A metamodelling-based approach for method fragment comparison

Massimo Cossentino[1], Salvatore Gaglio[1,2], Brian Henderson-Sellers[3] and Valeria Seidita[2]

[1] Istituto di Calcolo delle Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Palermo, Italy
cossentino@pa.icar.cnr.it
[2] Dipartimento di Ingegneria Informatica – University of Palermo, Italy
gaglio@unipa.it, seidita@csai.unipa.it
[3] Department of Software Engineering, Faculty of Information Technology University of Technology, Sydney, Australia
brian@it.uts.edu.au

Abstract. Several different approaches to Situational Method Engineering exist. They differ in terms of the primary element of the paradigm: the method fragment definition. Here, we introduce four method fragment definitions from the literature and compare their metamodels according to structural and functional criteria. The structural comparison showed a general alignment of some concepts that are sometimes referred with different names while the study of the compositional aspects results in evidence of substantial differences.

Keywords: Metamodelling; method engineering

1 Introduction

Method Engineering, and in particular Situational Method Engineering (SME) [1][12][13], is based on the assumption that one development process cannot fit all the existing problems and development contexts [4]. Rather, it allows the construction of a specific process to meet the requirements of each particular situation by reusing and assembling parts of existing methodologies (here used as a synonym for software development process, SDP) called Method Fragments [12]. Many researchers have adopted this approach for the construction of ad hoc solutions [1][6][16].

Applying the SME paradigm consists of executing the following phases: identification, storage in a Methodbase, selection and retrieval, and assembling of method fragments. In this process, a specific stakeholder, called the method engineer, is responsible for building the repository of fragments (methodbase) after having identified and extracted the pieces from existing methodologies [19] or having generated them from a metamodel [14]; after that he/she (or another method engineer) can select and assemble the proper fragments in order to create the new SDP.

The creation of the fragment repository, consisting of a number of adequately described fragments, is of fundamental importance in this process; thus in this paper we will consider SME aspects regarding the identification, description and representation of fragments in the repository. A method fragment may often be identified through a re-engineering process of existing methods possibly represented through a metamodel; the metamodel itself can be useful for identifying fragments from existing methods. In this context, a metalevel representation is very important. It is generally used to specify the concepts and relationships that define a SDP; since a method fragment is a part of a SDP, it can consequently be described by a metamodel.

Here we present three of the most representative fragment metamodels found in literature plus a proposal coming from a standardization organization in the agent-oriented context. For each, we describe the main constituent elements and how they are represented in the repository (section 2), then make a comparison (section 3) in order to highlight their commonalities and main differences. While common elements are used to sketch a simple model that can be seen as the common denominator of all of them, differences will be studied, in a future work, in order to verify whether they facilitate significant changes in the results of the following phases of SME (selection of fragments, retrieval from the repository and assembling of a new SDP).

Such comparative knowledge will then permit not only the likely convergence of these approaches, but also highlight various future research issues, such as a detailed study of the optimal granularity for method fragments.

2 Existing Fragments

A metamodel deals with all the different aspects of a method. Consequently, it has an important role to play in the analysis of method features; we use these considerations in this section to represent different method fragments from different authors: Brinkkemper and colleagues [1-3,13]; Rolland, Ralyté and colleagues [19-22]; the OPEN Process Framework work [6,14,15]; and the FIPA methodology group [17]. For each, we describe in the following four subsections the metamodel in terms of concepts it contains in relation to the definition of a fragment from each author as well as in terms of the elements indicating the fragment representation in the methodbase.

[Note that in each approach, we have retained the original terminology; indeed, comparison of such names provides one element of future ontological comparisons.]

2.1 Method Fragments (Brinkkemper et al.'s approach)

Method fragments[1][3] are coherent pieces of information systems development methods; there are two kinds of method fragments: the product fragment and the process fragment, the former concerns the structure of a process product, representing deliverables, diagrams, table, models and milestone documents and it can be composed of other product fragments. The latter kind of fragment models the

development process, describing the stages, activities and tasks to be performed to produce a product fragment. It can be composed of other process fragments and may have relationships with other process fragments. The metamodel used here to describe the method fragment is an ERA diagram where the terms concept, association and property are used in place of entity, relationship and attribute. Using this metamodel, both process and product fragments can be readily represented.

Brinkkemper et al. [2] proposed an approach to method fragment metamodelling based on three orthogonal dimensions: perspective, abstraction level and layer of granularity. The perspective dimension takes into account the product and process perspective on methods providing a view on the process (stages, activities and tasks) and the product features (deliverables etc.). The abstraction level dimension comprises the conceptual level, the technical level and the external level. In the conceptual level, a method fragment is considered to be a description of the process (or part of it); for instance, it can be the description of a specific phase or of a particular diagram. The technical level represents the executable part of a fragment; for example, specification of implementation, tools and repositories. The external level provides multiple views on the same method from different project roles (analyst, programmer and so on). One conceptual method fragment can be related to several external and technical ones in the sense that external method fragments are derived from conceptual method fragments, which are, in turn, supported by technical method fragments.

2.2 Method Chunks (Rolland, Ralyté and colleagues' approach)

Ralyté et al. [19][20][22] consider a method as composed of a collection of method fragments, although they prefer to call it *method chunk* in order to highlight the consistency and autonomy of this component. The method chunk integrates two aspects of the method fragment, the product and the process, so it represents a portion of process together with its related product(s). Guideline, situation and intention are the basic elements of method chunks. They are represented in Fig. 1 where the method chunk metamodel is represented using UML notation [22].

As above, we can consider this metamodel to be composed of two parts: the process model and the product model. As can be seen, a method is composed of chunks. Each chunk can be a simple chunk (atomic) or an aggregate of several; a fundamental relationship of the chunk concept is to the guideline concept, such that each chunk is represented by a guideline that the authors describe as 'the element that embodies the method knowledge to guide the engineer in achieving an intention in a given situation' [22]. A guideline is composed of an interface, describing the condition of chunk applicability (situation), and a body, representing the set of indications on how to proceed to achieve an objective (intention). The former represents the chunk input while the latter is the goal that the chunk aims to achieve. Both have a relationship with product, the situation (chunk input) being an aggregate of products that is the target of the intention.

The body of a guideline can be described graphically or informally by using three kinds of guideline: simple, tactical and strategic. A simple guideline describes in an

informal narrative form how to proceed to gain the target product. A tactical guideline, following the NATURE process modelling formalism, proposes a tree of context for producing a product. A strategic guideline gives a strategic view about which intention can be achieved following another. The strategic guideline is the most complex of the three guidelines. It is also called map, a map being a labelled directed graph where each node is an intention and each edge between two intentions represents the strategy to achieve that intention. Following a map, it is possible to dynamically construct a process model [21].

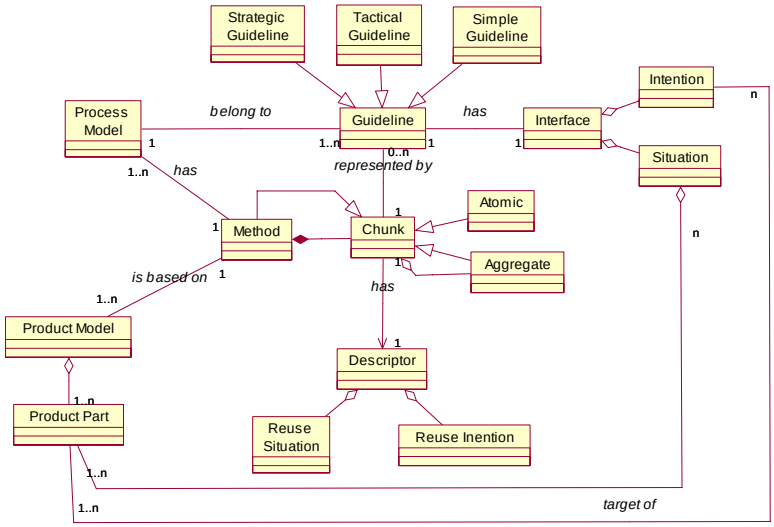

Fig. 1. The metamodel of the chunk (redrawn from [22])

The portion of the metamodel relating to the product aspect of the methodology is composed of three elements: Product Model, Product Part and Guidelines (shared with the Process Model portion). The cardinality of relationship between Method and Product Model indicates that for each method there is at least one product. In addition, in the metamodel the concept of guideline is related to the product part explicitly, meaning that a guideline is also useful for producing the product as well as suggesting the set of actions to perform in order to achieve an intention.

Another important element, in relationship to the guideline, is the descriptor, which outlines the situation in which the chunk can be reused; it conceptually extends the meaning of chunk interface, containing a set of attributes (ID, name, type, application domain, etc.) useful for selection and retrieval of the chunk from the repository.

2.3 OPF method fragments

The OPEN Process Framework [6][14] consists of a metamodel from which a large number of method fragments are generated and stored in a repository together with a

set of construction guidelines that are considered to be parts of existing methodologies used to construct new methodologies. The OPF metamodel is composed of five main metaclasses [10][15]: Stages, Producers, Work Units, Work Products and Languages (Fig. 2); when instantiated, each metaclass produces a method fragment.

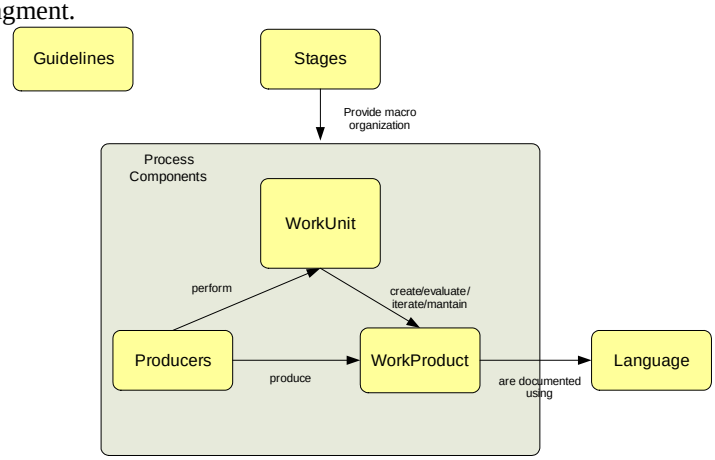

Fig. 2. OPF MetaModel (redrawn from [6])

Producers, Work Products and Work Units are the main metaclasses in the OPF; they are the main elements (process components) of a development process.

Producers are responsible for Work Products. They can perform actions such as creating, maintaining, iterating and evaluating on one or more Work Products; Producers in OPF are organizations, teams, persons, tools and roles.

Work Products are things produced during the development process by Producers performing a Work Unit. They are used as input for another Work Unit or delivered to clients. Work Products are, for instance: documents, models and diagrams.

Work Units are *functionally cohesive operations* performed by Producers during the development process. OPF distinguishes three main classes of Work Unit: Activity, Tasks and Technique. Both Activity and Task describe what is to be done in the development process but, whereas Activity represents a long term Work Unit of certain duration and is composed of a set of tasks, a Task gives more detailed information on the Work Unit being the smallest unit of work; Requirements engineering and requirements elicitation are respectively examples of Activity and Task in OPF. Technique represents the description of how a Task has to be performed; for instance, referring to requirement elicitation, use case modelling can be a technique to carry out this task.

Language is used to document a Work Product; for example, UML is the language used to model a use case or an object model, an implementation language (such as Java) specifies a code document and a natural language can be used for documents.

A Guideline helps method engineers both to instantiate the metamodel elements to create method components and to choose the best method components (from the method repository) in order to create the method itself; in addition, guidelines are provided to select work products, producers and work units and to provide guidance

on how to allocate tasks and associated techniques to producers and how to group the tasks into activities. Finally, stages (including phases and lifecycles), providing the organization to the process in terms of duration or point in time (phase, build, milestone and cycles), are chosen. This may be a Phase, a long stage occurring once during a process, or a Build, a short stage repeated during the process lifecycle, generally resulting in a system prototype.

The OPF metamodel, as currently being realigned with embryonic international standards, can be divided into the two main parts that a methodology must include: process and product. Process elements, as already described, are shown in Fig.3 together with their relationships while in Fig.4 Work Products elements are detailed.

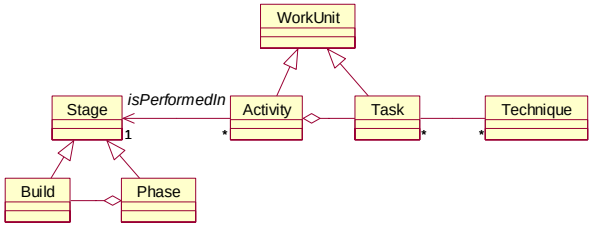

Fig. 3. WorkUnit (process) elements (redrawn from [11])

Fig.3 deals strictly with the process perspective – in summary, what is to be done, when and how, through Work Unit and Stage; a Work Unit (*what*) can be an Activity which is composed of several Tasks, the detailed execution of each being described by a Technique (*how*), each Activity being performed during a Stage (*when*) and classified depending on its duration and aim.

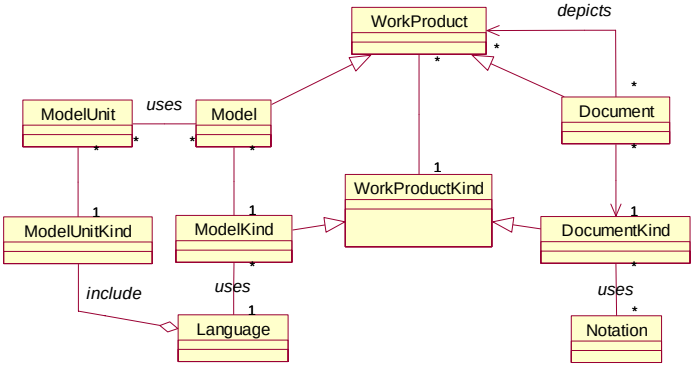

Fig. 4. Work Product elements (redrawn from [11]).

In Fig.4, the Work Product element is described in detail. It has two subtypes, Model and Document. The former is the conceptual entity representing the object of the development process, the latter is composed of texts, diagrams etc. Also, each document represents a work product (the many-to-many association). Each Model uses one or more ModelUnits that represent basic components of a Model, for example a Class is a kind of ModelUnit to represent static concepts. There are many

kinds of ModelUnit represented by the one–to-many relationship between ModelUnit and ModelUnit Kind, as well as between WorkProduct and WorkProductKind, Document and DocumentKind and Model and ModelKind.

Each DocumentKind uses a particular Notation and each ModelKind uses a Language that relates to groups of ModelUnitKinds concerning the same application context; Notation and Language are two aspects of modelling language, in that they share the same difference of syntax and semantic.

Instances of classes and sub-classes of these two metamodels are process components that can be stored in the repository. The OPF repository already contains a large number of components - there are about 30 predefined instances of Activity, 160 instances of Task, 200 instances of Techniques, and 76 instances of Role.

Applying the SME paradigm to the OPF repository consists of selecting appropriate process components from the repository and combining them to form an actual process within the methodology. This construction process depends on many factors relevant to the particular organization developing the new process, including CMM level of organization maturity, existing resources (people, tools, skills etc.).

2.4 FIPA method fragments

FIPA (Foundation for Intelligent Physical Agents) in late 2005 entered the IEEE Computer Society Standards Committee with the mission of promoting agent-based technology and the interoperability of agents with other technologies. It is actually going through a re-organization phase that involves all of its previous activities, working groups and technical committees (TC) in order to cope with the IEEE structure. One of the technical committees of the old structure (now in the re-structuring phase) was the Methodology TC whose scope was to define a proposal of standardization for agent-oriented design methodologies that adopted the SME paradigm. The definition of method fragment we report in this subsection is the result of that committee work and has been adopted by its members in several papers [5][7][8][9] although it is not (yet) part of the FIPA specification body.

The FIPA method fragment is based on the model of process description (the so-called process metamodel) shown in Fig. 5.

According to this metamodel (derived from the OMG Software Process Engineering metamodel, SPEM [18]), a process is composed of a set of activities performed by some active entities called Process Role whose task is to produce a well-defined state of an Artifact; each Process Role is responsible for one or more activities that produce/consume artifacts as output/input. According to this model, a process is strongly oriented to the production of products. However, although not intended nor likely to occur, it is possible for a role to be responsible for an artefact that is created as an output of an activity yet for the role to have no association to that activity[1].

Starting from the previous description, the FIPA Methodology TC defined a method fragment[17] as a reusable part of a design process composed of two elements: the structure of the product (the artifact resulting from the developer

[1] As pointed out by one of the anonymous reviewers.

activity) and the procedures necessary to construct the artifact, as illustrated by the metamodel shown in Fig. 6

In order to analyse this metamodel we divide it in three areas, the first concerning the description of the fragment in the sense of process (activities), process role and work product, the second illustrating the conditions for fragment reusability and the third describing the fragment as it is represented in the methodbase.

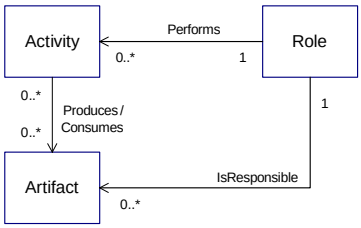

Fig.5. Process description metamodel

Considering the first area, a fragment can be depicted as being composed of a Process Description, that is the specification of what it is to be done and in what order. It aggregates activities describing a piece of work performed by one Role Actor (the performer of the described activity or even the assistant in the activity); activity has an input and an output of MAS Model element type and an activity product of Work Product type. A Work Product is everything produced, consumed or modified by a fragment (for instance, a Text Document or a Diagram like a UML diagram); a Work Product may be associated with a Role Actor responsible for its production. The FIPA Methodology TC focuses on the identification of a methodology for developing multi-agent systems so a fragment refers to a MAS metamodel and its aim is to refine/define MAS MetaModel elements that are the constituent part of Work Products.

The second area describes the elements required for fragment reusability and assembly; a fragment includes a glossary and a list of terms that facilitate the understanding of fragment concepts when applied to a context different from the one from which it was extracted. The aspect is useful to detect the field of fragment application, for instance a tool to be used to aid in the performance of an activity, a specific platform for system implementation etc.. An aspect has the form of a textual description. A method fragment has two kinds of guidance to indicate its own purpose; *guideline* relates to the fragment as a portion of a process i.e. a set of rules providing a detailed description on how to perform an activity, a *composition guideline* describes the context from which it is extracted, indicating the reuse possibility for the fragment.

The fragment dependency, in the third area, is the only element belonging to the methodbase; it is composed of a list of dependee and dependant fragments useful for composing different fragments. As noted above, the FIPA Methodology TC proposes the use of SPEM as modelling language for the description of the process aspect of fragments in the methodbase; in particular SPEM activity diagrams are used to describe the activities to be done in creating a specific product and the role that is involved and is used *de facto* to represent the core of FIPA fragment.

Several well known agent design methodologies (Adelfe, Gaia, PASSI, Tropos) have been described usingthe FIPA Methodology TC's approach and the fragments extracted from all of them can be found in the working page of the Methodology TC[2].

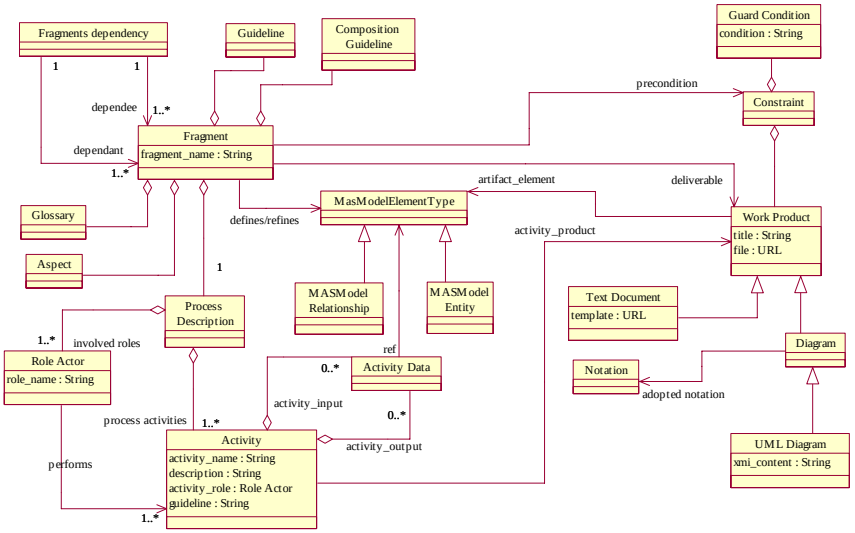

Fig. 6. The metamodel of the FIPA method fragment

3 Comparison

In this section, we present the result of a comparison between the different method fragments on the basis of both a structural and (less rigorously) a functional viewpoint. For the structural comparison, we examined the metamodel of the fragment highlighting differences among elements and underpinning concepts. In particular, we analyzed the kind of elements contained in each metamodel, whether different elements (with different names) have the same meaning or whether the same elements (with the same name in different metamodel) have a different meaning (Table 1).

A Process is essentially composed of four principal elements: *who* does *what*, *how* and *when*. These elements are reported in the rows of Table 1 while in the columns the different method fragments are represented. We explicitly consider two different aspects of fragment: the process and the product, consistent with the previous discussion. With regard to the process, there is also a row relating to the reuse support of the fragment itself, as offered by each specific approach.

As can easily be seen, while three metamodels specify the elements related to the work to be performed (sometimes at different levels of details, for instance: Activity, Tasks and Steps), only the method chunk metamodel does not present this element because in its metamodel the chunk element itself represents the work to be performed during the development process providing a detailed set of guidelines (the

[2] http://www.pa.icar.cnr.it/cossentino/FIPAmeth/metamodel.htm

row *How* in the table). Input and output to the activities are described in different ways: in the method chunk they are part of the chunk interface; in the FIPA fragment they are instances of the MAS metamodel (represented through work products); in the case of method fragment a work product is an input or output of each activity. Similarly, in the OPF, Work Products are seen as input/output to Work Units, notably Tasks.

Information on temporal distribution of the work is present only in the OPF metamodel (the row *When*) whereas in the method chunk and in the FIPA fragment this element is implicit in the description of the fragment itself in the repository in the form of a graph, in the first case, and of a SPEM activity diagram, in the second one.

From our analysis we can point out that all of the fragments presented in this paper are basically defined from the different authors in a similar way: they are meaningful constituents or parts of a (software development) process which we defined at the beginning of this section; the presence of a stakeholder element (*who* performs the work) in only two of the metamodels highlights that only the two referring fragments are really based on that process definition (OPF and FIPA fragment) while the other two are limited to modelling the process and product aspects.

Another important point that the table highlights is that all the metamodels contain one or more elements related to the possibility of reusing and assembling the fragments in order to create a new process. Sometimes they use different names and present different features; for instance in the method chunk we can find a Descriptor element, which possesses attributes like ID, name, application domain and provides a narrative description of chunk objectives; OPF and FIPA fragments respectively present Guideline and CompositionGuideline. In addition, in the FIPA metamodel there is a Glossary of terms and FragmentDependencies in the form of a SPEM dependency diagram, which can be logically associated to the “relationships with other fragments” of the method fragment metamodel.

With regard to Product aspects, the WorkProduct obviously is a central element in all the four metamodels some of which also specify the modelling language.

Finally, as an example of elements sharing the same name but with different meaning; Guideline is defined, in the method chunk and in the FIPA metamodels, as something guiding a stakeholder to perform its work during the development process while in the OPF metamodel this meaning is associated with the Technique element whereas Guideline refers to the way of composing the fragments.

The second kind of comparison we carried out (the functional one) started from the consideration that, in the situational method engineering context, a development process is constructed by assembling method fragments extracted from a methodbase in order to obtain the best process for a specific need/problem; thus the repository creation becomes an important phase that cannot be neglected; each fragment stored in the repository may be gleaned from existing development processes, through a decomposition and reengineering process, or may be created from scratch.

The fragment extraction or fragment creation phases are guided by the particular philosophy on concepts/elements one wants to represent through the fragment so we now compare the presented metamodels with the aim of discovering which of their features are essential to lead the engineer in the construction of method fragments; we called this process functional comparison referring to the possibility of specifying an

element that characterizes a particular functionality of the fragment capable of serving the purpose of creating a new process.

For instance, the Ralyté approach for the construction of a method chunk considers both the process and the product aspects in the decomposition of an existing process; in addition, she uses an ad hoc approach in creating new fragments focussing only on some specific intention and specific situation [19]; the method fragment [1][13] uses the same rationale - it can be inferred from the process construction rules presented in [3], which allow us to say that a method fragment depends on the process or on the product one wants to construct, that a product fragment should be produced by a corresponding process fragment and principally that, since the method fragments are hierarchically modelled and classified, the construction of a method at a particular level implies its successive assembly with another one of the same level.

Table 1. The structural comparison of metamodels

		Method Fragment	Method Chunk	OPF Fragment	FIPA Fragment
Process	*What*	Stage, Activity, Task and Steps		Activity, Task	Activity
	Input/ output		Situation (input), Intention (output)		ActivityData (input/output)
	How		Guideline, Interface	Technique	Guideline
	When			Stage, Build, Phase	
	Who			Producer (directly, undirectly)	RoleActor (Perform,Assist)
	Reusability and Assembly	Relationship with other fragments	Descriptor (ID, name, type, application domain)	Guideline	Glossary, Aspect, Composition-Guideline, Fragment-Dependencies
Product		Deliverables, Milestones Documents, models and diagrams	ProductPart Guideline	WorkProduct Model Document	WorkProduct MMMElements
				Modelling-Language (Notation, Language)	

In the FIPA Methodology TC proposal, method fragment construction is Work Product-oriented in the sense that a method engineer identifies the work product he/she wants to deliver and then he/she extracts the portion of process dealing with it. In this context, a method fragment is considered as being strictly linked to a product - it must deliver a product. However, referring to the previous definition of this fragment, an output is not necessarily a WorkProduct; the FIPA method fragment stands at a work product level of granularity.

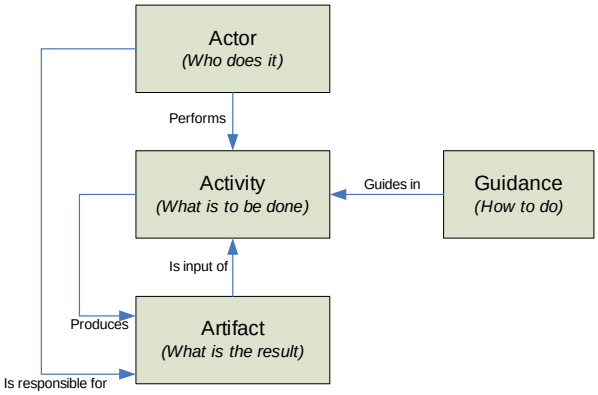

Fig. 7. General metamodel summarizing the fundamental elements of the approaches studied

OPF fragments are constructed as an instance of metamodel concepts so they are derived from existing processes or created from scratch considering one concept at a time; in so doing the fragments are constructed at a relatively low level of granularity, for example there are Task fragments, Technique fragments and Role fragments. There is no element of the metamodel that principally leads to the method fragment construction, they are created to populate a repository (= methodbase) from which they are taken out to construct a process that is primarily WorkUnit oriented, the WorkProduct being conceived as being created when a process is applied.

From these analyses, we can highlight that all the fragments share the same rationale based on the definition of fragments themselves. They are portion of processes underpinning a metamodel of concepts related to the principal elements of a development process, the work to be done (Activity, WorkUnit, etc), the delivered products, the stakeholders performing the work and the guidelines for the fragments reusability and assembly. All the fragments may be composed from other fragments.

The conclusion of this study is that a kind of higher level metamodel could be drawn that summarizes the different approaches and highlights their differences (when it is not possible to reconcile them). This would probably include (see Fig. 7) such general elements as the activity to be done, the guidance that can be applied, the stakeholder/process role involved in doing it and the resulting artefact. Pre- and post-conditions, composition features, notation languages and so on can be further added in specializing the model towards the different analysed proposals.

4 Conclusions and future work

In this paper, we have compared four different approaches to the definition of the fundamental brick of method engineering: the method fragment. Three of the definitions we considered come from well known authors in the field and the last one is from a standardization attempt that has been carried out within the FIPA (Foundation for Intelligent Physical Agents) organization, now part of the Standards Committee of the IEEE Computer Society.

After presenting the different definitions we compared them considering their structure and their 'functional' aspects (with this we mean the attitude presented by each contribution for a specific perspective in fragment extraction from an existing methodology and the following composition process).

The results are quite interesting since despite a general similarity (similar concepts are reported in almost all the definitions although sometimes with different names), we identified several differences in the details. As a final contribution, we drew a synthesis of all the approaches in the form of a simple model that unifies all the common denominators of the different method fragments and that could be referred as a central core for all of them.

These results provide knowledge and understanding of the various approaches currently utilized for the definition and description of method chunks/fragments. Without this knowledge, users of repositories based on the various fragment models will not appreciate the differences and therefore how to use the repositories effectively. This understanding also lays the groundwork for future merging of the repositories so that industry will have an agreed "standard" (metamodel, terminology and repository contents) thus facilitating interoperability of design support systems (CAME/CASE tools) in the future (a topic of significant current interest [23].

This work is also one step in a more extensive research plan in which we aim to compare the different fragments 'at work'; this will consist in a comparison of the actual results of an extraction process of fragments belonging to different definitions from a closed set of existing methodologies. The resulting methodbase will then be used with the aim of verifying whether fragments coming from different approaches can be intermixed in a new methodology or if the differences in their initial specification inhibit such interoperability.

References

1. Brinkkemper, S. Method engineering: engineering of information systems development methods and tools. Information and Software Technology**.** 38(7) (1996) 275-280
2. Brinkkemper, S., Saeki, M. and Harmsen, F. Assembly techniques for method engineering. Advanced Information Systems Engineering. Procs. 10th International Conference, CAiSE'98, Springer-Verlag, LNCS1413 (eds. B. Pernici and C. Thanos) (1998) 381-400
3. Brinkkemper, S., Saeki, M. and Harmsen F. Metamodelling based assembly techniques for situational method engineering. Information Systems, 24(3), (1999).209-228
4. Cockburn, A. Selecting a project's methodology, IEEE Software, 17(4) (2000) 64-71

5. Cossentino, M. and Seidita, V. Composition of a new process to meet agile needs using method engineering. Software Engineering for Large Multi-Agent Systems Vol. III. LNCS Series, vol.3390. Springer-Verlag GmbH (2005)
6. Firesmith, D. and Henderson-Sellers, B. The OPEN Process Framework — An Introduction. Addison-Wesley: Harlow, UK (2002)
7. Fortino, G., Garro, A. and Russo W. From modeling to simulation of multi-agent systems: an integrated approach and a case study. In G. Lindemann, J. Denzinger, I.J. Timm, R. Unland (eds.), Multiagent System Technologies, LNAI 3187, Springer-Verlag (2004) 213-227
8. Garro, A. and Palopoli, L. An XML multi-agent system for e-learning and skill management. In Ryszard Kowalczyk, Jorg P. Muller, Huaglory Tianfield, Rainer Unland, editors, Agent Technologies, Infrastructures, Tools, and Applications for E-Services , LNAI 2592, Springer-Verlag (2003) 283-294
9. Garro, A., Terracina, G. and Ursino, D. A multi-agent system for supporting the prediction of protein structures. Integrated Computer-Aided Engineering (ICAE), 11(3) IOS Press, Amsterdam, The Netherlands (2004), 259-280
10. Gonzalez-Perez, C., McBride, T. and Henderson-Sellers, B. A metamodel for assessable software development methodologies. Software Quality Journal, 13(2) (2005) 195-214
11. Gonzalez-Perez, C. and Henderson-Sellers, B. A powertype-based metamodelling framework. Software & System Modeling. 4(4) (2005) 1–19
12. Harmsen A.F., Situational Method Engineering. Moret Ernst & Young (1997)
13. Harmsen A.F., Brinkkemper, S. and Oei, H. Situational method engineering for information system projects. In Olle T.W. and A.A. Verrijn Stuart (Eds.), Methods and Associated Tools for the Information Systems Life Cycle, Proc. of the IFIP WG8.1 Working Conference CRIS'94, North-Holland, Amsterdam, (1994) 169-194
14. Henderson-Sellers, B. Process metamodelling and process construction: examples using the OPEN Process Framework (OPF). Ann. Softw. Eng. 14(1-4) (2002) 341-362
15. Henderson-Sellers, B., Serour, M., McBride, T., Gonzalez-Perez, C. and Dagher, L. Process construction and customization, Journal of Universal Computer Science, 10(3), online journal accessible at http://www.jucs.org (2004)
16. Kumar, K. and Welke, R.J. Methodology engineering: a proposal for situation specific methodology construction. In Challenges and Strategies for Research in Systems Development (eds. W.W. Cotterman and J.A. Senn), J. Wiley, Chichester (1992) 257-269
17. Method fragment definition. FIPA Document, http://www.fipa.org/activities/methodology.html, (Nov 2003).
18. OMG. Software Process Engineering Metamodel Specification, Version 1.0, Object Management Group, formal/02-11-14 (Nov 2002)
19. Ralyté, J. Towards situational methods for information systems development: engineering reusable method chunks, Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education Vilnius Gediminas Technical University, Vilnius, Lithuania (2004) 271-282
20. Ralyté, J. and Rolland, C. An assembly process model for method engineering, *Advanced Information Systems Engineering*), LNCS2068, Springer (2001) 267-283
21. Ralyté J. Reusing scenario based approaches in requirement engineering methods: CREWS method base. Proc. of the 10th Int. Workshop on Database and Expert Systems Applications (DEXA'99), 1st Int. REP'99 Workshop, Florence, Italy (1999)
22. Ralyté, J. and Rolland, C. An approach for method reengineering. 20th International Conference on Conceptual Modeling (ER2001) LNCS 2224, Springer (2001) 471-484
23. Konstantas, D., Bourrières, J.-P., Léonard, M. and Boudjlida, N. (eds.), Interoperability of Enterprise Software and Applications, Springer, London (2006) 466pp