Forward Self-Combined Method Fragments

Noelie Bonjean, Marie-Pierre Gleizes, Christine Maurel, Frédéric Migeon

Forward Self-combined Method Fragments

Noélie Bonjean, Marie-Pierre Gleizes, Christine Maurel, and Frédéric Migeon

IRIT, Université Paul Sabatier
F-31062 Toulouse cedex 9, France
Firstname.Name@irit.fr

Abstract. Developing complex systems is generally simplified if designer is guided by method from Software Engineering. However a single engineering process is often not enough to cover all the possible requirements due to different levels of expertise and systems to design. Currently, Agent Oriented Software Engineering methods aim at providing an adaptive engineering process. The method processes have been broken up into different parts called fragments, enabling the mix of different engineering processes' parts to get better adequacy between the system to be done and the process. But some difficulties still remain concerning the expertise needed to compose these fragments when the amount of fragments prevents the composition to be done by hand. This paper presents an Adaptive Multi-Agent Systems (AMAS) to deal with a new paradigm of automated fragments combining. This process is made from both the characteristics of users and system and the known fragments. Thanks to their information, agents of the AMAS self-organise and design a tailored method process. The developed system is described and then usual tests are depicted.

1 Introduction

Software reuse is generally considered as one of the most effective ways of increasing productivity and improving quality of software. To make software reuse happens, however, there is a change in the way engineers develop software: software is currently developed for reuse and with reuse. Component-based software engineering [1] is a software engineering paradigm in which applications are developed by integrating existing components. Reuse of software engineering is becoming more and more important in a variety of aspects of software engineering.

In the same way, in Agent-Oriented Software Engineering (AOSE), a lot of different methods, each with its advantages and its drawbacks [2]. Methods have to deal with these characteristics and capabilities of agents or systems. An attempt is to benefit from different methods by combining their particular features. For example, attempts have been made to combine requirements analysis in TROPOS and self-adaptation in ADELFE [3].

Coming from Situational Method Engineering, decomposing processes into pieces has interested the AOSE community because of the expected benefits of flexibility. The aim is to adapt the process to the characteristics of the business

problem and to the level of expertise of engineer teams by proposing to assemble pieces of methods, named fragments, of various processes. In a first step, several teams have to split up methods into fragments and provide a precise description of them (ADELFE[4], INGENIAS [5], PASSI [6], TROPOS [7] ...). Two main results have been obtained from this step: (i) a means to precisely compare the different methods and (ii) a potential repository of fragments that will serve the community to compose new processes [8]. This kind of work is mainly done in the Foundation for Intelligent Physical Agents[1] (FIPA) context.

Currently some propositions to combine fragments have been already made, but they are mainly based on the know-how of the method engineer. In this paper, we propose a first step forward an automatic way to design a method process based on MAS technology. The process is constructed by combining fragments "on the fly" to be adapted to the specific situations of the projects at hand. The new process is based on both fragment compatibility and user characteristics. In order to respond to this need, the presented work details the use of an Adaptive Multi-Agent Systems (AMAS) which relies the cooperation of its agents to work together, making this approach especially suited to deal with highly dynamic systems such as the design of an interactive and adaptive Software Engineering Process (SEP) [4]. In this work, an AMAS is built by modelling fragments as autonomous entities.

This paper is organized as follows. First, section 2 explain the aim of this system. Then, section 3 introduces the system of Self-Combining fRagments (SCoRe) and details the behaviour of the involved agents as well as their interactions. Section 4 focuses on some usual tests and explain the results obtained. Finally, section 5 describes related works before concluding in section 6.

2 Why Such a System?

Request of Tailored Method. While the demand for specific, complex and varied system continues to grow, current methods in the MAS domain remain limited and sometimes not well adapted. For example, in order to propose a simulation-based process for the development of MASs which incorporates a simulation phase for the prototyping of the MAS being developed and for functional and non-functional validation, PASSIM was obtained by integrating method fragments from the PASSI for carrying out the analysis, design and coding phases, and the Distilled State Charts (DSC)-based simulation method for supporting the simulation phase [9]. The need for well-defined guidelines that will make the development process more efficient and more effective has become crucial. Currently, there is no single methodology that can be uniquely pointed as "the best". Until now methodology adjustments to the specific requirements and constraints are mixed in "local" adaptations and modifications. In order to succeed in creating good situational methodologies, i.e., methodologies that best fit given situations, fragment representation and cataloguing are very important

[1] `http://www.fipa.org`

activities. In particular, the fragments (sometimes addressed as process fragments, method fragments or chunks) have to be represented in a uniform way that includes all the necessary information that may influence their retrieval and assembling.

Fragment Standardisation. Method fragments are first identified by examining existing methods. These method fragments are made according to templates defined by repository designers. Therefore the choice of fragments granularity relies on designers. According to the RUP [10], the methods are defined following different levels of granularity: phase, activity and step. The granularity issue of these method fragments poses important challenges. The "step" level involves a specific and fiddly task but also requires perfect knowledge of methods and long work. This fragmentation is very fine-grained and provides a greater number of fragments. For instance, in ADELFE, the analysis phase is composed of four activities, the first of which is *Analysis of domain*. *Analysis of domain* consists of two steps: *Identify the active and passive entities* and *Study interactions between the entities*. These steps are related and interdependent. This low level of granularity is therefore useless and inaccurate in this situation. On the other hand, the "phase" level of granularity could form huge complete fragments. The coarse-grained granularity promotes the redundancy issue. The duplication of activities or steps may occur with high granularity. An activity or step may be included in different fragments. The risk that happens grows up with the level of granularity. In addition, the joining possibilities are therefore minimized.

Amount of Fragments. Currently, ten AOSE methods are fragmented, each one composed of approximately twenty fragments. Such fragments constitute the root constructs of the methodology itself and they have been extracted by considering a precise granularity criterion: each group of activities (composing the fragment) should significantly contribute to the production/refinement of one of the main artefacts of the methodology (for instance, a diagram or a set of diagrams of the same type). Following this assumption, fragments obtained from different methodologies are based on a similar level of granularity.

Besides, to design a process manually means studying for the compatibility of each fragment with the others i.e. approximately twenty thousand possible combinations. Although this number can be decreased by the knowledge and the know-how of process engineers, the work remains long and irksome. It is why we propose the automated combining of fragments.

Assist Designer. In our approach, a new complete process is firstly self-designed contingent on situation. The complete process enables engineers to visualise all activities and to have a whole view of the process. Then, we focus on adaptation during process execution. In every step the development team is advised on its next fragment choice according to the running features. If the features evolve, this advice may therefore differ from the following fragment initially suggested.

The studied solution resides in fragments agentification in order to design an adaptive process. This choice is justified by the problem complexity which is mainly due to the huge number of fragments. Indeed, a complex system cannot currently be designed without bugs from designers. Assist the designer during the system utilization would reduce the number of bugs and make the system most suitable to the current situation. The adaptation is therefore required. As components assembling, fragments assembling needs assembling features. In our approach, a fragments assembling is based on MAS Metamodel Elements (MMME). Two fragments are therefore assembling if one produces the MMMEs required by another.

3 Combining Method Fragments with an Adaptive Multi-agent System

The general structure of the Self-Combined method fRagments system (SCoRe) proposed is described in this section, before detailing the behaviours and the interactions of the agents composing it.

3.1 General Structure of SCoRe

We consider a method process as a set of assembled method fragments which are linked through their own required or produced MMMEs. Establishing a method process consists in linking some of the fragments toward user-defined objectives and knowledge. So, the main goal of SCoRe is to suggest a tailored process. For that, SCoRe learns the context to apply on fragments, in order to sustain this evolution. SCoRe has to act without relying on a model of the processes, meaning that it is only able to take into account the users' knowledge and needs, and to observe the evolution of the running process on which MMMEs are available, in order to decide on the fragments to add. The best possible running process is therefore designing according to a situation and the best adapted fragment has to be found at any moment.

SCoRe is composed of four distinct kinds of agents following a perception-decision-action lifecycle which cooperate according to the AMAS theory described in [11]. The basic idea underlying this cooperation consists, for every agent in an AMAS, in always trying to help the agent which encounters the most critical situation from its own point of view. Figure 1 gives the structure of a SCoRe system designing a method process. The different types of agents involved are shown, as well as the links modelling the existing interactions between them. Actually, our system is made up of:

- **MAS Metamodel Element (MMME):** required or produced by a Running Fragment, its aim is to decide fragments whom it will be linked.
- **Waiting Fragment (WF):** its purpose is to be integrate in a process once it is in an adequate situation.
- **Running Fragment (RF):** it aims at finding its place inside the running process.

- **Context (C)**: related to a fragment, it aims at evaluate its pertinence according to the MMMEs already involved in the running process and the users' needs and knowledge.

Next sections will provide a more in-depth description of these agents and interactions.

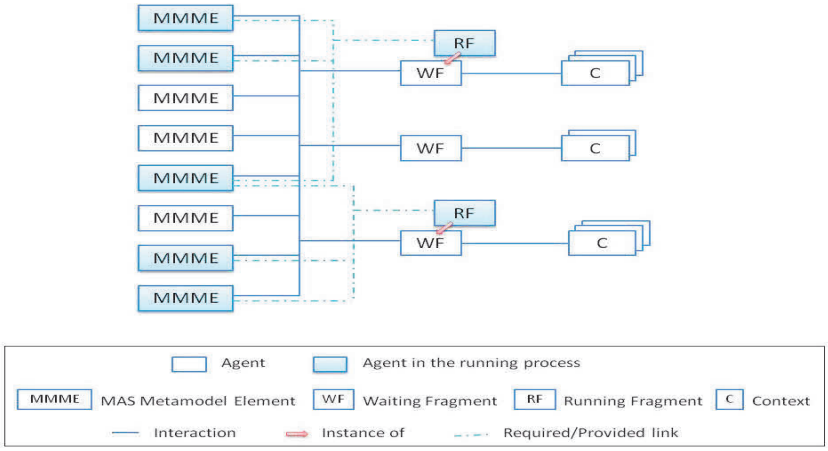

Fig. 1. Example of agents and their relationships in SCoRe

3.2 Behaviour of Agents

MMME Agents. The MMME agents represent the links between the running fragment agents. Their goal is to be incorporated in the running process. The MMMEs behaviour is represented by an automaton with two states: *non incorporated* and *incorporated*. The *non incorporated* state corresponds to a MMME linked to at least one running fragment which produces or consumes it. In this state, it requests fragments (consumer or producer). It is looking for a fragment at which it can be linked. It receives some answers from fragments with their relevancy. The most relevant fragment will be single out by the MMME agent for being put it forward as a candidate to be added in the running process. Furthermore, these agents are able to evaluate their own criticality. This criticality estimates the difference between the current and expected designed process and represents the degree of satisfaction of an agent. Therefore, the MMME agents cooperate on the selection of the most relevant fragment among the ones suggested according to their own criticality.

The *incorporated* state is reached when the MMME agent is linked with at least two fragments: one consumer and one producer. The given or required MMMEs by the designer have only to be linked respectively to at least one consumer and one producer.

Waiting Fragment Agents. The waiting fragment agents are reactive agents. Actually their goal is to notify the other agents of any requests from MMMEs. They receive messages from MMMEs which are looking for a fragment. If the waiting fragment agent considers himself as a potential solution then it forward the request to his context agents. It waits the answer from their context agent and answers his relevancy to the MMME. Should the opposite occur, the waiting fragment agent sends an answer to MMME with no relevance. Besides, when the waiting fragment agent receive a message from the MMME to inform it that it is selected, it transmits the information to the context agents. Then the waiting fragment agent spreads to create a running fragment agent.

Running Fragment Agents. The running fragment agent is created by the waiting fragment agent which represents in the running process. It is introduced on time in the process. His aim is to be incorporated in the method process. His behaviour changes according to his current state and his perception. The current state of a running fragment agent corresponds to *non incorporated* and *incorporated*. Actually, a running fragment agent is said *incorporated* when all the required MMMEs are in the *incorporated* state and at least one of the provided MMMEs is *incorporated*. Otherwise his state is *non incorporated* and the running fragment agent makes links with each MMME agent existing in the running process on which a link is physically possible. If some MMME agents are missing in the running process, the running fragment agent adds them. Furthermore, these agents are able to evaluate their own criticality. This criticality estimates the difference between the current and expected method process. It is calculated from the criticality of required or produced MMME(s) and their current state.

Context Agents. The context agents have the most complex behaviour in the SCoRe system. Their goal is to represent a situation leading to a specific method process. They do not aim to model what is happening inside the system, but rather aim at selecting the fragment to add in the current situation to reach the objectives. When such an agent finds itself in its triggering situations, it notifies the waiting fragment agent, by submitting its confidence according to its own knowledge.

In order to know when the fragment is relevant, a context agent relies on two different sets of information. First, a collection of input values represents the set of user and system characteristics. This element enables the context agent to know if it has to be triggered or not. Then, a context agent possesses a set of forecasts, which describes the impact of the action proposed on the criticality of the different variables of the system. Then, a context agent possesses a set of metrics, which describes the impact of the action proposed on the running process [12]. Those input values are modified during the life of a context agent. According to its behaviour, a context agent therefore adjusts its confidence from different feedback that it receives.

Finally, the behaviour of a context agent is represented by an automaton. Each state relates its current role in the MAS. A total of three different states exist: *disabled*, *enabled* and *selected*. The context agent can switch from a state to another thanks to the messages it receives from other agents in the system. A disabled context agent considers itself non-relevant in this specific situation. An enabled context agent thinks that it is relevant and potentially deserves to be selected. It then computes its confidence and sends them to the corresponding waiting fragment agent. Finally, a selected context agent is validated by a waiting fragment agent and its associated fragment is added in the running process. This selected context agent has then to observe the consequences of its action in order to reinforce or update its confidence.

4 Usual Tests

Considering the large number of existing method fragments, the volume of supporting studies and the users' profile, the need arises for a designed method. The designed method is conceived of not as a single interdependent entity but as a set of disparate fragments. Therefore, in order to show the rightness of a new method process, the method process has to be evaluated by several engineers for some specific system. The experience results of empirical studies that have been conducted by many practitioners and researchers. This kind of experience is complex and can take a long time to obtain sufficient results. Therefore, we firstly focus on the functional adequacy and the dynamic adaptation to specific situation.

We defined test sets corresponding on the one hand to the feasibility of this system and on the other hand to the specific situations encountered and solved by cooperation between agents. The first test is based on a set of fragments from current methods such as ADELFE[2], INGENIAS[3] and PASSI[4]. Fragments description can be found respectively on corresponding research team site. This first test aims at verifying that the system self-designs and proposes a complete method process. The sets of fragments from ADELFE, INGENIAS and PASSI enables to show the accurate behaviour of agents and their right assembling. In this case, at the set-up, all fragments are provided without order and the process is built up again. According to the users and system characteristics, one of them is therefore built up and suggested to the designer. The system is therefore able to propose the known processes.

The following test set uses fictive fragments to highlight accurate configurations. Two processes named A and B are defined. A is broken in four fragments a1, a2, a3, a4 where all fragments are sequential except for a2 and a3 which are alternative. The process B is broken in four sequential fragments b1, b2, b3 and b4. Moreover, the two processes are totally disjointed (except for last test case). They do not share MMMEs. For these tests, we chose to provide few fragments

[2] `ftp://ftp.irit.fr/IRIT/SMAC/DOCUMENTS/RAPPORTS/`
[3] `http://grasia.fdi.ucm.es/main/node/241`
[4] `http://www.pa.icar.cnr.it/cossentino/FIPAmeth/docs/`

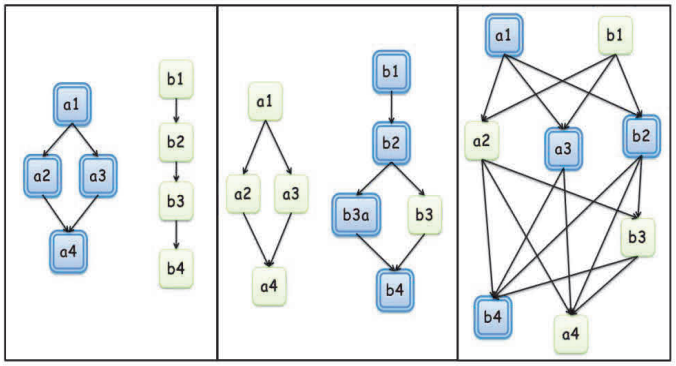

Fig. 2. Test Cases: Context Adaptation (left); Dynamic Adaptation (middle); Processes Combination (right)

for better readability and to show the system adaptation by cooperation between agents which are in particular situations. Moreover, in the following tests, the both processes are shown but the most accurate method process is only designed.

The second test verifies adaptation according to users and system characteristics. Indeed, in any situation, it controls that the system advises the most adapted process. In this case, the waiting fragment agents represent only the fragments from the independent fictive processes A and B. As a result, one is chosen as the most adapted for the specified situation. This test is showed in the figure 2 (left). The double borderline around a fragment shows the process A as being the most accurate process.

The third test is about dynamic adaptation. Considering open systems, waiting fragment agents are added or removed in system during runtime. The system has to take into account these modifications and reorganises itself according to its new state. The initial conditions are the same as in the previous test. A new waiting fragment agent named b3a is then introduced in the system. This fragment more accurate for the situation is an alternative fragment to b3. It is therefore included in the running process and a new process is defined. Figure 2 (middle) shows system adaptation after the introduction of a new fragment.

The last test case shows combining processes. In this case, fragments from different processes are assembled in order to obtain a new process more adapted. In this test case, we supposed that the provided fragments from A and B are compatible. The required MMMEs are also provided by a fragment from another process. The system is therefore able to produce a new process based on fragments from both initial processes in addition to processes already known. Figure 2 (right) shows this test where the new process composed of a1, b2, a3 and b4 is advised as the most accurate.

5 Related Works

In the MAS community, the first works on fragments, their definition and their composition have been started by the working group "Methodology Technical Committee" in 2003 [8]. Currently, the working group named "Design Process Documentation and Fragmentation Working Group" aims at providing IEEE FIPA specification of fragments. The working group approach is based on SPEM extension of OMG [13], and tailored to needs of agents and MAS.

The objective of SME approach in agent oriented engineering field is to propose the most accurate process in development context. The PRoDe (PRocess for the Design of Design PRocesses) [14] approach proposes to use the MAS metamodel as a central element for selecting and assembling fragments. PRoDe contains three phases: process analysis, process design and process deployment. The analysis phase elicits requirements and leads to MAS metamodel definition. The design phase helps designer to select fragments to assemble in a new process. Finally in the process deployment phase, the new process is used to solve a specific problem.

Based on PRoDe approch, MEnSA[5] project dissents to it from analysis phase. Indeed, in this phase, requirements are used to chose fragments and fragments contribute to metamodel definition [15]. The fragments repository includes fragments from the following agent oriented methods: PASSI, GAIA, TROPOS and SODA.

The OPEN framework [16] is object oriented method based on reusable methods components. It was extend to take into account agent oriented methods and come to FAME (Framework for Agent-oriented Method) conception. FAME is an agent oriented methods repository containing for example GAIA, TROPOS or PROMOTHEUS [17].

Tools are also developed in order to make easier the methods designing by combination of fragments, as MetaMeth [18]. It is a computer-aided process engineering tool (CAPE tool) and plug-ins to assist designer during process design from available fragments included data base.

As presented approaches, ours is based on current data base of fragments and on MAS metamodel. On the other hand, it is original because it proposes an automation of fragments composition. The designer is less called upon than ProDe or MEnSA approach because the most accurate fragments are presented to him already placed in the process. Moreover, in running development, our approach can take into account process adaptation according to development context.

6 Conclusion and Future Works

This paper presented an Adaptive Multi-Agent System to design a tailored process by linking fragments together. Each agent composing the AMAS follows a local and cooperative behaviour, driven by the use of their confidence. The four

[5] `http://apice.unibo.it/xwiki/bin/view/MEnSA/`

different kinds of agents composing SCoRe were designed in order to self-design a tailored method process without relying on the method engineer. The resulting behaviour of SCoRe is the ability to design a process and adjust the proposed process according to the characteristics of application domain and users profile. This first prototype allowed to enhance our experience about practical problems such as metamodel compatibility, parameters composition or fragments adaptation to specific field.

However, there is still room from improvements in some aspects of this approach. For example, the inter-operability and the semantic matching of fragments from different methods are still missing. In this problem, some works axis on standardisation of fragments notion and of their description. The metamodel definition or ontologies for software process could be used. Another approach from model-driven engineering is the Model Transformation By Example (TTBE). The concept is to make easier model transformation writing without generic model in favour of requested generated transformation. Thus fragments drawing on similar metamodels could be made up automatically.

Moreover, another important point is the evaluation of the designed process. Actually, despite the proposal of elaborate tailored method processes, methods are built intuitively by adopting some fragments from different methods. It is therefore difficult to evaluate and compare methods. In order to made a right choice, it is necessary to evaluate the method.

Finally, this SCoRe system will be confronted to real users' problems with known method fragments, in order to allow its comparison with existing methods.

References

[1] CBSE: Component-based software engineering, 13th international symposium, Prague, Czech Republic (June 2010)

[2] Bergenti, F., Gleizes, M., Zambonelli, F.: Methodologies and Software Engineering for Agent Systems: The Agent-oriented Software Engineering Handbook. Kluwer Academic Pub. (2004)

[3] Morandini, M., Migeon, F., Gleizes, M.P., Maurel, C., Penserini, L., Perini, A.: A goal-oriented approach for modelling self-organising MAS. In: Aldewereld, H., Dignum, V., Picard, G. (eds.) ESAW 2009. LNCS, vol. 5881, pp. 33–48. Springer, Heidelberg (2009)

[4] 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, NY (2005) ISBN 1-59140-581-5

[5] Pavòn, J., Gòmez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. In: Agent Oriented Methodologies, pp. 236–276.

[6] Cossentino, M.: From requirements to code with the PASSI methodology. In: Agent Oriented Methodologies, pp. 79–106.

[7] Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Tropos: An agent-oriented software development methodology. Autonomous Agent and Multi-Agent Systems 8(3), 203–236 (2004)

[8] Cossentino, M., Gaglio, S., Garro, A., Seidita, V.: Method fragments for agent design methodologies: from standardisation to research. International Journal of Agent Oriented Software Engineering 1(1), 91–121 (2007)
[9] Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., Russo, W.: A simulation-based process for the development of multi-agent systems. International Journal on Agent Oriented Software Engineering, IJAOSE (2008)
[10] Jacobson, I., Booch, G., Rumbaugh, J.: The unified software development process. Addison-Wesley Longman Publishing Co., Inc., Boston (1999)
[11] Capera, D., Georg, J.P., Gleizes, M.P., Glize, P.: The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In: International Workshop on Theory And Practice of Open Computational Systems (TAPOCS at IEEE 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2003) (TAPOCS), Linz, Austria, June 9-11, pp. 389–394. IEEE Computer Society (2003), http://www.computer.org
[12] Bonjean, N., Chella, A., Cossentino, M., Gleizes, M.P., Migeon, F., Seidita, V.: Metamodel-Based Metrics for Agent-Oriented Methodologies (regular paper). In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), June 4-6, Valencia (2012)
[13] OMG: Software Process Engineering Metamodel. Version 2.0. Object Management Group (March 2007)
[14] Seidita, V., Cossentino, M., Galland, S., Gaud, N., Hilaire, V., Koukam, A., Gaglio, S.: The metamodel: a starting point for design processes construction. International Journal of Software Engineering and Knowledge Engineering 20(4), 575–608 (2010)
[15] Puviani, M., Cossentino, M., Cabri, G., Molesini, A.: Building an agent methodology from fragments: the mensa experience. In: SAC, pp. 920–927 (2010)
[16] Firesmith, D., Henderson-Sellers, B.: The OPEN Process Framework: An Introduction. Addison-Wesley (2002)
[17] 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 (LNAI), vol. 3690, pp. 142–152. Springer, Heidelberg (2005)
[18] Cossentino, M., Sabatucci, L., Seidita, V.: A collaborative tool for designing and enacting design processes. In: Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M. (eds.) 24th Annual ACM Symposium on Applied Computing (SAC 2009), vol. 2, pp. 715–721. ACM, Honolulu (December 8, 2009)