Methodology Fragments Definition in SPEM for Designing Adaptive Methodology : a First Step

Sylvain Rougemaille, Frédéric Migeon, Thierry Millan, Marie-Pierre Gleizes

Methodology Fragments Definition in SPEM for Designing Adaptive Methodology: A First Step

Sylvain Rougemaille[1], Frederic Migeon[1], Thierry Millan[2], and Marie-Pierre Gleizes[1]

[1] SMAC team,
[2] MACAO team,
IRIT Computer Science Research Institut of Toulouse,
Université Paul Sabatier,
118 Route de Narbonne,
F-31062 TOULOUSE CEDEX 9, France
{rougemai,migeon,Thierry.Milan,Marie-Pierre.Gleizes}@irit.fr
http://www.irit.fr

Abstract. The aim of this paper is to highlight how SPEM (Software and System Process Engineering Meta-model) 2.0 OMG (Object Management Group) can participate to design adaptive methodology process. The idea follows the FIPA Methodology Technical Committee (TC) one which consists in expressing a methodology in several fragments. Then, designer has to combine the relevant fragments to compose his own methodology. In this paper, we have chosen SPEM 2.0 OMG to express the fragments. The latest SPEM version improves methodology content and process re-usability, by introducing new capabilities as a clear separation between structural and dynamic methodology concerns. Those improvements in the field of methodology specification, are studied to determine their interests in the scope of Agent-Oriented Software Engineering (AOSE) and particularly, their impact on "methodology fragments" definition. ADELFE and PASSI methodologies have been taken as example to illustrate the use of SPEM 2.0 in the scope of "fragment" definition. In this paper, only the first step of the general objective consisting in expressing the fragments, is done and presented.

Keywords: SPEM 2.0, ADELFE, Methodology Fragments, Agent Oriented Software Engineering, Process Engineering.

1 Introduction

Many agent-oriented methodologies have been developed last decade (e.g. [1,12]: ASPECS [7], ADELFE [2], Gaia [22], INGENIAS [9], PASSI [6], Prometheus [18], SODA [17], Tropos [4]). Each has its own specificities: ADELFE is dedicated to adaptive system and cooperative agents design, ASPECS is dedicated to holonic multi-agent systems, Gaia focuses on static organization and roles, whereas

PASSI focuses on agent social aspects thanks to ontology, SODA highlights the notion of environment, etc. However, Agent-Oriented Software Engineering (AOSE) research community agrees the necessity of several methodologies and advocates the impossibility to build one general and universal one as it was pointed out in works like [10]. Thus, we cannot assume these methodologies can be applied to build every multi-agent applications. It seems that some methodologies or some parts of methodologies are more relevant than other to achieve some kinds of task. For example Tropos treats very well the preliminary requirements and provides models to realize it. That is the reason why the FIPA Methodology TC has proposed to define fragments. A fragment represents a portion of a methodology. Then, as it is also explained in [11], designers can choose methodological components from different methodologies in order to build their own relevant one. Software and System Process Engineering Metamodel (SPEM) is a standard specified by the Object Management Group (OMG) which latest revision 2.0 has just been adopted [16]. Its scope is the definition of a minimal set of concepts to design and manage software and system development process. In this paper, our aim is to highlight the relevance of this meta-model to express FIPA fragments.

Section 2 briefly describes the ADELFE methodology process which is used to illustrate the concepts developed in the paper. Section 3 presents the SPEM 2.0 OMG freshly adopted standard and studies more specifically new concepts such as *Method Plugin*, that are promising in the scope of AOSE. We also argue that this latest OMG vision of software process modeling is compliant to the FIPA Methodology TC concepts of "Methodology fragment" [8] (see section 5). Section 4 focuses on the aspects of SPEM 2.0 which are interesting for AOSE. Section 5 defines the fragment notion and the translation between the fragment and SPEM 2.0; this is illustrated on ADELFE and PASSI in section 6. The papers ends with the analysis and some perspectives of this work.

2 Introducing ADELFE Methodology

In order to illustrate the use of SPEM 2.0 [16] in AOSE, the ADELFE methodology [2] which was described with the previous SPEM version, has been taken as an example throughout the paper. ADELFE is devoted to the development of softwares with emergent functionalities and conforming to the Adaptive Multi-Agent System (AMAS) theory [3]. It is based on the Rational Unified Process (RUP) which was modified to fit specific AMAS needs. We recently have migrated this definition in SPEM 2.0 thanks to the Eclipse Process Framework (EPF)[1]. The following sections are illustrated with this definition conforming to the SPEM 2.0 recommended notations (SPEM 2.0 profile). ADELFE consists of five phases:

- *Preliminary* and *Final requirements*: they represent typical phases in object-oriented software development and are based on the RUP. However,

[1] `http://www.eclipse.org/epf`

in addition to the classical approach, they define AMAS specific tasks such as precise study of the system environment.
- *Analysis*: this is a specific phase that allows analysts to determine whether an AMAS is needed or not.
- *Design*: this phase is devoted to the determination of software architecture. It strongly depends on the previous ones (object or AMAS specific design).
- *Development*: this is a model-driven phase which allows automatic code generation of the previously designed agents. This phase is still under development and was presented in [19].

This paper focuses on the analysis phase. It is composed of four tasks devoted to the definition of a primary software architecture from the requirements that have been previously established (see figure 1). It determines the adequacy between the problem domain and an AMAS solution. This is done thanks to the "Verify AMAS Adequacy" task that allows the "Agent Analyst" to elect an AMAS approach, if needed, by answering questions about the systems functionalities.

To ease the analyst task, a visual tool which provides the adequacy degree at local and global level, has been developed. This verification task is specific to the ADELFE methodology, thus it has been defined as a fragment, and is a part of section 6 examples.

3 SPEM 2.0 Overview

SPEM 2.0 is a MOF 2.0 [13] based meta-model that defines extension to the UML 2.0 infrastructure [14] meta-model. A notation has also been defined, which is based on a UML 2.0 superstructure [15] profile. The following use this notation to illustrate the underlying meta-model. Roughly, the SPEM 2.0 meta-model consists of seven packages:

- The *Core* package defines SPEM 2.0 foundations: its main concepts and the way they can be extended (based on UML 2.0 infrastructure)
- The *Process Structure* package contains the basics elements to describe a development process as a breakdown structure.
- The *Process Behavior* package contains concepts enabling the description of process execution (*State,Transition, ControlFlow*, etc.). It does not define a specific formalism but links process structure and external behavioral concepts letting implementers feel free to select the appropriate language (UML activity or state machine diagrams, BPMN[2], etc.)
- The *Managed Content* package defines concepts for textual descriptions of processes and methodology contents.
- The *Method Content* package defines the core concepts of every methodology such as *Tasks*, *Roles* and *Work products*.
- The *Process With Method* package binds method contents (i.e. what have to be done) to process structure elements (i.e. the scheduling of these tasks).

[2] Business Process Modeling Notation `http://www.bpmn.org/`

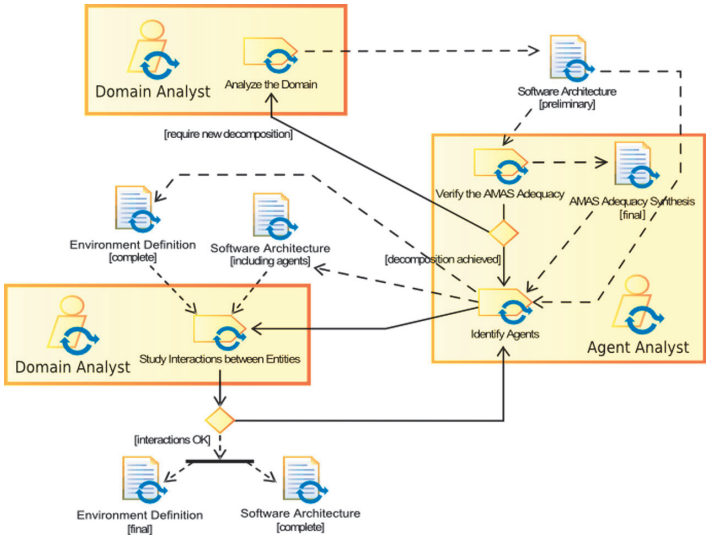

Fig. 1. ADELFE analysis phase workflow

– The *Method Plugin* package defines large scale plugin mechanisms to ease the definition of separate extensible methodology. It provides the concept of *Method Library* which is the container for *Method Plugin* and *Configuration* (those two concepts and their interests are discussed in further sections).

As a process engineering meta-model, SPEM 2.0 allows to model, document, present, manage, interchange development processes and methods. *Phases* can be described from a static or structural point of view, in terms of *Method Content* concepts, or from a more dynamic point of view, using *Process with Methods* concepts. Figure 1 is an example of a workflow (UML activity diagram) describing the tasks sequence and the products involved in the ADELFE analysis phase.

4 SPEM 2.0 Capabilities

SPEM 2.0 structure has been improved and some of its concepts have been clarified, for instance, a clear separation have been defined between *Method Content Element* and *process Element*. Those improvements bring capabilities that we found interesting in the scope of agent oriented methodology fragments definition and use.

4.1 Concerns Separation

SPEM 2.0 provides clear separation between the definition of methodology contents *Tasks, Work Products* and *Roles* and their application in a specific delivery process (Activities, Role Use, Work Product use). In fact, those two concerns

are respectively managed by the *Method Content* and *Process Structure* SPEM packages. Thus, process description is separated from the structure, it could be defined on its own (*Process Behavior* package) as, for example, an extension of UML Superstructure behavioral concepts (*Activity* and *State Chart* diagrams specialization). This separation is enforced by the definition of different *PackageableElements*: on one hand *MethodContentPackage* and on the other *ProcessPackage* which represents two distinct "containers". The first one contains the building bricks of methodologies (*Task, WorkProduct, Guidelines,...*), the second defines their use through *BreakdownElements* such as *Activity, TaskUse, RoleUse, WorkProductUse* and so on.

4.2 Modularity and Re-usability

At the Method level. The *Method Plugin* package authorizes modularization, eases extensibility and thus ensures that every methodology could be tailored in the more suitable way. SPEM 2.0 allows the description of *Method Plugin* which can be considered as a kind of extensible method packages container; it defines a whole method (content and use). The idea can be summarize as follow, a *Method Library* defines both *Method Plugins* and *Method Configurations* that represent respectively the content of the library and visibility "rules" over this content; in other words, building bricks that method engineers can select (the set of methods contained by the library), and what is effectively presented to the end-user (parts of those methods covering specific needs of the process). *Method Plugin* defines a first level of modularity. In addition, *Method Library* concept represents the topmost container for plugins and configurations, this last notion constitutes a kind of repository.

At the Process Level. *ProcessPattern* is a special kind of activity defined in the SPEM 2.0 base plugin, such as *Iteration, Phase,* etc. It refers to an un-applied general process, i.e. a breakdown of elements unbound to *Method Content* elements. It could be used as a building brick for development life cycle, the rationale for this concept is to describe reusable cluster of activities known to achieve the production of some deliverables (sets of Work products) in an efficient way.

Within a Process. Considering a fine-grain level of modularity and encapsulation, SPEM 2.0 offers the concept of *Process Component*, which conforms to the notion of software component [21] (provided and required interfaces, white or black box vision, ports, assembly, etc.). Taking a closer look to the adopted specification[16], it is a specialization of *ProcessPackage*, which is the primary process element container (*Activities, Task Use, Work Product Use*, etc.). It defines a process within an activity as a "black box" simply qualified by *Work Product Ports* that could gather its inputs and outputs (as could be software components interfaces). The main goal of process component is to keep some part of a process unresolved, i.e. the development phase, in order to choose the better implementation, using a code-centric approach or a model-driven one for example.

5 AOSE Methodology Fragments

This section presents the notion of Methodology Fragment in the scope of AOSE. It is promoted by the FIPA Methodology TC [8] and shares principles with *Situational Method Engineering* as it was quoted in [20]. FIPA method fragments are tightly connected to the SPEM 1.0 OMG standard as they use the same main concepts: *Activities, Artifacts, Role* and so on.

5.1 Definition

This section presents the method fragment as it has been defined by the FIPA methodology TC, an much more extensive definition of this proposition can be found in [5]. According to the FIPA, a fragment is composed of the following parts:

1. A portion of process.
2. The result of the work. It represents some kind of product/artifact like (A)UML/UML diagrams, text documents, etc.
3. Some preconditions, kind of constraints specifying when it is possible to start the process specified in the fragment because of missing required inputs or because of guard condition violation.
4. A list of concepts (related to the MAS meta-model) 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 (to avoid misunderstandings and to ease re-usability).
7. Composition guidelines - A description of the context/problem that is behind the methodology from which the specific fragment is extracted.
8. Aspects of fragment. Textual description of specific issues like for example: platform to be used, application area, etc.
9. Dependency relationships useful to define fragments assembly.

It should be noticed that not all of these parts are always mandatory. However, from this definition, it is already possible to note some interesting points. First of all, fragments use concepts that are still defined in the SPEM 2.0 specification (*Activities, Work Products, Roles*) but it needs clarification, because concepts have been moved into separate packages and are associated thanks to new elements such as *WorkProductUse* and *RoleUse*.

5.2 Fragment Compliance with SPEM 2.0

Rationale. SPEM possesses a wide audience and its use is enabled by many implementations. We advocate that being compliant with this “de facto” standard is important to broaden the use of agent-oriented methodologies and principles. Furthermore, as fragments aim to provide a common framework (language, repository) for agent-oriented methodologies and that the latest SPEM version partially meets these requirements we propose a translation, mapping between FIPA fragments and SPEM.

Mapping Fragments to SPEM 2.0. A method fragment is a portion of a development process defining deliverables (work products), guidelines, preconditions, and sets of concepts and key-words characterizing it in the scope of the method in which it was defined. Considering fragments through this rough definition, eases the determination of the more suitable SPEM 2.0 concepts, if any. However, one of the proposals of [5], which presents an enhanced version of the fragment meta-model from the FIPA methodology TC, is that fragments should be considered from different points of view whether you are interested in their reuse, storing, implementation or in the definition of the process they represent. So it seems obvious that the election of a method fragment depend on the point of view and the requirements defined by the method engineer. By taking a closer look to the SPEM 2.0 specification, *Method plugin* package provides concepts that fulfill most of FIPA requirements:

- *Method Plugin* defines by the means of *Method content packages* and *Process packages* sets of reusable process portions (see section 5.1). It goes further than this by splitting those reusable parts into methodology contents and process.
- FIPA vision of fragment corresponds to a process portion. Of course, this portion must embody some composition rules or constraints. FIPA elected *Process Component* as the SPEM 1.0 closer concept for fragment [5]. This notion has been improved and clarify in the latest specification so that it seems it better fits fragment needs. In fact, it defines precise input and output constraints in terms of *Work Product Ports*. A process component encapsulates a single Activity which can be broken down into sub-activities representing the process leading to be delivered of output work product using the declared inputs.
- Referring to the above description, fragment is equipped with glossary, guidelines and other textual descriptions or concepts intended to ease method engineer to achieve its composition. All those elements can be seen as *Guidance* special kinds. In the *Core Package Guidance* is an *Extensible Element* specialization, therefore it can be extended and defines its own *Kinds*. *Glossary, Aspect, Guideline* and *Composition Guideline* can be derived from *Guidance* and applied to any elements defined in the *Process Component*.
- In [20] authors promote the idea of a common repository for method fragments indexed thanks to concepts defined by MAS meta-models, they also have implemented it. SPEM integrates the concept of repository: *Method Library* which is the container for *Method Plugins*. A library contains several plugins, both *Method Content* and *Process Package* elements of these plugins can be referred by *Method Configuration* to tailor and to present a new Software Engineering Process (SEP).
- Dependency relationship are defined at different levels: between plugins, between process component as well as mapping from process pattern and the methodology contents they use. In fact, it only depends on fragment granularity.

Granularity. Fragment does not match a single SPEM 2.0 concept but should be considered as different ones depending, for instance, on its granularity. As SPEM offers different re-usable concepts with different granularity, from *task* to *Method Plugin*.

Concerns. Fragment is a portion of process. However, process element in SPEM 2.0 has been divided into definition and use. Thus, this separation needs to be considered while mapping fragment to SPEM concepts. Fragments that define elementary work part, related products and roles should be mapped to *Method Content* concepts. Whereas fragments defining some "good practice" or a common way to deliver a kind of products (i.e. an Model Driven approach to produce code), should be mapped to *Process Content* concepts.

Custom Categories. Method plugin allows the re-usability of its whole contents definitions and process uses. In order to ease the reuse of fragments, we propose to integrate their definition into two *Method Plugin* customs categories, *Fragments* and *Fragments Guidelines*. Those two categories are intended to group, on one hand, the work to do and the way it can be done (*Tasks* and *Process Patterns*) and, on the other hand, guidances (concepts and all useful documents for the use of fragments). Those categories could be themselves categorized thanks to inner custom categories (see 6.2).

Going Further. *Method Library* can be used as general purpose SEP container. It could even contain sets of reusable elements (content and process) that do not belong to a specific SEP, but could be used as building bricks. More than an agent-oriented methodologies repository, SPEM 2.0 *Method plugin* and *Library* allow to reuse any other method plugin elements since they have been imported in the same library. The EPF eclipse plugin is a SPEM 2.0 implementation which provides all those capabilities, thus it can be used to define method plugins into library as well as tailor new SEP from those plugins (*Method Configuration*). Projects such as OpenUP[3] meet the requirements of the FIPA methodology TC by using the modularity and re-usability skills of SPEM through the EPF plugin. In fact, OpenUP provides an open-source, common and extensible base for iterative incremental SEP. It seems obvious that AOSE will make profit of such projects by defining specific AO method plugins and reusing predefined ones.

6 SPEM 2.0 Fragments Definition

This section presents a kind of fragment definition "process", that is illustrated with ADELFE, and PASSI fragment. First of all, method plugins have to be defined. This allows us to describe re-usable method and process contents (*Tasks, Work Products, Roles* and *Guidance*, etc.). Then, some customization is needed. New plugins are equipped with two *Custom Categories*: *Fragments* and *Fragments Guidelines* which will gather all elements that are needed for the

[3] `http://epf.eclipse.org/wikis/openup`

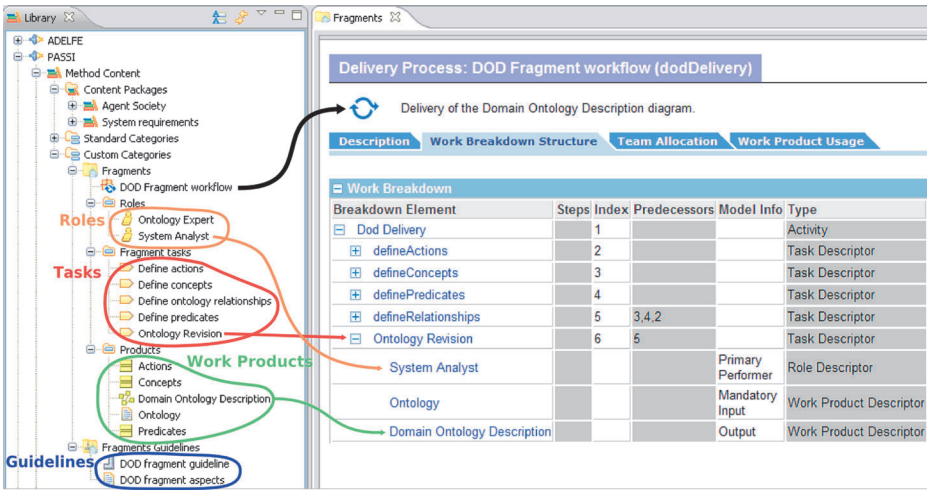

Fig. 2. DOD PASSI fragment definition in SPEM 2.0 using EPF

description of re-usable method fragments. At this stage, the definition of fragments only depends on their characteristics (see section 5.1). The following presents SPEM 2.0 fragment definitions. ADELFE characteristic parts (previously identified as fragments) as first instance, as well as a PASSI fragment. We use the EPF eclipse plugin to realize these definitions.

6.1 ADELFE Fragments

ADELFE has been defined as a sequence of *Work Definitions* containing *Activities, Work Products* and *Role*. As this description could not be straight forward translated to SPEM 2.0, due to concern separation mainly, we have to gather basic work definitions (*Tasks* in SPEM 2.0) into some kinds of categories. This is the purpose of *Discipline*: the *Tasks* intended to be executed during a particular *Phase* or *Iteration* are grouped in the same *Discipline*. In this example, we focus on the fragment called: "Verify AMAS adequacy" *Task*. As a SPEM 2.0 *Task*, it can be re-used and even extended in other method plugin. It consists of two *Steps*: "Verify adequacy at the global" and "local level". This *Task*/fragment is related to specific *Guidances*, which ease its use and integration within an other method: the "AMAS Adequacy" *Concept* and the "Adequacy tool" *Tool Mentor*[4]. We have added this task at the *Fragments* category, as well as its related guidances to the *Fragments Guidelines* category (as shown in the right panel of the screen shot in figure 2).

[4] *Concept* and *Tool Mentor* are both *Kind* of *Guidance*, they are defined in the SPEM 2.0 base plugin.

6.2 PASSI DOD Fragment

We choose the "Domain Ontology Description Fragment" as an other instance of SPEM 2.0 fragment description. According to its definition [5], it is a "coarse grain" fragment. It involves several tasks, work products in a specific workflow. We have made the choice of mapping it to a *Process With Method* element: a *Delivery Process*[5] which contains an *Activity* named "DOD delivery". This activity is broken down into tasks leading to the delivery of the DOD. We also have defined some inner categories to distinguish tasks from process, roles and products (see figure 2). Moreover, the DOD PASSI fragment defines "fragment aspects" that haven't been defined in ADELFE fragment. We have determined that this concepts should mapped to a "generic kind of guidance": *Supporting material* as it is used to contain "information not specifically covered by the other guidance types".

7 Discussion and Prospects

According to the previous examples, it seems that SPEM 2.0, thanks to its capabilities (as shown in section 4.2), goes one step further toward the provision of an agent-oriented *Method library* where all the identified methodology fragment should be defined and stored as reusable and extensible *Method Content* and *Processes* within *Method plugins*. Moreover, the SPEM standard provides a common frame or language to define methods, good practices and gained expertise in the field of AOSE, as well as a widespread use and tool support for the description, management or even enactment of processes.

Situational Method Engineering has promoted for years, the idea that not a single method could fit all method engineers needs, it strongly depends on the studied domain. From this idea, FIPA has proposed the concept of fragments which is intended to cope with this problem. We assume that the SPEM 2.0 version has reached a stage where modularity and re-usability area is sufficient to allow fragments definition. However, even if a descriptive specification is already possible using SPEM, some important issues still have to be faced. For instance, we have presented the description of fragments with EPF, this is not straightforward. In fact, *Process Component* is not implemented in EPF, methodological units can't be expressed so easily with encapsulation and interfaces (provided or requested). Although a task can be defined with input and output work products (as we discussed it in section 5.2) the SPEM 2.0 base plug-in provided by EPF does not allow to define coarse-grained encapsulated units.

Therefore, what about the assembly of such fragment? How can we fix problems such as input/output type verifications, name conflicts, etc.? How can we assist method engineer during the election of the more suitable fragment while they are devoted to different agent paradigms? One of the ideas proposed in [20] for the browsing and search into fragment repository, is the use of an ontology, or at least

[5] SPEM 2.0 base plugin special kind of *Process Package*.

a taxonomy, containing the concepts linked to each fragment and which can be use as a guide for the tailoring of a new fragment-based methodology.

Fragments interest is precisely that they focus on specific agent paradigms (ADELFE for emergent system and cooperative agent, ASPECS for holonic MAS, etc.). Thus, they depend on the underpinning method meta-model from which they are derived. This implies that, to perform fragment connection to another, a kind of mapping or "glue" have to be created because related concepts may belong to different paradigm or domain. Thus, building ontology over those concepts may ease this "glue" generation, maybe using model transformation. However, if we reconsider the ADELFE fragment of adequacy verification, is it really specific to AMAS or could it be generalized to any type of agent? Therefore, could it be used to verify adequacy of a system with holonic paradigm, provided that it has been extended or adapted. This implies that fragments should be parameterized with concepts like the type of agent or system (cooperative, BDI, holonic, etc.). For instance, the adequacy verification will become: "verification of the adequacy between solution and the problem domain".

However, assembly is a well-known problem in the scope of programming language, composing software component or aspects is not trivial. Therefore, composing methodology fragments appears to be a complicated task, thus automate the design of a new methodology will result in a much more complicated work.

This paper has presented some preliminary steps towards this further goal. The main work will be to propose tools to combine these fragments. Because it is a complex problem requiring adaptation, we can propose an adaptive multi-agent system to solve it where the fragments will be agentified.

References

1. Bergenti, F., Gleizes, M.-P., Zambonelli, F. (eds.): Methodologies and Software Engineering for Agent Sytems. Kluwer Academic Publishers, Dordrecht (2004)
2. Bernon, C., Camps, V., Gleizes, M.-P., Picard, G.: Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology. In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies, pp. 172–202. Idea Group Pub., USA (2005)
3. Capera, D., Georgé, J.-P., Gleizes, M.-P., Glize, P.: The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In: TAPOCS 2003 at WETICE 2003, Linz, Austria. IEEE CS, Los Alamitos (2003)
4. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the Tropos project. Information Systems 27(6) (2002)
5. Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. Int. J. of Agent-Oriented Software Engineering 1, 91–121 (2007)
6. Cossentino, M., Gaglio, S., Sabatucci, L., Seidita, V.: The PASSI and agile PASSI MAS meta-models compared with a unifying proposal. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS, vol. 3690, pp. 183–192. Springer, Heidelberg (2005)
7. Cossentino, M., Gaud, N., Galland, S., Hilaire, V., Koukam, A.: A holonic metamodel for agent-oriented analysis and design. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS, vol. 4659, pp. 237–246. Springer, Heidelberg (2007)

8. FIPA. Method fragment definition, fipa document edition (November 2003)
9. Gomez-Sanz, J., Pavon, J.: Agent Oriented Software Engineering with INGENIAS. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS, vol. 2691, p. 394. Springer, Heidelberg (2003)
10. Henderson-Sellers, B.: Evaluating the feasibility of method engineering for the creation of agent-oriented methodologies. In: Pěchouček, M., Petta, P., Varga, L.Z. (eds.) CEEMAS 2005. LNCS, vol. 3690, pp. 142–152. Springer, Heidelberg (2005)
11. Juan, T., Sterling, L., Martelli, M., Mascardi, V.: Customizing AOSE methodologies by reusing AOSE features. In: Rosenschein, J.S., Sandholm, T., Wooldridge, M., Yokoo, M. (eds.) International Conference on Autonomas Agents and Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, pp. 1024–1025 (2003)
12. Luck, M., Ashri, R., d'Inverno, M.: Agent-Based Software Development. Artech House, Inc., Norwood (2004)
13. Object Management Group, Inc. Meta Object Facility (MOF) 2.0 Core Specification (October 2003)
14. Object Management Group, Inc. Unified Modeling Language (UML) 2.0 Infrastructure Specification. Final Adopted Specification (August 2003)
15. Object Management Group, Inc. Unified Modeling Language (UML) 2.0 Superstructure Specification. Final Adopted Specification (August 2003)
16. Object Management Group, Inc. Software & Systems Process Engineering Metamodel Specification v2.0, omg edition (October 2007)
17. Omicini, A.: SODA: Societies and infrastructures in the analysis and design of agent-based systems. In: Ciancarini, P., Wooldridge, M.J. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 185–193. Springer, Heidelberg (2001)
18. Padgham, L., Winikoff, M.: Prometheus: A methodology for developing intelligent agents. In: Proceedings of the Third International Workshop on Agent Oriented Software Engineering at AAMAS (July 2002)
19. Rougemaille, S., Migeon, F., Maurel, C., Gleizes, M.-P.: Model Driven Engineering for Designing Adaptive Multi-Agent Systems. In: Artikis, A., O'Hare, G.M.P., Stathis, K., Vouros, G. (eds.) ESAW 2007. LNCS, vol. 4995, Springer, Heidelberg (2008)
20. Seidita, V., Cossentino, M., Gaglio, S.: A repository of fragments for agent system design. In: Paoli, F.D., Stefano, A.D., Omicini, A., Santoro, C. (eds.) Proceedings of the 7th WOA 2006 Workshop From Objects to Agents, September 2006. CEUR Workshop Proceedings, vol. 204, CEUR-WS.org (2006)
21. Szyperski, C., Gruntz, D., Murer, S.: Component Software: Beyond Object-Oriented Programming, 2nd edn. ACM Press/Addison-Wesley (2002)
22. Wooldridge, M., Jennings, N.R., Kinny, D.: The gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems 3(3), 285–312 (2000); Times Cited: 3 Article English Cited References Count: 34 412fd