# The future of AOSE: exploiting SME for a new conception of methodologies

Mariachiara Puviani, Giacomo Cabri, Letizia Leonardi
Università degli Studi di Modena e Reggio Emilia
Via Vignolese 905, 41125 Modena, Italy
{mariachiara.puviani, giacomo.cabri, letizia.leonardi}@unimore.it

## ABSTRACT

In the last years, the software engineering field has provided developers with different methodologies to support their work. Nevertheless, existing methodologies can hardly meet the requirements of all existing scenarios, which are more and more complex and highly differentiated. This problem can be faced by applying the Situational Method Engineering (SME) approach, which enables to build appropriate methodologies by composing "fragments" of existing ones. We envision this approach as the future of software engineering in general, and in particular if applied in Agent Oriented Software Engineering (AOSE). This approach has also the valuable advantage of reusing models, solutions, experiences and tools of existing and tested methodologies.

In this paper we report three examples of application of the Situational Method Engineering approach in AOSE. We show that this approach can be applied following different directions, and in particular: *entity-driven*, *metamodel-driven*, and *characteristic-driven*. To concretely show these directions, we present three examples of methodologies for developing agent systems (one regarding self-organising systems), all constructed composing methodology fragments to meet the scenario requirements.

## Keywords

SME, methodology, fragments, AOSE.

## 1. INTRODUCTION

Nowadays software engineering provides a lot of methodologies that help developers pass from models to implementations when building a wide range of software systems. Different scenarios can have different requirements at the *domain* level, which are more general than the ones at the *application*-level, and methodologies can suit such requirements with different degrees. When a developer needs a methodology to build her system, she has to choose among the different approaches in general and the different methodologies in particular. Usually a developer chooses the methodology she better knows and she has experience of, or a methodology that is particularly focused on the kind of system she would like to build. Others prefer to use methodologies that have strict connections with the implementation phase (i.e., with infrastructures), or that have a supporting tool that will guide the developer during all the process, from the design to the implementation of the system. Actually, different methodologies have no supporting tool, or are not strictly connected to infrastructures. In addition, a lot of them are suited for specific classes of problems and it is still very hard to cover all the range of possible scenarios. Even a deep developer's experience in one methodology cannot help if this methodology is not suitable for the scenario the developer must face.

To solve this problem, the developer can decide to create a new brand methodology for facing the requirements of the specific scenario, but this approach requires a lot of experience from the developer herself and, due to the bounded knowledge of a person, the result can be worse compared to the use of some existing methodologies, also if they are not created for the specific kind of faced problems.

The approach we exploit in this paper is called Situational Method Engineering (SME) ([25] and [22]) and it aims at not starting from scratch every time a developer has to face a specific scenario in building a new system. Instead, it supports the reuse of existing experiences and, at the same time, it enables to customize a specific methodology, on the base of specific needs. To do that, we will compose a new methodology exploiting *fragments* [7] (pieces of process) coming from existing methodologies and creating the missing fragments *ad hoc* to complete the new process. This is an approach already known in the software engineering field, even if not very spread yet due to its complexity. In this paper we present three different examples of application of this approach in the context of Agent Oriented Software Engineering (AOSE) [24]. The presented examples follow three different directions: *entity-driven*, *metamodel-driven*, and *characteristic-driven*. We belive that the use of the SME approach can be the future of AOSE with regard to methodologies. It can use existing experiences and new "ideas and features" to better exploit agent methodologies.

In the following we will explain the SME fragment approach (Section 2), then in Section 3 we present three different examples that use this approach to build *ad hoc* methodologies. Finally, after a short survey of related work (Section 4), in Section 5 we propose some conclusions and future work.

## 2. SME

The Situational Method Engineering (SME), first proposed by Kumar and Welke in [25], aims at defining methods to develop systems, by reusing and assembling different existing *portion of process*. Following the more general Method Engineering paradigm, every existing methodology can be decomposed not only into phases, but also into small parts of

process called *method fragments* that are stored in a *method base*. A method base is composed of contributions coming from existing methodologies. In Figure 1 we can better see the SME approach. The term *fragment* was first coined by Harmesen in [1].

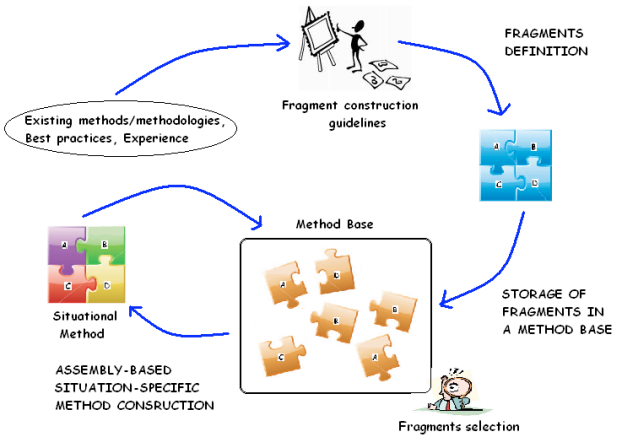


**Figure 1: Situational Method Engineering**

In the literature different kinds of method fragments have been proposed:

- Brinkkemperet et al.'s approach [6],
- Ralyté and Rolland (Method Chunk) [33],
- OPF (OPEN Process Framework) [19],
- FIPA [35].

In the follow, we briefly describe every different approach.

According to Brinkkemperet et al., a method fragment is a coherent piece of information system development. This approach considers two (sub)kinds of method fragment: the *Process fragment* that describes the stage, activities and tasks; and the *Product fragment* that concerns the structure of a process product (deliverables, diagrams, etc.). A fragment can be composed of other fragments and can have relationships with other fragments. Fur further details on this approach see [6].

Ralyté and Rolland say that a *method chunk* is a consistent and autonomous component that represents a portion of process with its resulting work products. It is represented using a metamodel (UML notation) composed of two parts: the *process aspect* and the *product aspect* [33].

The OPF method fragment is part of existing methodologies and is used to build new ones. It is based on the Object-oriented Process, Environment, and Notation (OPEN) approach, and it is generated and stored in a repository with all its guidelines based on OPF metamodel, which is composed of five metaclasses. Each metaclass produces a method fragment (process or product fragment). See [19] for details.

The FIPA method fragment is a reusable part of a design process composed of a set of activities performed by process roles in order to produce a kind of artefact (work product). It is based on the process description model from the OMG SPEM (Software Process Engineering Metamodel) and uses the related notation [29].

Some comparisons of these approaches have been made in literature (see for example [13], [23] and [26]). The approach proposed in this paper uses the FIPA fragment's notation, because it is a well known and widely used notation, it better suits our needs, and we find the SPEM support very useful. In the following, Section 2.1, we give more details about the FIPA method fragment.

## 2.1 Use of FIPA Fragments in SME

The FIPA Technical Committee Methodology [18] defines a fragment as a portion of the development process for Multi Agent Systems (MASs), composed of those (not always all) elements:

- A portion of process (what is to be done, in what order) defined using a SPEM diagram,
- One or more deliverables (artefacts like AUML/UML diagrams, text documents, etc.),
- Some preconditions (required input data or required guard conditions),
- A list of concepts (related to the MAS meta-model) to be defined (designed) or refined during the specified process fragment,
- Guidelines that illustrate how to apply the fragment and best practices related to that,
- A glossary of terms used in the fragment,
- Other information (composition guidelines, platform to be used, application area and dependency relationships useful to assemble fragments).

We use FIPA fragments to compose methodologies to meet the scenario's requirements. In the following examples we will face two kinds of scenario requirements: (i) the connection between agent methodologies and infrastructures (examples 1 and 2) and (ii) the characteristics of self-organizing system (example 3). Following the FIPA approach, our job will be supported by the use of SPEM [35].

Usually, in order to compose a methodology, the approach can be divided into three main phases, following the SME paradigm as assumed in [21]: *Process Requirement Analysis*, *Fragment Selection* and *Fragment Assembly*.

In the first phase, the process engineers have to evaluate the scenario requirements to be satisfied. This can be done in different ways, for example defining important entities that have to be considered in order to develop the system, or creating a metamodel, or considering the peculiar specifications of the scenario (see the three different examples reported in Section 3).

Then, starting from the evaluated requirements, the process engineers have to extract from the repository the fragments that both define these entities and suit the process. Sometimes fragments for their specific purpose do not exist, so with the help of SPEM they have to create *ad hoc* fragments starting from an existing methodology, or simply from necessities. While making the selection, the process engineers have to evaluate the fragments coming from different methodologies: sometimes the same entity can be defined by different fragments in different ways, and, after considering the original methodology of the fragment, they have to decide which one represent a better solution for their problems.

Until now the Fragment Assembly phase is the most difficult one because the process engineers have to evaluate all the different starting methodologies (methodologies from which the used fragments were extracted), and they have to connect the different fragments considering the meaning of entities and their relations. Today there is not any specific supporting tool for this work, but the PRoDe (PRocess for the Design of Design PRocesses) Fragment Repository created by Seidita et al. [36] has the Metameth tool to support the design process that is interesting to use and further develop.

# 3. USING THE FRAGMENT APPROACH

Studying SME, we find out that it can be very useful to build *ad hoc* methodologies by composing existing fragments or to adapt existing methodologies by adding specific fragments. This will permit to extend the use of existing "modified" methodologies to different scenarios. We used SME in three different directions, using three different approaches. In the following we present these examples in the AOSE field to show the possible uses of *fragment composition*.

The first example we present is MAR&A, a methodology that was created to be strictly connected to infrastructures; in this case, the methodology was built starting from the main *entities* of the infrastructures, so this direction can be called *entity-driven*. The second example is the MEnSA methodology, created during the MEnSA Project [37] to fill the existing gap between methodologies and infrastructures; this second direction is called *metamodel-driven*, since inside the project we first defined a metamodel and then we built the methodology based on it. The third example is related to self-organising systems, where no real methodology is available, while existing fragments can be exploited to fulfil the scenario's requirements; in this case, we have taken into account the *characteristics* of the scenario, which are not only the entities of the first case, leading to a direction *characteristic-driven*.

These three examples can be viewed as three different directions that can be followed for fragments composition.

## 3.1 The MAR&A methodology

Our work started with the evaluation of the state-of-the-art in the field of agent methodologies and infrastructures. With regard to methodologies for developing agent systems, we have chosen the most diffused ones, such as *ADELFE* [31], *Gaia* [38], *PASSI* [14], *SODA* [28] and *Tropos* [5]. With regard to the agent infrastructures, we studied *CArtAgO* [34], *JACK* [2], *JADE* [3], *MARS* [9], *RoleX* [8], *TOTA* [27] and *TuCSon* [30].

Starting from these methodologies and infrastructures, we aimed at integrating them, evaluating possible matching between their concepts, represented by *entities*. Starting from their metamodels we evaluated whether and to which extent a matching exists, in order to give a continuous support in the systems' development. Then, we chose the methodology fragments that manage the matching entities (reported in Figures 2, 3 and 4 using the SPEM notation). We call this direction *entity*-driven.

We focused on (i) the common processes and entities of the methodologies and (ii) the entities that enable a connection with the infrastructures. First of all, we have found out the entities common to the different infrastructures. This was useful to extrapolate the "core" entities, which deserved a support by the methodologies. From the evaluation of the infrastructures, these entities are *agent*, *role* and *action*.

We decided to propose a composed agent methodology, which has the infrastructures' main entities as goal for its outcome. We remark that such entities are not only common to different infrastructures, but also part of their foundation. This makes them the best candidates to be considered as the outcome of the new methodology; this also has the consequence that the outcomes of the new methodology can be implemented using different infrastructures. We started form these entities and construct all the entities around, looking at which where better connected in the starting methodologies.

The resulting composed methodology is called MAR&A (Methodology for Agent: Role & Action) (see [10] and [11]); it is an almost complete methodology, since nowadays three phases out of four of the whole process have been defined: *Requirements phase* (2), *Analysis phase* (3) and *Design phase* (4); the *Implementation phase* is under construction.

## 3.2 The MEnSA methodology

The MEnSA methodology was proposed in the context of the MEnSA project [37], whose aim was, as for MAR&A, to fill the existing gap between methodologies and infrastructures in the development of agent systems. However, differently from the MAR&A approach, for MEnSA we defined a *metamodel* for the methodology starting from the infrastructure-related requirements, in which we summarized the features of the methodology in terms of *entities* and *relationships* among them. These requirements helped to build a general methodology, which can be used in connection with different infrastructures. In this sense, we can say that this direction is *metamodel*-driven.

In this work we exploited the modified version of PRoDe (PRocess for the Design of Design PRocesses) approach, which is based on: the classic SME main phases cited before; a specific definition of method fragments (Process Fragment) [12]; and the system metamodel. PRoDe starts from the identification of the requirements of the new process in terms of development context, problem type and organization capabilities/existing processes maturity; these requirements are used to define the initial metamodel. Then, dif-

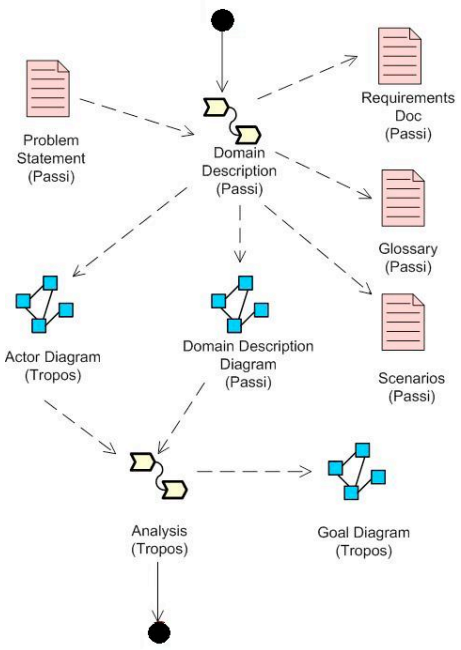


**Figure 2: MAR&A Requirement phase**

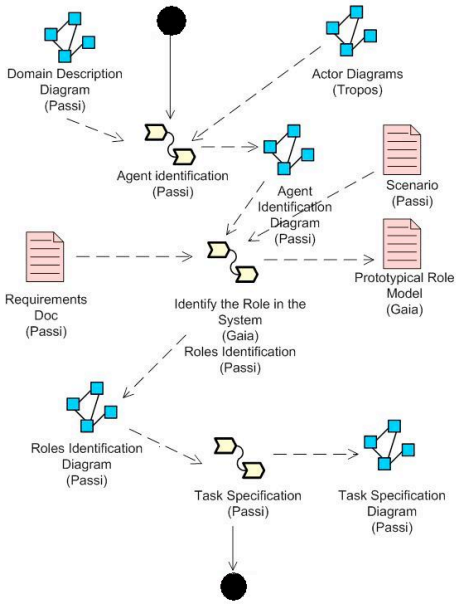


**Figure 3: MAR&A Analysis phase**

ferent from PRoDe, starting from the entities proposed, we have studied the project starting methodologies (Gaia [38], PASSI [14], SODA [28] and Tropos [5]) and we have found out a preliminary list of fragments that can define these entities. In parallel, we have identified the *order* in which the metamodel elements have to be instantiated during the development of the new process. To this purpose, we have exploited the *Prioritization algorithm* [36], which enabled us to choose the fragments relying on the relationships in the metamodel. For non existing fragments (i.e., those fragments that have not been extracted from an existing methodology yet, or cannot be derived from existing methodologies), we had to construct them. Finally, we had to check and, if needed, adjust the connections between the output of a fragment and the input of the next one.

In Figure 5 we can see a first result of fragment composition for the MEnSA methodology. It follows the *Prioritization algorithm* (e.g. the dark rectangles are fragments that have to be created).

## 3.3 Guidance for Self-organising Systems

Self-organising (simply *self-org* from now on) systems [16] are more and more exploited to solve complex problems, so it is very important to have methodologies that can help developers to build this kind of system. Nature provides many examples of self-organising systems: from non-living systems (e.g. Bénard convection cells, mud cracks) to living systems, such as biological processes of pattern formation. These systems exploit desirable (complex) properties such as robustness, resilience or self-reconfiguration, while the individual entities forming these systems can be seen in terms of agents, having a certain degree of autonomy, proactiveness and able to interact.

Studying the state of the art, we find out that there are some approaches to build self-org systems, but they are not real methodologies (except for ADELFE, which has other limitations not reported here). As we can see in [32], our study points out that there are well-defined *characteristics* of the self-org behaviour; for developers it is very important to be able to address them. We call this direction *characteristics*-driven.

Among others, the main characteristics that could be supported by existing methodologies' fragments (in particular taken from AOSE) are:

- *endogenous global order*: the system reaches some global state that is produced from within the system itself;
- *locality*: the components of the system are aware of their location and of local resources;
- *emergent properties*: there are properties that cannot be found out by simply observing individual behaviour;
- *adaptation*: the system can react to the changes of the environment.

As we can see from the above list, these characteristics differ in nature from the entities exploited in the MAR&A approach, even if they can involve one or more entities. It is also an approach different from the MEnSA one, since a metamodel is not involved.

A unique methodology cannot be general enough for describing every self-org system. For this purpose, it can be very convenient to have the possibility of reuse the methods to develop important features, already provided by existing methodologies such as *Adelfe* [31] or a *General Methodology* [20].

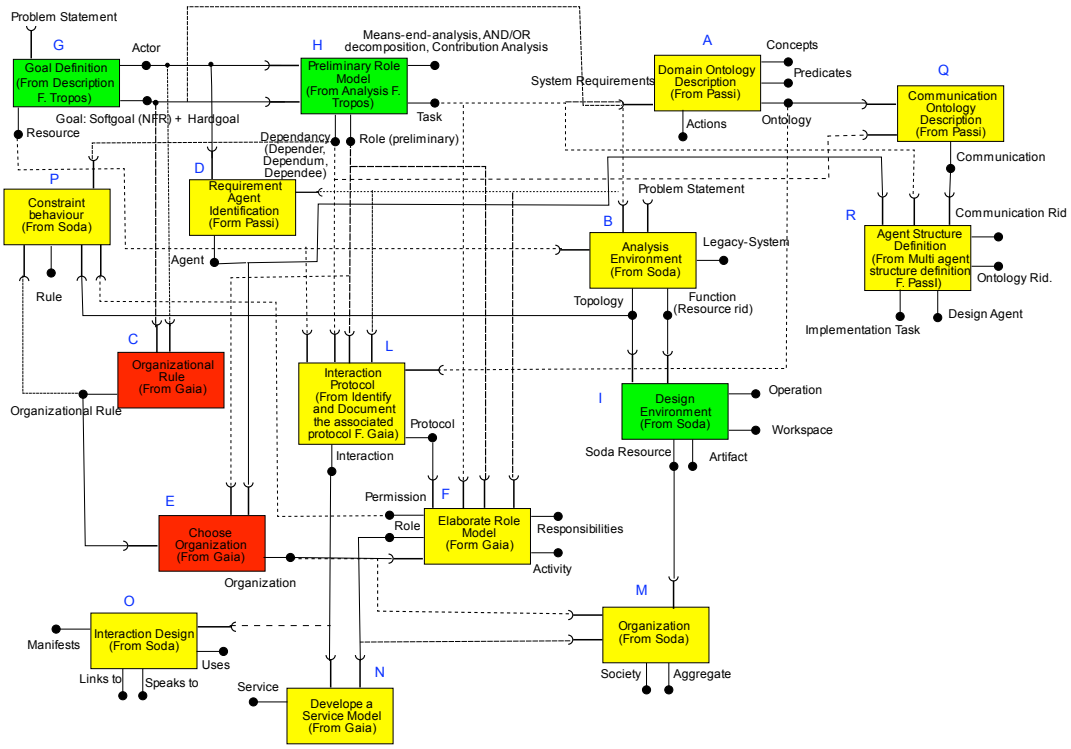


**Figure 5: MEnSA methodology following the Prioritization algorithm**

We exploited some case studies [32] to find out which fragments are necessary to build a self-org system, then we extracted them from the considered methodologies (see for example Figure Figure 6), to make the developers able to integrate them in their own methodology.

These fragments alone are not enough to build a complete methodology, because we have to consider more entities like goal, role, activity, plan, ontology and so on, according to developer's needs; nevertheless, we wrote some guidelines [32] to help developer integrate these fragments in existing methodologies. In these guidelines we underline how a fragment is related to a specific characteristic of self-org systems, and in this way a developer can be able to insert the specific fragment into her methodology.

## 4. RELATED WORK

In literature we find other approaches concerning fragments composition. Since 1999, [7] proposes a framework for hierarchical method modelling (meta-modelling) using methods and/or method fragments. The paper explains how to assemble method fragments into a situational method and formalize rules to construct meaningful methods. In [4] metamodels of existing methodologies are compared in order to be composed in a single metamodel. Here the fragment approach is not directly used but the main idea is the same of ours: to compose existing methodologies in order to create another one. In [15], the authors start form PASSI and some important requirements to compose the original methodology with other fragments in order to create Agile PASSI. This methodology was born to allow the quick prototyping of agent-oriented applications. One of the last examples is GORMAS (Guidelines for ORganizational Multi-Agent Systems) [17], an Organizational-Oriented Methodology that, using SPEM, aims at integrating MOISE, ANEMONA and INGENIAS meta-models, so as to cover all typical aspects for designing agent systems, but also all organizational and functional aspects needed from both Organization Theory and Service-Oriented paradigm it is based on.

## 5. CONCLUSIONS AND FUTURE WORK

Situational Method Engineering (SME) is an approach that enables the definition of new methodologies by composing fragments of existing ones. It is a very powerful approach and we figure out that the complexity of current and future scenarios will require such an approach in order to support the job of process engineers, even if it is currently considered quite complex to apply. This is why we think that it can be the future of AOSE.

In this paper we have shown three examples of method composition applying SME, which follow three different directions:

- an *entity-driven* direction (MAR&A), which considers the main entities of the scenario;
- a *metamodel-driven* direction (MEnSA), which relies on a metamodel that defines entities and their relationships;
- a *characteristic-driven* direction (self-org), which considers characteristics of the scenario in which the application is to be developed; we remark that entities of

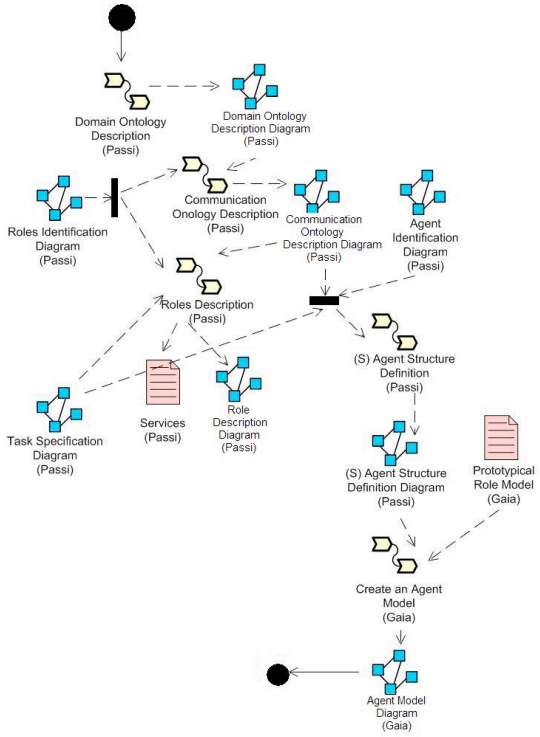


**Figure 4: MAR&A Design phase**

the first direction can be a *subset* of the characteristics of this direction.

Comparing them, we found out that the *metamodel-driven* direction is more complex, because a new metamodel must be defined, but it leads to a well-founded and more flexible methodology, allowing easier modifications and maintenance. Moreover, it is a more general direction, which can be followed even the developer does not know in details the interested entities (or infrastructures in our case).

Instead, the *entity-driven* direction allows to focus on specific entities (of infrastructures in our case), thus producing a more suited methodology for specific needs, but more difficult to adapt in case of changes in the scenario.

Finally, the *characteristic-driven* direction relies on higher-level requirements, such as the ones related to characteristics, instead of entities; so it can be adopted where the scenario does not impose well-defined entities, rather functional or non-functional requirements. This approach can be easier to apply because it permits to reuse a whole existing methodology, adding the specific fragments. But at the same time the developer needs a great capacity of integration and adaptation to introduce the new fragments in the known methodology.

As mentioned in the introduction, the SME approach is very powerful but at the same time quite complex. With regard to future work, our aim is to propose a semi-automatic approach that can be exploited to support the composition of methodologies following the SME paradigm; to this purpose we will exploit state-of-the-art techniques to manage the fragments.

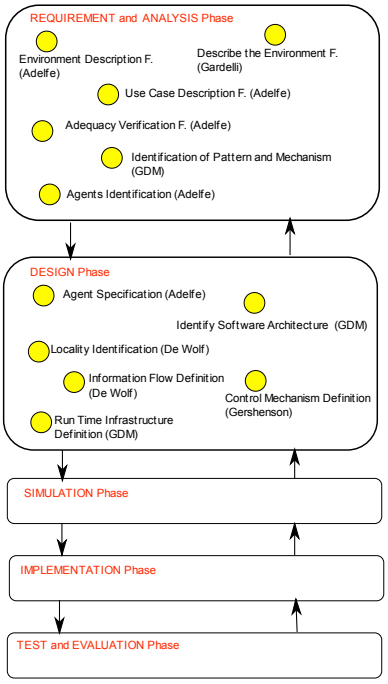


**Figure 6: Positioning of extracted fragments for self-org systems**

# 6. REFERENCES


[1] H. A.F., S. Brinkkemper, and H. Oei. Situational Method Engineering for Information System Projects. In *Proc. of the IFIP WG8.1 Working Conference CRIS'94*, pages 169–194, 1994.

[2] AOS Autonomous Decision-Making Software. Jack agent platform. http://www.agent-software.com/, 2008.

[3] F. Bellifemine. Developing multi-agent systems with jade. In *Proceedings of PAAM 99, London (UK)*, pages 97–108, 1999.

[4] C. Bernon, M. Cossentino, M. Gleizes, and P. Turci. A Study of Some Multi-agent Meta-models. In *Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004, New York, NY, USA, July 19, 2004: Revised Selected Papers*, page 62. Springer, 2005.

[5] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[6] S. Brinkkemper, M. Saeki, and F. Harmsen. Assembly techniques for method engineering. *Lecture Notes in Computer Science*, 1413:381–400, 1998.

[7] S. Brinkkemper, M. Saeki, and F. Harmsen. Meta-modelling based assembly techniques for situational method engineering. *Information Systems*, 24(3):209–228, 1999.

[8] G. Cabri, L. Ferrari, and L. Leonardi. Enabling mobile agents to dynamically assume roles. In *Proceedings of the ACM Symposium on Applied Computing, Melbourne (USA), March*, pages 56–60, 2003.

[9] G. Cabri, L. Leonardi, and F. Zambonelli. MARS: a programmable coordination architecture for mobile agents. *Internet Computing, IEEE*, 4(4):26–35, 2000.

[10] G. Cabri, M. Puviani, and L. Leonardi. The mar&a methodology to develop agent systems. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART 2009), Porto Portugal, January 2009*, 1 2009.

[11] G. Cabri, M. Puviani, and R. Quitadamo. Connecting Methodologies and Infrastructures in the Development of Agent Systems. In *The V Workshop on Agent-Based Computing (ABC 08) at IMCSIT 2008, Wisla, Poland*, 2008.

[12] M. Cossentino, S. Gaglio, A. Garro, and V. Seidita. Method fragments for agent design methodologies: from standardisation to research. *International Journal of Agent-Oriented Software Engineering*, 1(1):91–121, 2007.

[13] M. Cossentino, S. Gaglio, B. Henderson-Sellers, and V. Seidita. A metamodelling-based approach for method fragment comparison. In *Proceedings of the 11 th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 06)*, 2006.

[14] M. Cossentino, L. Sabatucci, S. Sorace, and A. Chella. Patterns reuse in the PASSI methodology. In *ESAWŠ03*, pages 29–31. Springer, 2003.

[15] M. Cossentino and V. Seidita. Composition of a New Process to Meet Agile Needs Using Method Engineering. *Software Engineering for Multi-agent Systems III: Research Issues And Practical Applications*, page 36, 2005.

[16] G. Di Marzo Serugendo, M. Gleizes, and A. Karageorgos. Self-organization in multi-agent systems. *Knowledge Engineering Review*, 20(2):165–189, 2005.

[17] V. B. E. Argente and V. Julian. GORMAS: An Organizational-Oriented Methodological Guideline for Open MAS. In *Agent-Oriented Software Engineering: 10th International Workshop, AOSE 2009, Budapest, Hungary, May 10, 2009*, pages 85–96. Springer, 2009.

[18] FIPA Methodology Technical Committee. FIPA. http://www.pa.icar.cnr.it/∼ *cossentino/FIPAmeth*.

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

[20] C. Gershenson. Design and control of self-organizing systems. *Vrije Universiteit Brussel*, 2007.

[21] D. Gupta and N. Prakash. Engineering methods from method requirements specifications. *Requirements Engineering*, 6(3):135–160, 2001.

[22] B. Henderson-Sellers. Method engineering: Theory and practice. In *Information Systems Technology and its Applications. 5th International Conference ISTA 2006*, pages 13–23, 2006.

[23] B. G.-P. Henderson-Sellers and J. C. Ralyté. Comparison of Method Chunks and Method Fragments for Situational Method Engineering. In *Proceedings of the 19th Australian Conference on Software Engineering*, pages 479–488. Technol. Univ., Sydney, 2008.

[24] N. Jennings and M. Wooldridge. Agent-oriented software engineering. *Lecture notes in computer science*, pages 4–10, 1999.

[25] K. Kumar and R. Welke. Methodology Engineering R: a proposal for situation-specific methodology construction. *John Wiley Information Systems*, pages 257–269, 1992.

[26] X. Larrucea. Situational Method Fragment Selection and Composition. In *Composition-Based Software Systems, 2008. ICCBSS 2008. Seventh International Conference on*, pages 243–243, 2008.

[27] F. Mamei and F. Zambonelli. Programming stigmergic coordination with the TOTA middleware. In *Proceedings of the $4^{th}$ international conference on Autonomous Agents and Multi-Agent Systems, New York (USA)*, pages 415Ű–422, 2005.

[28] A. Molesini, A. Omicini, E. Denti, and A. Ricci. SODA: A roadmap to artefacts. *Engineering Societies in the Agents World VI*, 3963:49–62, 2006.

[29] Object Management Group. SPEM. http://www.omg.org/technology/documents/formal/spem.htm.

[30] A. Omicini and F. Zambonelli. Coordination for internet application development. *Autonomous Agents and Multi-Agent Systems*, 2(3):251–269, 1999.

[31] G. Picard and M. Gleizes. The ADELFE Methodology–Designing Adaptive Cooperative Multi-Agent Systems. *Methodologies and Software Engineering for Agent Systems. Kluwer Publishing*, 2004.

[32] M. Puviani and G. D. M. Serugendo. Methodologies for Self-Organising Systems: a SPEM Approach - preliminary draft. Technical Report DII-AG-2008-2, Dipartimento di Ingegneria dell'Informazione University of Modena and Reggio Emilia, 2008.

[33] J. Ralyté and C. Rolland. An approach for method reengineering. *Lecture Notes in Computer Science*, pages 471–484, 2001.

[34] A. Ricci, M. Viroli, and A. Omicini. CArtAgO: A framework for prototyping artifact-based environments in MAS. In D. Weyns, H. V. D. Parunak, and F. Michel, editors, *Environments for MultiAgent Systems*, volume 4389 of *LNAI*, pages 67–86. Springer, Feb. 2007.

[35] V. Seidita, M. Cossentino, and S. Gaglio. Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies. *Proceedings of CAiSE 07*, 2007.

[36] V. Seidita, M. Cossentino, V. Hilaire, N. Gaud, S. Galland, A. Koukam, and S. Gaglio. The Metamodel: a Starting Point for Design Processes Construction. *International Journal of Software Engineering and Knowledge Engineering*, 2009.

[37] The MEnSA project. MEnSA Web site. http://www.mensa-project.org/.

[38] F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3):317–370, 2003.