A Repository of Fragments for Agent Systems Design

Valeria Seidita[1], Massimo Cossentino[2,3] and Salvatore Gaglio[1,2]

[1]Dipartimento di Ingegneria Informatica - University of Palermo
Viale delle Scienze 90128 Palermo, Italy
[2]Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche
Viale delle Scienze, 90128 Palermo, Italy
[3]SET - Université de Technologie, Belfort-Montbliard
90010 Belfort cedex, France
seidita@csai.unipa.it, cossentino@pa.icar.cnr.it, gaglio@unipa.it

Abstract

The creation of a new design process for a specific situation using the method engineering approach is based on the composition of a set of reusable method fragments. The request for these reusable method fragments leads to the need for a repository containing standardized fragments that can be easily selected and assembled in new design processes. In this work we present a definition of method fragment coming from the work of the FIPA Technical Committee Methodology and a repository where fragments are classified according to the specific process component (activity, process role, and work product) they underpin and on the specific MAS Metamodel element(s) they work on.

1 Introduction

Today a relevant number of design processes for developing multi agent systems can be found in literature; each of them is well suited for a specific purpose or for a specific agent architecture (BDI, reactive, state-based,...); one unique (and eventually standardized) design process fitting all possible situation does not exist; in the agent-based development context, we are now facing the same problem some researchers faced a few years ago when the definition of a new discipline was given, the Method Engineering.

Method Engineering aims at solving the previously said problem focusing on the creation of new techniques and tools allowing the construction of a specific design process (in literature referred as a situational method) [18]. Many researchers applied this paradigm and shared similar approaches: constructing and adapting new design processes by assembling (reusing) *method fragment* from a repository (called *method base*) built by splitting up some existing design processes [16][22][21][3][17].

Method engineer is the key stakeholder during a process construction activity; he develops two main phases, the first one regards extracting, defining, standardizing and storing in the repository the method fragments coming from existing processes while the second one consists in composing the new process through the selection and the assembly of the right fragments.

Our activity in this field started a few years ago within the FIPA Technical Committee (TC) Methodology from where the basis of this work arose. More specifically we acknowledge a great dependence of our method fragment definition with the one proposed in the FIPA context and also the design processes we studied are among the most important in that context (Adelfe, Gaia, PASSI, Tropos).

In this paper we introduce the repository we used for storing the fragments extracted from the above cited design processes. It is essentially a database where method fragments are stored in form of text documents and can be accessed using a categorization based on the process metamodel elements that we consider central in agent-systems design (process role, phase/activity, work product and MAS Metamodel element). It is interesting to note that, in our opinion, this latter element (the MAS Metamodel element, MMM element hereafter) is one of the keys of agent-system design today. The lack of a standardized or at least widely accepted MAS Metamodel brings to several different interpretations for it. Method engineers cannot neglect this aspect and we think that one of the first activities while building the new process is defining the MMM elements that he will instantiate and their relationships.

One of the most difficult activities in constructing our repository was its conceiving in such a way that fragments could be easily retrieved. We think that our solution to this problem, coherently with the choice of basing categorization on the four cited elements of the process metamodel is interesting: we built a taxonomy within each of the four basic categories (process role, phase/activity, work product, MMM element). In this way, a method engineer who aims at retrieving a fragment that produces a structural diagram (a kind of work product in our taxonomy) and involves a specific process role (like the Domain Analyst that is another item of our taxonomies), can easily find a list of all the fragments in the repository satisfying these criteria. A similarly interesting search could be related to the need for designing some kind of MMM element (suppose ontological concepts) because the method engineer wants to introduce a fragment about that in his/her new process.

The paper is organized as follows: in the next section we introduce our method fragment definition, in section three we describe the structure of our repository and in section four we provide an overview on the content of our method base; finally in section five, some conclusions are drawn.

2 Method Fragment

FIPA TC Methodology approach shares a similar meaning of method fragment with Harmsen and Brinkkemper [3][16][14]. A method fragment is a portion of a design process composed of two main parts, the process and the product. In our specific approach (also grounded on the process model proposed by an OMG specification, SPEM [20]) main process elements are Activity, Role and Artefact; more explicitly a development process is composed of Activities performed by one (or more) Role(s) responsible for producing artefacts, Activities produce or consume Artefacts as inputs or outputs. From now on, in order to be compliant with SPEM notation, we will refer to WorkProduct and ProcessRole in place of, respectively, Artefact and Role.

According to our approach a method fragment is composed as follows :

1. A portion of process (what is to be done, in what order), defined with a SPEM diagram.
2. One or more deliverables (WorkProducts like (A)UML/UML diagrams, text documents including code and so on). The result of the work could also be some kind of product/artefact that is not be delivered to anyone outside the development process. It also includes a reference to a recommended notation/language/ structure to be used.
3. Some preconditions (they are a kind of constraint because it is not possible to start the portion of process specified in the fragment without the required input data or without verifying the required guard condition).
4. A list of concepts (related to the MAS Metamodel) to be defined (designed) or refined during the specified process fragment.
5. Guideline(s) that illustrates how to apply the fragment and best practices related to that.
6. A glossary of terms used in the fragment (in order to avoid misunderstandings if the fragment is reused in a context that is different from the original one).
7. Composition guidelines are a description of the context/ problem that is behind the portion of methodology from which the specific fragment is extracted; it can be used to facilitate fragment reuse in the proper context.
8. Aspects of fragment are a textual description of specific issues like for instance: implementation platform for which the fragment is more suitable, application area, etc.
9. Dependency relationships that can be used to identify fragments strongly related to the considered one.

It can be represented by the metamodel shown in Fig. 1 where the presented elements are logically divided in three areas, the first one concerns the fundamental process features (Activity, ProcessRole, WorkProduct and MMMElement- drawn in grey in the figure), the second one concerns the reuse features of the fragment and the third one its representation in the repository.
As regards the first area, the main element is the Fragment that is part of a Development Process based on a well defined LifeCycle (for example waterfall[13], iterative/incremental[2]); Lifecycle, in the area concerning the reuse of the fragment, allows positioning the fragment in the proper place in the development cycle.

A fragment is composed of elements such as Activity, ProcessRole and WorkProduct useful for the description of the portion of process related to the specific fragment: an Activity describes a portion of work, performed by a ProcessRole [20] during which some data are used as input or produced as output, besides an Activity is an element of a Phase and it is composed of Steps, which are the smaller parts of work to be performed. At last an Activity can produce one or more WorkProducts, whatever consumed, produced, modified or refined during the portion of work considered in the fragment; a WorkProduct can be a diagram (for example an UML diagram) or a text document, and

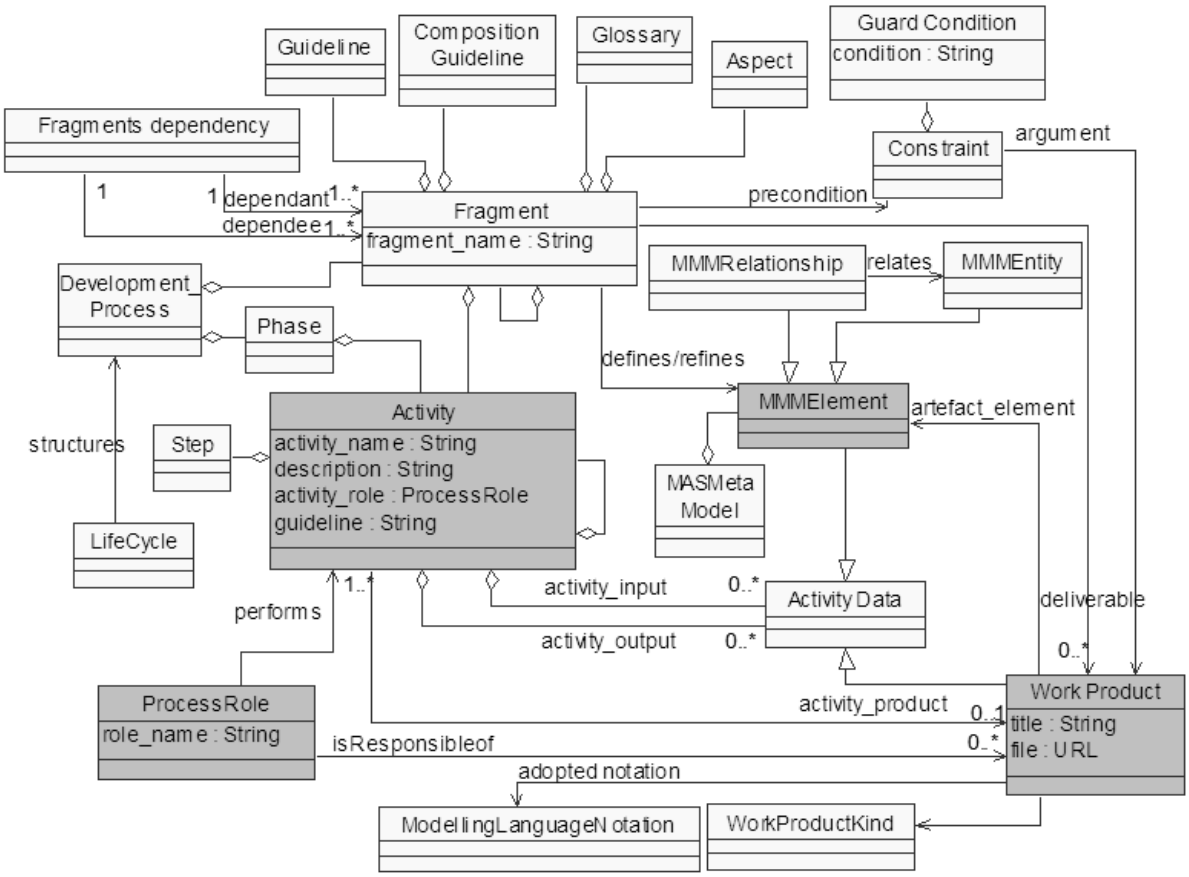

Figure 1. The method fragment metamodel

refers to a Modelling Language Notation. A WorkProduct is classified according to a WorkProductKind that describes the WorkProduct category, for instance text document, model, code library etc., it is built by a ProcessRole, which is responsible for its production. MMMElement (the Multi agent system Meta-Model Element) is a very important element of the fragment metamodel we adopted, it is the generic component of a MAS Metamodel (Multi Agent System metamodel) which constitutes the main building block of an agent design methodology. A MAS Metamodel is a structural/ontological representation of the elements (for instance agent, role, behaviour, ontology, ...) and relationships that will be instantiated in an actual system; instances of MAS Metamodel elements are described in WorkProducts and are also considered as input/output of activities; it is worth to note that in our approach the ultimate aim of the work performed in a fragment is to define/refine/relate MMMElements.

Two elements of the definition, belonging to this area, are not explicitly presented in the metamodel, but included in other elements, they are: portion of process, it is represented by the activities that are parts of the whole process, and list of concepts, which is composed of the set of MMMelements that are designed during the portion of process the fragment represents.

The second area (in Fig. 1) shows all the elements allowing the fragment reuse and assembly, they are the Glossary of terms used in the fragment, the Aspect, in the form of a textual description useful to identify the field of fragment application (several fragments are rather specific for some kind of agent architecture like the BDI one or an implementation platform like JADE and are not suitable for different contexts without a significant maintenance), and the Composition Guideline (while Aspects deal with the target system features, Composition Guidelines concern the development process and describe how and in which context the fragment can be profitably reused).

The third area describes the fragment as it is stored in the repository, Fragment Dependency is the only element it contains, this is a list of fragments that have a dependant or dependee relationship with the the specific reused fragment.

This definition of fragment is the basis for the extraction of method fragments from existing methodologies; in our opinion the extraction process aims at reifying the concepts of activity, role, work product and MAS Metamodel element. A fragment is physically stored in the repository in the form of a text document [7]; in these documents, SPEM is used as a process modeling language, in particular SPEM notation is employed to represent the main fragment elements (Activity, ProcessRole and WorkProduct) and some diagrams (for instance SPEM Activity Diagrams) are used to depict the work flow performed in the fragment and the relationships among these elements.

3 Fragment Repository

As we said in section 1 method engineering is a discipline which aim is to design, construct and adapt methods for information systems development [3][22][18]; one of the most common applications of the method engineering paradigm is an approach based on reuse, where the constituent parts of a development process (method fragments) are stored in a repository (method base) from which they could opportunely be selected and retrieved in order to be successively assembled in the new required process (this operation sometimes requires an adjustment of the fragment in order to properly place it in the new process).

In literature we found two repositories of fragments, the first one [15] is a method base associated to the Decamerone CAME tool and the second one is the OPF repository [11]. These repositories share the same aim: to facilitate the selection and retrieval of method fragments. In constructing our repository we share the same aim of the two previously cited examples but our approach is quite different.
The method base in Decamerone is based on the assumption that a method engineer creates a repository of fragments coming, only, from processes suitable for solving a particular problem; this brings about a first selection of design processes. The extraction of method fragments, that once stored in the repository may be selected and assembled in the new design process, regards only a limited number of fragments. Instead we aim at collecting a relevant number of the existing agent design processes and at storing all the fragments we can extract from them, in so doing, during the creation of a specific design process, we have a larger repository to be used and we can select and eventually adopt a fragment coming from a process that could not fit, in its wholeness, the problem we want to solve; we think this choice constitutes a richness for the selection activity. As regard OPF, which is the most important element of the OPEN approach [11], it comprises a metamodel representing all process elements which instantiation generates a method fragment; OPF contains a very large number of method fragments accessible from a website and it provides methods fragments at a very low level of granularity.
In our previous works [8][9][10][12] we extracted a lot of method fragments coming from different agent design processes created by different research groups and suited to deal with very different multi-agent system design philosophies, they are: Adelfe, Gaia, PASSI and Tropos [1][24][6][4].
All of these processes present substantial differences in the terms they adopted and in their meanings for specifying the design process elements; for instance in Adelfe process the process role called Requirement Analyst in some activities performs the same work performed by the System Analyst in the PASSI process; besides each process underpins a specific MAS Metamodel which elements have a meaning that can be different from the corresponding one in another process even if sometimes they share the same name. Therefore the fragments, once stored in the repository, lose importance and usefulness if we do not dispose a method for their easy retrieval. We thought that the rationale for storing the fragments in the repository (and then make clever query) was to consider the work performed by method engineer when trying to assemble new methodologies, he selects only the fragments he can really use; with this we mean that, for instance, if there is not any ontology designer among the workers of an organization, it is useless to include in the process under construction an activity to be performed by such a stakeholder (this would bring to a process that for its application would need skills that are unavailable). We think to facilitate the discrimination of the right fragments classifying them in categories based on the main process elements (2): Activity, ProcessRole, WorkProduct and MMMElement. However while categorizing the fragments we met a great problem: from the studied processes we collected about sixteen different process roles, seventeen phases (each of them is composed of several activities), a lot of work products and of MMM elements. Therefore we thought useful to categorize all the available method fragments according to a taxonomy unifying (and mapping) different elements (from different approaches) under a unique definition. We firstly identified the set of common activities, a design process is usually composed of, referring to the main phases of a software engineering design process [23][13], then the principal process roles performing these phases and finally a set of work product kinds; Fig. 2 shows the clustering rationale for phase and process role, it is just an illustrative representation of our taxonomy that does not exclude some possibility of intersection among different areas; for the work product kinds taxonomy we adopted the structure shown in Fig. 3. In the following subsections we will better illustrate the taxonomies.

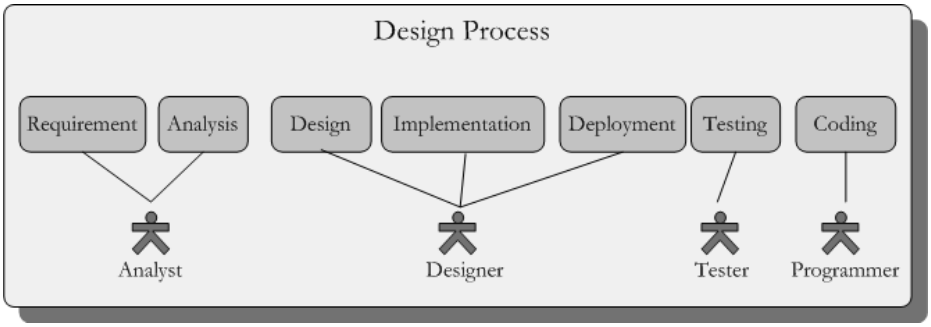

Figure 2. Categories for phases and process roles in the design process

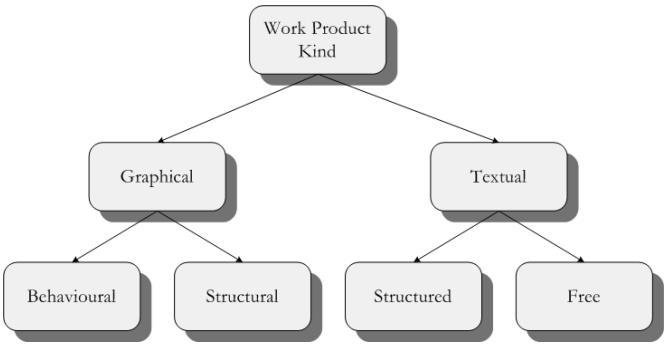

Figure 3. Categories for work product kind

3.1 Phase

Any kind of design process for information system production, and in our case for multi agent system production, can be decomposed in a set of activities (or phases) organized in sequential steps depending on the specific chosen process model; regardless of their organization some kind of activities have to be performed to develop any system. We examined all the phases our the 4 studied processes present (in terms of the work they carry on and their aim) and, referring to the main phases of a software engineering process, we clustered them in the following phases:

Requirements, it consists in the requirements elicitation phase during which a functional model is given to provide the purpose of the system and the interactions between the system and the environment.

Analysis, it consists in all the activities aiming at understanding the system and its structure (without reference to any implementation detail), identifying and defining the main entities of a MAS (such as role, communication, etc.).

Design, the aim of this phase is to define the agent architecture, describing agents' behaviours and to investigate how a society of agents cooperate to realise the system- level goals, and what is required of each individual agent in order to do this; all the aspects of the agent society are faced.

Implementation, gives a view on the system architecture, methods and classes are used to describe the agent's structure and behaviour.

Testing, it is composed of a unit test, the verify of the single agent's behaviour with regards to the original requirements of the system and a society test, the validation of efficient cooperation between agents.

Deployment, this phase defines and describes how agents are deployed and which constraints are present for their migration and mobility.

Coding, the phase of writing the code eventually with the aid of reusable code and source code.

Some of these phases are fundamental in a classic software development process while some others are specific for the agent oriented context, for instance design phase deals with the concept of agent and explores the social aspect of a multi agent system while a classic design phase concerns the way the different system components provide system functionalities.

3.2 Process Role

Starting from the phases identified in the previous subsection and from the examination of all the existing stakeholders in the referring agent design processes, we clustered, under the same element, a set of process roles performing similar activities. First of all we associated for each phase a process role, in Fig. 2 we can see that a general role called Analyst performs the requirement and analysis phases, the Designer performs design, implementation and deployment, a tester performs testing and the Programmer performs the coding phase, then examining all the process role involved in the the studied processes we succeeded in detailing each of these higher level stakeholder in the following:

System Analyst: models the current system and generates information about the future system, he is responsible of detailing use cases and he is an expert of the development

domain thus identifying and modeling the main elements of the multi agent system under construction (referring to the MMM elements).
Domain Analyst: analyzes the system environment in order to determine and model MAS domain elements.
User: defines and validates the system requirements.
Agent Analyst: analyzes the system to be in order to establish which entities can be agents and to research which architecture is necessary to build the system.
Agent Designer: analyzes and designs all the MMM elements strictly related to the concept of agent (in each design process), such as role, task, services, interaction language and so on.
User Interface Designer: identifies and defines the interface among actors and the system.
Programmer: is responsible of writing the code.
Test Designer: designs a test activity basing on system requirements an agent has to satisfy.
Test Developer: executes the designed tests.
Again we can see that some process roles are classic ones for object oriented context while some others, for instance agent designers, are specific stakeholders of the agent oriented context.

3.3 Work Product Kind

A generic work product produced by a process activity can be of a certain kind representing a specific category, for instance text document, code an so on; we clustered all the possible work product kinds under two main categories (Fig. 3): graphical and textual.

A work product which kind is graphical can be furthermore categorized as a structural or a behavioural one, when used to model respectively the static or the dynamic aspect of a system; for instance a behavioural work product points out the flow of messages along the time among different agents.

As regard the textual work product we decided to classify them as structured or free, in the sense that a text document can be hold by a particular template or grammar, for example to build a table or to write a code document, or can be freely written in a natural language.

Sometimes some work products can combine two different kinds (this is the case of a document including both a diagram and the related description) so we introduced the term atomic or composite to mean a work product of a single kind or a work product of two or more combined kinds.

3.4 MAS Model Elements

The MAS Metamodel gives a structural representation of the concepts belonging to the system under construction; in our previous works [7][8] we divided the metamodel of

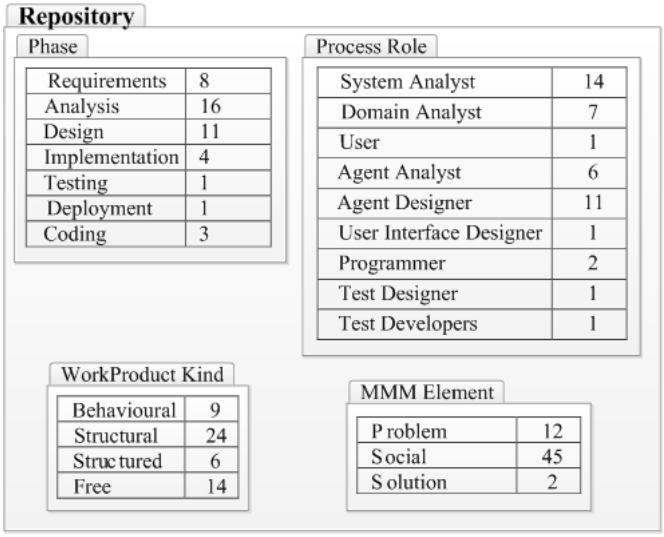

Figure 4. Repository Content

the multi agent system in three areas, to better deal with the different domain abstractions relevant to a system design; the first area represents all the aspects of the user's problem description including the environment representation, the second deals with agent based concepts that are useful to define a solution strategy and the third describes the structure of the code solution.

We claim that all the existing MAS Metamodels can be divided in the cited three areas thus allowing the creation of three categories of MMM elements; we named them: Problem, Social and Solution.

4 Overview on our repository content

Our work starts from a re-engineering activity of the analyzed processes [8][9][10][12], that let us to represent all of them in a standardized way using SPEM and to extract forty-five fragments, each of them stored in our repository on the basis of the process elements and the MMM elements categories it deals with; Fig. 4 summarizes how much fragments we stored in each category.

Some repositories of fragments already exist [11][15], in our work we propose a repository structured with the aim of minimizing the effort necessary for finding the best fragment for a specific purpose and a specific application context.

In building our repository we adopted an approach that is someway similar to the OPF one [11](we reengineered the studied processes, expressed them in a standardized notation, and then extracted the fragments); in so doing we adopted a specific choice at the basis of the extraction process: we looked at the work products as the beam for split-

ting down the process in its constituting method fragments. The result is a relatively small number of method fragments but we think this could be the right level of granularity for our purposes because, in this way, the process construction can be lead by a concrete and tangible entity, the work product, and finally it results in an easier and faster extraction and selection of fragments.
We already made an experiment on constructing a new design process using some method fragments stored in the repository; it consisted in building an agile process for rapid prototyping of applications in our laboratory. The result was the Agile PASSI [5] process that was largely based on PASSI fragments, because we wanted to reuse the expertise we accumulated in several years of using it.

The repository we presented can be accessed from a website [1] that allows the user to query the method base looking for matches on several keys from the categories we proposed in section 3; in the repository the method fragments are stored in form of text documents and the metadata are managed using a relational data structure, some relationships are, for instance, the following: *Fragment(ID,FragName,FileName,ID_Phase)* contains the data identifying each fragment with the link to its representing document and the phase it belongs to, *Phase(ID,PAName,PADescription)* contains the information on phases and in the same way for all the other elements.

5 Conclusion

In this paper we presented a repository of method fragments that can be used for composing a new design process for multi-agent systems. The peculiarities of this repository can be summarized as follows: (i) this is a specifically agent-oriented conceived repository; as a consequence a specific attention has been given to agent-oriented peculiarities like the MAS Metamodel elements that are explicitly present in both the method fragments descriptions and in their categorization; (ii) fragments have been defined (and extracted from existing methodologies) following a work product-based approach; we mean that each method fragment is supposed to produce at least one work product (not necessarily from scratch, it can also refine an existing one); (iii) we adopted a specific philosophy for enabling fragments retrieval from the method base: fragments are categorized according to 4 basic criteria: process roles involved in the design activities, phase of the overall design process in which the specific fragment can be reused, kind of work product produced by the repository and finally, MAS metamodel elements that are managed in the activities involved in the fragment. In the future we plan of extending the repository by including some other methodologies (we are currently working on Ingenias and Prometheus) and then incorporating that in a CAME/CASE tool we are building in order to effectively support the work of the process designer first (while he defines the new process) and the system designer later (while he designs the agent system).

[1] http://www.pa.icar.cnr.it/passi/fragment.html

References

[1] Bergenti, F., Gleizes, M.P., Zambonelli, F.: Methodologies and Software Engineer- ing for Agent Systems. Kluwer (2004)

[2] Boehm, B.: A Spiral Model of Software Development and Enhancement. IEEE Computer, Vol. 21, N 5, May, (1988) pp. 61-72

[3] Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Information and Software Technology. 38(7): p. 275-280 (1996)

[4] Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the tropos project. Inf. Syst. 27 (2002) 365389

[5] Chella, A. Cossentino, M., Sabatucci, L., Seidita, V.: Agile PASSI: An Agile Process for Designing Agents. International Journal of Computer Systems Science & Engineering. Special issue on "Software Engineering for Multi-Agent Systems". May 2006. in printing

[6] Cossentino, M.: From requirements to code with the PASSI methodology. In Henderson-Sellers, B., Giorgini, P., eds.: Agent-Oriented Methodologies, Idea Group Inc. (2005)

[7] Cossentino, M., Sabatucci, L., Seidita, V.: Method Fragments from the PASSI process. Technical Report ICAR-CNR n. 21-03 (2003)

[8] Cossentino, M., Sabatucci, L., Seidita, V.: SPEM description of the PASSI process. Technical Report ICAR-CNR n. 20-03 (2003) Available on line at http://www.pa.icar.cnr.it/cossentino/FIPAmeth/metamodel.htm.

[9] Cossentino, M., Seidita, V.: SPEM Description of ADELFE Process. Technical Report ICAR-CNR n.05-07 (2005)

[10] Cossentino, M., Seidita, V.: Tropos: Processo e frammenti. Technical Report ICAR-CNR n.05-06 (2005)

[11] Firesmith, D. and Henderson-Sellers, B.:The OPEN Process Framework - An Introduction. Addison-Wesley: Harlow, UK (2002)

[12] Garro, A., Turci, P.: Gaia Fragments. available on line at http://www.pa.icar.cnr.it/cossentino/FIPAmeth/metamodel.htm

[13] Ghezzi, C., Jazayeri, M., and Mandrioli, D.: Fundamentals of Software Engineering. Prentice Hall International, Upper Saddle River, NJ (USA) (1991)

[14] Harmsen A.F.: Situational Method Engineering. Moret Ernst & Young (1997)

[15] Harmsen,A.F., Brinkkemper, S.: Design and Implementation of a Method Base Management System for a Situational CASE Environment. APSEC 1995: 430-438

[16] Harmsen A.F., Brinkkemper, S., 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, pp. 169-194, North-Holland, Amsterdam, (1994)

[17] Henderson-Sellers, B.: Process Metamodelling and Process Construction: Examples Using the OPEN Process Framework (OPF). Ann. Softw. Eng. 14, 1-4, 341-362 (Dec. 2002)

[18] Kumar K., Welke R.: Methodology engineering: a proposal for situation-specific methodology construction. In Challenges and Strategies for Research in Systems Development, pages 257269, 1992.

[19] Method fragment definition. FIPA Document, http://www.fipa.org/activities/methodology.html, (Nov 2003)

[20] OMG, 2002, Software Process Engineering Metamodel Specification, Version 1.0, Object Management Group, formal/02-11-14 (Nov 2002)

[21] 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, 271-282 (2004)

[22] Saeki, M.: Software Specification & Design Methods and Method Engineering. International Journal of Software Engineering and Knowledge Engineering.

[23] Sommerville, I.: Software Engineering. Addison-Wesley (2004)

[24] Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: the gaia methodology. ACM Transactions on Software Engineering and Methodology 12 (2003) 417470