a definition to be listed in a glossary; the definition is mainly useful during selection when the method designer wants know which kind of metamodel construct better fits with the metamodel construct s/he is dealing with.

Until now we explored the process and product part of the fragment through a set of elements that has to be necessarily present in the fragment documentation, now let us quickly focus on the elements that principally deal with the reuse aspect of the fragment: *Goal*, and *Dependency* guidelines. The fragment goal is the objective the process part of the fragment wants to pursue and it is to be used during fragment selection from the repository. For this reason it is related to the new design process requirements, in other words, a goal describes the contribution a fragment may give to the accomplishment of some design process requirements.

The dependency guideline aims at describing specific constraints, if they exist, for the fragment to be composed with other ones, for instance, there can be fragments dealing with system metamodel elements that are very specific to particular application domains, in this case it should be possible that such fragments can be composed with fragments coming from the same classes of design processes.

It is important noting that the way the work has to be performed inside one fragment may slightly change depending on the notation of the work product produced; if the result has to be a graphical work product the activity and the related guidelines are different if we want to use two different notations. Since the fragment aims at designing a specific system metamodel construct, we can consider the fragment itself independent from the specific notation. The same result can be obtained by producing different work products in different notations.

In the following table we give the detailed definition of all the elements composing a process fragment:

Table 1: Process Fragment Elements Definitions

Term	**Definition**
Design Process	It is the design process from which the fragment has been extracted.
Phase	A specification of the fragment position in the design workflow. Usually referring to a taxonomy (i.e. Requirements elicitation, Analysis, Design, etc.)
Goal	The process-oriented objective of the fragment.
Activity	A portion of work assignable to a performer (role). An activity may be atomic (sometimes addressed as Action) or composed by other activities.
Work Product	The resulting product of the work done in the fragment; it can be realized in different ways (diagram, text,..) also depending on the specific adopted notation.
WP_Kind	Represents the specific kind one work product can be; it strictly depends on the means the adopted notation provides. One work product can be: Structured or Free text, Structural, Behavioural or Composite
Notation	Each deliverable can be drawn by using a specific notation. Concepts dealt by the fragment have to find a mapping in the notation. Notation usually includes a metamodel and a set of pictorial prescriptions used to represent the instantiation of metamodel elements.
Role	The stakeholder performing the work in the process and responsible of producing a work product (or a part of it). Usually referring to a taxonomy (i.e. System Analyst, Test Designer, etc.)
System Metamodel Construct	(abstract class) The concept the fragment deals with, for instance a fragment aiming at defining the system requirements has to define and to identify the concept of requirements. Each metamodel construct has to be defined during, at least, one portion of process work and has to appear in at least one work product.
System Metamodel Element	It is an entity of the metamodel that is instantiable into an entity of the system model. Examples of System Metamodel Elements (SMME) are: classes, use cases,. . . .
System Metamodel Relationship	It is the construct used for representing the existence of a relationship between two (or more) instances of SMMEs. For instance, the aggregation relationship among two instances of a SMME class is an instance of the SMMR association.
System Metamodel Attribute	It is a particular kind of elements used for adding properties to SMMEs. *An SMMA is a structural feature and it relates an instance of the class to a value or collection of values of the type of the attribute.* [21]. The attributes type is a SMME.
System Metamodel Operation	It is a behavioral feature of a classifier that specifies the name, type, parameters, and constraints for invoking an associated behavior [21].
Glossary	A list of definitions for the system metamodel constructs.
Description	It is the textual and pictural description of the fragment; it provides a bird-eye on the whole process the fragments comes from and the fragment overview in terms of tasks to be performed, roles and work product kind to be delivered.

Table 1: continues in the following page

Table 1: continues from the previous page

Term	Definition
Composition Guideline	A set of guidelines for assembling/composing the fragments with others. This may include notational specifications, and constraints (also addressing issues like platform to be used for system implementation and application area)
Dependency	The description of specific dependencies of this fragment from other ones; it is useful for composition.
Enactment Guideline	The description of how to perform the prescribed activity. This may include best practices and specific techniques for achieving the expected results.

Table 1: ends in this page

4 The Process Fragment Documentation

The document used for the Process Fragment description is made of six main sections (the template is shown in Figure 2), each of them refers to one (or a set of) element(s) of the Process Fragment representation (see Figure 1). Three sections deals with the three main elements a design process is composed of, as we stated in section 2, they are: *Stakeholders*, *Workflow* and *Deliverable*, hence the description of who performs the work to be done and how, in order to deliver an artefact of the system model.

The *Stakeholders* have to be simply described through the name and the description of the activities (the work) their are responsible for. They are named Role in compliance with SPEM 2.0 [13].

The *Workflow* section serves for documenting all that regards the structure of the portion of work to be done in the process fragment. It covers the set of procedural rules for sequencing design activities and documents/artefacts exchanged among Roles in order to produce the main output of the fragment.

The concept of workflow we had in mind when we created this document template is the one introduced by [22], it is structured by work breakdown elements that give us the possibility to represent portion of design work at every level of granularity, hence we can represent phases, activities and tasks.

The Workflow description is made with one SPEM 2.0 activity diagram that represents the portion of work related to the role performing it and all the needed input and output documents. Each work breakdown element is completed with a textual description of information such as the name, the kind e.g. if it is a task, an activity or other else, the description and the roles involved in the work. Besides the list of all the input and output system metamodel constructs and the list of all the input and output work products are needed in order to have means for analyzing the process fragment, also automatically, during the selection and assembly phase when a new design process is being creating.

The *Deliverable* section is made of two main parts, the first deals with the truly description of the document kinds to be produced in order to provide guidelines for producing them and the second handles the relationships of the work product with the constructs of the system metamodel here managed. So in

1. Fragment Description
    1.1. Fragment Goal
    1.2. Fragment Granularity
        1.2.1. Composing fragments
    1.3. Fragment Origin
        1.3.1. The Process Lifecycle
    1.4. Fragment Overview
2. System metamodel
    2.1. Definition of System metamodel elements
    2.2. Definition of System metamodel relationships
    2.3. Definition of System metamodel attributes
    2.4. Definition of System metamodel operations
        2.4.1. Fragment Input/Output in Terms of System Metamodel Constructs
        2.4.2. Definition of input system metamodel constructs
3. Stakeholders
    3.1. Role 1
4. Workflow
    4.1. Workflow description
    4.2. Work Break Down Elements description
    4.3. Work Break Down Elements' input/output in terms of system metamodel constructs
    4.4. Fragment's Input/Output in terms of Work Product
5. Deliverables
    5.1. Document name
        5.1.1. Deliverable notation
        5.1.2. Deliverable content in terms of system metamodel constructs
6. Guidelines
    6.1. Enactment Guidelines
    6.2. Reuse Guidelines
        6.2.1. Composition
        6.2.2. Dependency Relationship with other fragments
7. Glossary
8. References

Fig. 2. Process Fragment Document Template

the first part of the section the description on how to produce the work product and an example on the specific notation used are given.

This part of the document aims at exhaustively providing all the information for the designer to produce the deliverables. In the second part of this section the said relationships are represented in a particular kind of diagram that the authors created by extending SPEM 2.0 [6], namely the *work product content diagram*.

The word *content* let us understand that this diagram aims at having a complete and detailed view on the elements managed during the production of the work product. Exactly this diagram collects all the system metamodel constructs that are managed during the enactment of the process fragment and are also reported in the work product, hence the design process input constructs that are not reported in the work product are not shown in this diagram.

Input constructs are used by designer for the analysis and for reasoning about the system to be produced. In the content diagram we also report information about the type of design actions made on each construct.

One specific design action is made on each metamodel construct and it is useful for catching various information about the fragment and the resulting work product. The list of possible design actions has been identified by analyzing the way of working of designer; we used for that a lot of agent oriented design processes under the hypothesis that each work product production aims at instantiating at least one metamodel construct.

Instantiating means defining one or more instances of metamodel construct that have to be represented in the work product following one specific notation. Often, during the definition of one construct designer needs to consider other constructs already defined in other process fragment and/or to report them in the work product he is producing. Another frequent situation is when designer relates one instance of one construct to another one, for instance a generalization among classes, in this case he defines a relationship.

Finally designer could need to refine constructs by adding information or features to an already defined one, in this case he defines attributes and/or operations for that construct. Therefore the possible design actions to be made on system metamodel constructs are:

- ***define***, instantiation of construct (element, relationship, attribute and operation), the label used is **D** for all construct except for the relationship in which case it is **R**,
- ***quote***, reporting a construct in the work product, the labels used are **Q**, **QR**, **QA** and **QO** respectively for element, relationship, attribute and operation. Quotation also introduces relationship, hence dependency, with other work products.

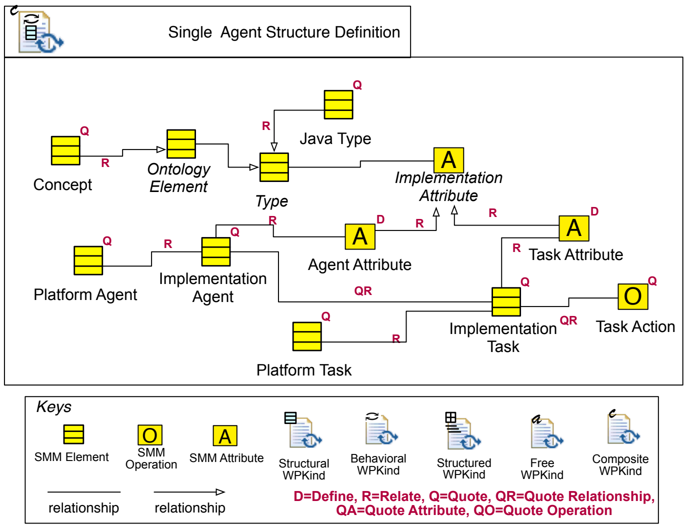


Fig. 3. An example of Work Product Content Diagram

In Figure 3 an example of work product content diagram is given, it represents the outcome of the Single Agent Structure Definition process fragment extracted from PASSI [3]; here we can see that the aim of this process fragment is to produce a work product where the Implementation Agent and Implementation Task are respectively refined by adding the Agent Attribute and the Task Attribute, hence these latter are defined whereas the former are quoted and related with them. Besides, in order to define the Agent Attribute and the Task Attribute, Concept and Java Type have to be quoted. Finally the work product prescribes to also report the Task Action and the Platform Agent that are consequently quoted. It is worth noting that the notational symbol used for System Metamodel Construct (SMMC) is not used for system metamodel relationship, even if we understand that this is not stylistically correct from a notational point of view we prefer to maintain that for reducing the complexity of producing and reading this kind of diagrams. Besides there can be more than one relationship among instances of the same constructs and this is shown by the number close to the R label. This kind of diagram let designer to easily identify all the system metamodel constructs the fragment is devoted to manage.

Another notational element that can be seen in this diagram and that is largely used in all the SPEM 2.0 diagrams of the fragment documentation is the Work Product Kind. Briefly, we needed to represent different kinds of work product so we extended SPEM in order to include the following kinds [3]:

- Behavioural, it is a graphical kind of work product and is used to represent the dynamic aspect of the system (for instance a sequence diagram representing the flow of messages among agents along time);
- Structural, it is also a graphical kind of work product and is used for representing the static aspect of the system, for instance a UML class diagram;
- Structured, it is a text document ruled by a particular template or grammar, for instance a table or a code document;
- Free, it a document freely written in natural language;
- Composite, this work product can be made by composing the previous work product kinds, for instance a diagram with a portion of text used for its description.

The main aim of the section on Deliverables is to provide, among the others, some kind of guidelines for producing the work product. Another kind of guidelines has to be documented in the *Guidelines* section, here there are two types of guidelines, the *enactment* and the *reuse*. The enactment guideline provides a textual description on how to carry out the work in the fragment by referring and describing in details how to manage the system metamodel constructs of the fragment.

The aim of the reuse guidelines is very different, they are directed to the reuse possibility of the fragment thus providing suggestions for composing the fragment with other ones and the dependencies from other fragments. Reuse guidelines supplies another view on the dependencies of the fragment already

[3] Definitions reported from our previous work on the matter in [18]

Fragment Goal
Describing semantic agent communications in terms of exchanged knowledge (referred to an ontology), content language and interaction protocol.
Fragment Origin.
The presented fragment has been extracted from PASSI (Process for Agent Societies Specification and Implementation) design process. PASSI (Process for Agent Societies Specification and Implementation) is a step-by-step requirement-to-code methodology for designing and developing multi-agent societies. The methodology integrates design models and concepts from both Object-Oriented software engineering and artificial intelligence approaches. PASSI has been conceived in order to design FIPA-compliant agent-based systems, initially for robotics and information systems applications. . . .

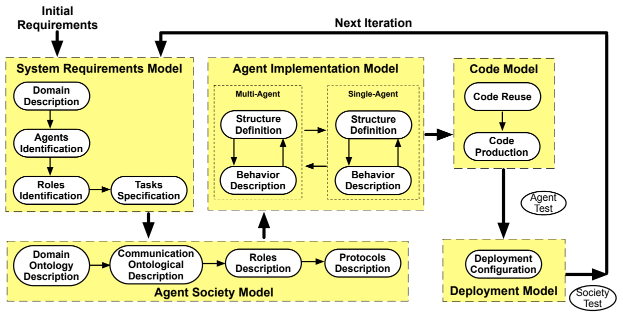


Fig. 4. A Portion of the COD Fragment Document - The Fragment Description

visible in the workflow description by means of the input work products and in the content diagram by means of the quoted elements so as in all the tables describing input and output constructs of the fragment.

What we consider the key concept of our approach to SME for representing design process in general and process fragment in particular, the *System Metamodel*, is documented in the second section on the proposed template. The attention paid to the System Metamodel, how it is conceived and it is composed of is the most important improvement the authors give to the previous fragment definition made in [17]. The section on metamodels includes a class diagram for representing the Complete System Metamodel of the fragment and the definition of each SMM Construct together with, when applicable, a statechart describing its different states while managed by the designer during the application of the fragment. An example reported from the Communication Ontological Description (COD) process fragment is shown later in Figure 5 (consider that this and the following figures are extracted from the COD documentation so other figures are present with different numeration). Then all the input/output system metamodel constructs are listed in a table where a distinction is made between the constructs to be designed and the ones to be quoted.

The first section regards the *Fragment Description* that includes: the *goal*, the *granularity*, the *origin* of the fragment and an *overview* on the fragment. The description of the fragment *goal* aims to provide the reader with a quick understanding of the goal pursued by the process fragment using a simple sentence like, for instance, "the aim of this fragment is collecting requirements",

possibly relating the description to common-sense in software engineering. The goal serves mainly in giving a mean for the method designer to select the right fragment for his purposes.
As regard the granularity, it establishes the length of the work done in the fragment and in some way the complexity of the fragment in terms of work product. As already said there can be three kinds of fragment: *phase*, *composed* and *atomic* (see also Figure 1).

Finally *Glossary* and *References* completes the documentation by providing useful description of the most important terms used in the fragment and a list of references for improving knowledge on the fragment, above all on the origin, the application context and so on.

5 An Example of Process Fragment Document

In the following an example of fragment documentation is given through a set of figures that we extracted from the document related to the *Communication Ontological Descritpion - COD* process fragment from PASSI [3]. Each figure represents a relevant portion of the document, the complete version of this fragment can be found in the FIPA DPDF working group website[4].

Looking at the fragment outline, it can be seen that first of all we focus on the fragment presentation through its goal and its origin, in so doing we reach a twofold objective, letting the designer have a quick idea on the focus and the domain in which the fragment might work and allowing a sort of automatic or semiautomatic selection of the fragment. Figure 4 shows the fragment goal, it is described in a very concise textual form that puts in evidence the main elements the fragment will deal with, for instance it can be noticed the words *agent communication*, *knowledge* and *protocol*. It is to be hoped that this part of the document were compiled using words focussing on the fragment scope. Figure 4 also shows a portion of the section dedicated to the design process the fragment has been extracted from, the importance of this early discussion has been already said.

As well as a design process, each process fragment is based on a MAS metamodel composed of elements and relationships; the fragment document has to explore this issue and to show all the elements type to be defined/quoted/related in the fragment. Figure 5 shows the COD document section about System metamodel. As already said the process fragment description, and documentation, is principally aimed at showing the process and product part of the fragment for easily identifying the way in which it can be reused. Figure 6 refers to the fragment description section and details the inputs, the outputs and the fragment workflow through a SPEM 2.0 activity diagram.

The fragment document continues with an example and the explanation on how to produce the work product (see Figure 7) and with a set of composition guidelines and dependency relationships.

[4] http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/docs.htm

System metamodel

The portion of metamodel of this fragment is:

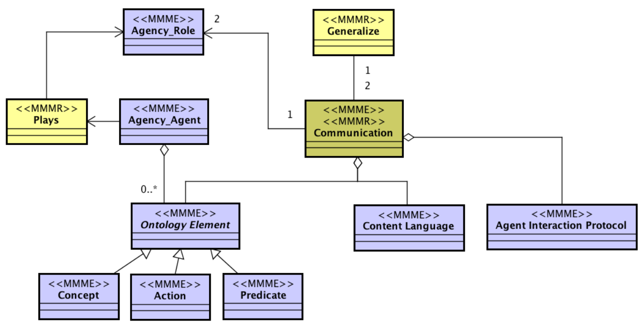


Figure a. The fragment System metamodel

This fragment refers to the MAS meta-model adopted in PASSI and contributes to define and describe the elements reported in Figure a.

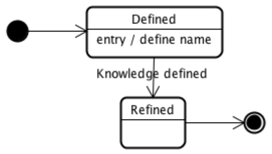
Definition of System metamodel elements.

This fragment underpins the following model elements:

Agency_Agent - an autonomous entity capable of pursuing an objective through its autonomous decisions, actions and social relationships. It is capable of performing actions in the environment it lives; it can communicate directly with other agents, typically using an Agent Communication Language; it possesses resources of its own; it is capable of perceiving its environment; it has a (partial) representation of this environment in form of an instantiation of the domain ontology (knowledge); it can offer services; it can play several, different (and sometimes concurrent or mutually exclusive) agency_roles.

Each agent may be refined by adding knowledge items necessary to store/manage communication contents.

The Agency_agent statechart is:



Description of the Agency_Agent states:

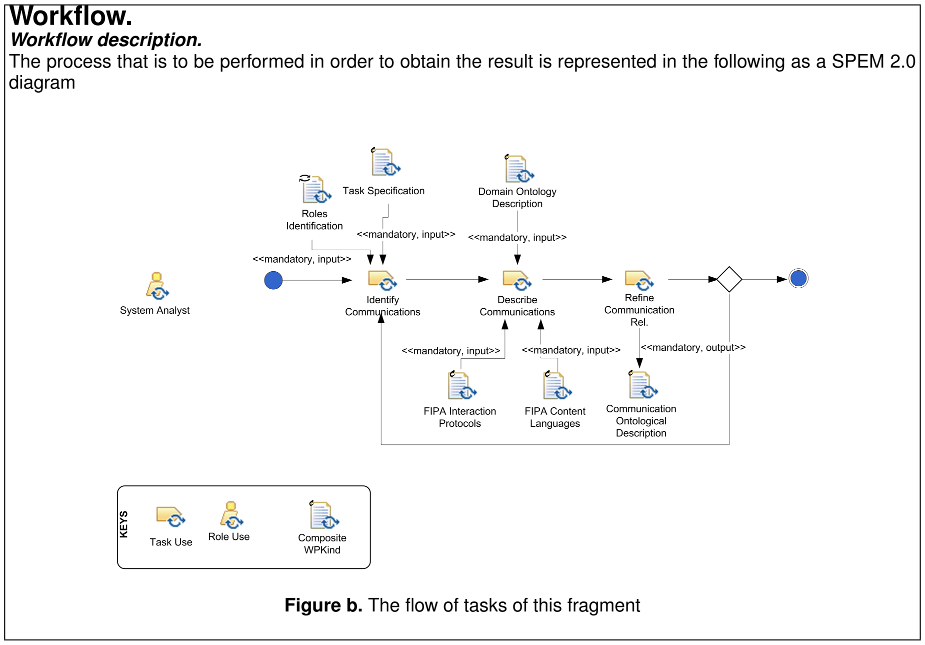
Defined: An Agency_Agent is in this state once it is instantiated in the system model. The agent's unique name has to be defined.

Refined: An Agency_Agent moves in this state once its knowledge chunks are defined.

Fig. 5. A Portion of the COD Fragment Document - The System Metamodel

6 Conclusions and Remarks

In this paper we presented the process fragment definition and the documentation template we use in our work. After a long experience done on the construction of design processes we realized that this template is an optimum starting point for the definition of a standard notion of process fragment. The presented document has been conceived with both a textual and a diagrammatic part in or-



Workflow.

Workflow description.

The process that is to be performed in order to obtain the result is represented in the following as a SPEM 2.0 diagram

Figure b. The flow of tasks of this fragment

Fig. 6. A Portion of the COD Fragment Document - The Workflow Description

der to provide different views on the fragment and in order to allow the designer to retrieve the most useful information for his own needs in a quick and also visual fashion. We created the document for being used for two purposes: reusing the fragment during the process creation in a (Situational) Method Engineering fashion and using it during design process enactment.

This work presents a fundamental improvement with respect to the work done some years ago and illustrated in [17], here the system metamodel was also considered as a component of the fragment but its importance has been now enriched by all the notions related to its constructs and how they can be defined. Moreover the fact that within PRoDe the System Metamodel is the central element for retrieving, selecting and assembling fragments have led to the need for its right and more fruitful representation in the fragment definition and documentation.

Another important outcome of our work is that since the fragment aims at designing a specific system metamodel construct, we can consider the fragment itself independent from the specific notation. The same result can be obtained by producing different work products in different notations. Such a feature is one of the strengths of the proposed fragment definition that is highly reusable and composable being mainly oriented to the metamodel construct it is aimed to define; for instance a fragment that delivers UML based work products can be easily composed to another fragment delivering free textual work product, it

Example.
In Figure c, the PurchaseManager agent starts a communication (see QueryForAdvice association class) with the PurchaseAdvisor agent. The communication contains the Course ontology, the Query protocol and the RDF language. This means that the PurchaseManager wants to perform a speech act based on the FIPA query protocol in order to ask the PurchaseAdvisor advice on how to purchase (supplier, number of stocks, number of items per each, purchase-money) provided the Course information.

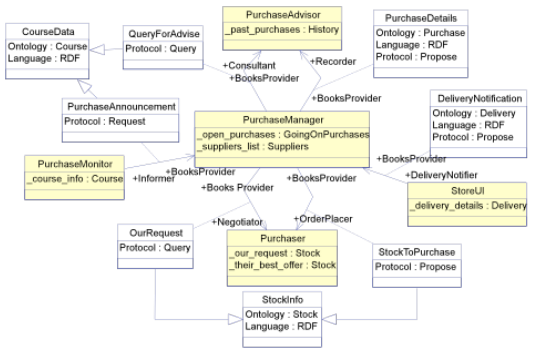


Figure c. An example of Communication Ontological Description diagram.

Fig. 7. A Portion of the COD Fragment Document - An Example of the Produced WP

is only important that the two have a matching set of input/output metamodel constructs. This fact overcomes the problem of interfaces among fragment and the problem, until now present, of having all fragments producing work products with the same notation; at worst we could create design processes where different parts have different notations but also this problem can be overcame by using a CAPE (Computer Aided Process Engineering) tool able to instantiate the right CASE tool for managing the enactment of the newly created design process. An example of such CAPE tool is Metameth, a prototype that we developed in the past in our laboratory [5].

Acknowledgment. This work has been partially supported by the EU project FP7-Humanobs.

References

1. S. Brinkkemper, M. Saeki, and F. Harmsen. Meta-modelling based assembly techniques for situational method engineering. *Information Systems, Vol. 24*, 24, 1999.
2. S. Brinkkemper, R.J. Welke, and K. Lyytinen. *Method Engineering: Principles of Method Construction and Tool Support.* Springer, 1996.
3. 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.

4. 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.
5. M. Cossentino, L. Sabatucci, and V. Seidita. A collaborative tool for designing and enacting design processes. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 715–721, New York, NY, USA, 2009. ACM.
6. 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.
7. D. Gupta and N. Prakash. Engineering Methods from Method Requirements Specifications. *Requirements Engineering*, 6(3):135–160, 2001.
8. AF Harmsen, S. Brinkkemper, and H. Oei. Situational method engineering for information system projects. In *Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8. 1 Working Conference CRISí94*, pages 169–194, 1994.
9. B. Henderson-Sellers. Method engineering: Theory and practice. In D. Karagiannis and editors Mayr, H. C., editors, *Information Systems Technology and its Applications.*, pages 13–23, 2006.
10. IEEE Foundation for Intelligent Physical Agents. *Design Process Documentation Template, Document number XC00097A-Experimental*, 2011.
11. 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.
12. I. Mirbel and J. Ralyté. Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11(1):58–78, 2006.
13. OMG. Object Management Group. Software & Software Process Engineering Metamodel. version 2.0. Document number: formal/2008-04-01. 2008, 2008.
14. J. Ralyté. 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*, pages 271–282, 2004.
15. M. Saeki. Software specification & design methods and method engineering. *International Journal of Software Engineering and Knowledge Engineering*, 1994.
16. Douglas C. Schmidt. Model-driven engineering. *Computer*, 39(2):25–31, Feb. 2006.
17. 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.
18. Valeria Seidita, Massimo Cossentino, and Salvatore Gaglio. Using and extending the spem specifications to represent agent oriented methodologies. In *AOSE*, pages 46–59, 2008.
19. 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.
20. ter Hofstede A.H.M. and Verhoef T.F. On the feasibility of situational method engineering. *Information Systems.*, 22(6/7):401–422, 1997.
21. UMLR-Revision-Taskforce. Omg uml specification v. 2.2. Object Management Group, 2009.
22. WfMC. The workflow management coalition. http:/www.wfmc.org., 2005.