A Self-organizing Multi-Agent System for Combining Method Fragments

Noélie Bonjean, Marie-Pierre Gleizes, Christine Maurel and Frédéric Migeon
Institut de Recherche en Informatique de Toulouse (IRIT), Université de Toulouse, Toulouse, France

Keywords: Method Fragments, Adaptive Multi-Agent System, Method Process.

Abstract: Software systems are becoming more and more complex. A common dilemma faced by software engineers in building complex systems is the lack of method adaptability. However, existing agent-based methodologies and tools are developed for particular system and are not tailored for new problems. This paper proposes an architecture of a new tool based on SME for self-constructing customized method processes. Our approach is based on two pillars: the process fragment and the MAS meta-model. These two elements are both defined and considered under a specific agent-oriented perspective thus creating a peculiar approach. Our work is based on the self-organization of agents, making it especially suited to deal with highly dynamic systems such as the design of an interactive and adaptive software engineering process.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The MAS community has been prolific to define software engineering methods (Bergenti et al., 2004; Henderson-Sellers and Giorgini, 2005) in order to guide designers with respect to the wide range of MAS properties. Facing the numerous methods, a development team needs help to execute the relevant process according to the development context which is defined by the system under study characteristics as well as the team capabilities and preferences. Our goal is to provide new tools for designing complex systems where the method must be adapted to the development context.

Coming from Situational Method Engineering research (Brinkkemper et al., 1998; Henderson-Sellers and Ralyté, 2010), the aim of decomposing processes into pieces is to adapt the process to the characteristics of the business problem and to the level of expertise of engineer teams (Ralyté, 2004). A process can then be defined by assembling the pieces of methods, called fragments, in order to suit the context (the situation) changes. The Agent-Oriented Software Engineering (AOSE) community contributed to this research splitting up methods into fragments and providing precise descriptions of them (INGENIAS[1], PASSI[2], ADELFE (Bernon et al., 2005), TROPOS[3]...).

We present in section 2 the aim of the SCoRe model (Self-Combined method fRagments) that we propose and its architecture in section 3 to automatically build a self-adaptive design process where each fragment is encapsulated in an autonomous agent. This approach relies on the self-organization of its agents, making it especially suited to deal with highly dynamic systems such as the design of an interactive and adaptive Software Engineering Process (SEP). Section 4 briefly presents tests and explains the results obtained. Finally, section 5 describes related works before concluding.

[1]http://ingenias.sourceforge.net/
[2]http://www.pa.icar.cnr.it/passi/Passi/PassiIndex.html
[3]http://www.troposproject.org/

2 REQUIREMENTS AND CHARACTERISTICS OF SCoRe

The contribution of the work lies in the self-adaptive multi-agent system implementation for self-composition and self-organization of method fragments. This section presents the challenges.

2.1 Adaptation

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 e.g. (Cossentino et al., 2008). 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

Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

now method adjustments to the specific requirements and constraints are mixed in ”local” adaptations and modifications of existing one. In order to succeed in creating good situational methods, i.e., methods that best fit given situations, fragment representation and cataloguing are very important activities. In particular, the fragments (sometimes addressed as process fragments, method fragments or chunks) have to be represented in an uniform way that includes all the necessary information that may influence their retrieval and assembling.

2.2 Fragment Standardisation

Method fragments are first identified by examining existing methods. These method fragments are made according to templates defined by repository designers and standardisation working groups [4]. Therefore the choice of fragments granularity relies on designers. According to the Rational Unified Process, methods are defined following different levels of granularity (phase, activity and step) which is an important factor. 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. This low level of granularity is useless and inaccurate when the steps are related and interdependent. 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. By consequences, an activity or step may be included in different fragments. The risk that this happens grows up with the level of granularity. In addition, the assembling possibilities are therefore minimized.

2.3 Complexity

Currently, ten AOSE methods are fragmented, each one composed of approximately twenty fragments. Such fragments constitute the root constructs of the method 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 method (for instance, a diagram or a set of diagrams of the same type). Following this assumption, fragments obtained from different methods 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 to design the system SCoRe to self-combine and self-organize fragments.

[4]http://www.pa.icar.cnr.it/cossentino/fipa-dpdf-wg/

2.4 User Requirements

In our approach, a new complete process is self-designed contingent on a situation. A complete process enables engineers to visualise all activities and to have a whole view of the process. SCoRe focuses on adaptation during process execution. At every step, the development team is advised by SCoRe 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 is based on the fragments agentification in order to self-design an adaptive process which can deal with the complexity mainly due to the huge number of fragments. Indeed, a complex system cannot currently be designed without bug caused by designers. Assisting 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 for components assembling, the fragments combination needs features. In our approach, they correspond to MAS Metamodel Elements (MMME). Two fragments are therefore combined if one produces the MMME required by another one.

3 ARCHITECTURE OF SCoRe

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

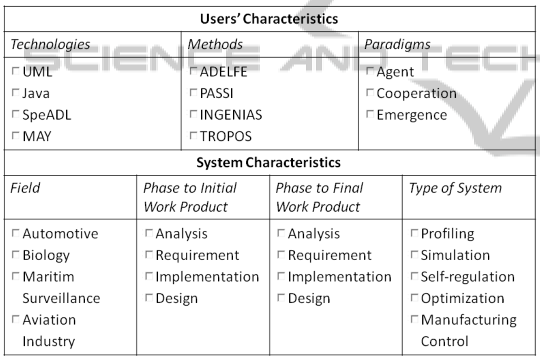
3.1 SCoRe Components

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 combining some of the fragments taking into account the user-defined objectives and knowledge.

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 acts without relying on a model of the processes,i.e. it is only able to take into account the designers' knowledge and objectives, and to observe the evolution of the running process on which MMMEs are available, in order to decide which fragments to add. The best possible running process is therefore designing according to a context.

3.1.1 Users' Objectives and Knowledge

In order to design a tailored process, SCoRe requires some information about the wanted system and about the users. Actually, these informations enable to select fragments which make up the suggested process. On the one hand, the user has to give his/her knowledge about the known technologies, methods and paradigms. On the other hand, the user has to describe briefly the intended system by defining the field of application, the phase corresponding to the initial and final work product and the type of system. Figure 1 shows an example of the both kind of characteristics but it is not an exhaustive list.



Users' Characteristics		
Technologies	*Methods*	*Paradigms*
☐ UML ☐ Java ☐ SpeADL ☐ MAY	☐ ADELFE ☐ PASSI ☐ INGENIAS ☐ TROPOS	☐ Agent ☐ Cooperation ☐ Emergence

System Characteristics			
Field	*Phase to Initial Work Product*	*Phase to Final Work Product*	*Type of System*
☐ Automotive ☐ Biology ☐ Maritim Surveillance ☐ Aviation Industry	☐ Analysis ☐ Requirement ☐ Implementation ☐ Design	☐ Analysis ☐ Requirement ☐ Implementation ☐ Design	☐ Profiling ☐ Simulation ☐ Self-regulation ☐ Optimization ☐ Manufacturing Control

Figure 1: Example of characteristics.

3.1.2 Context

The context is a set of elements external to the activity of an entity. It describes the environment wherein the entity evolves. Moreover, the context has an influence on the process of fragments selection.

In processes under construction, the context is made up of users' objectives and knowledge, available fragments and the elements included in the running process.

3.1.3 Agents

SCoRe is composed of four distinct kinds of agents, following a perception-decision-action lifecycle, which cooperate according to the adaptive multi-agent systems theory described in (Capera et al., 2003). The basic idea underlying this cooperation consists, for every agent, in always trying to help the agent which encounters the most critical situation from its own point of view.

Figure 2 gives the structure of SCoRe designing a method process. The different types of agents involved are shown, as well as the links modelling the existing interactions between them. Actually, SCoRe is made up of the following agents:

- **MAS Metamodel Element (MMME):** required or produced by a Running Fragment, its aim is to choose which fragment it will be linked to. A MMME is connected with all the waiting fragment and the Running Fragments which are included in the running process and which produce or consume it.
- **Waiting Fragment (WF):** its purpose is to integrate its instances (running fragment) in a process once it is in an adequate situation. They are linked to the MMMEs and a set of context.
- **Running Fragment (RF):** it aims at finding its localization inside the running process. A RF is linked with the MMMEs encompassed in the running process that it produces or requires.
- **Context (C):** related to a fragment, it aims at evaluating its relevance according to the MMMEs already involved in the running process and the users' objective and knowledge. The context agent is related to a fragment for which it evaluates its relevance to be added in the running process.

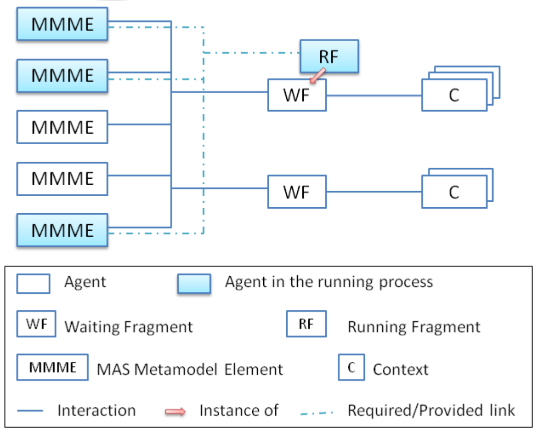


Figure 2: Example of agents and their relationships in SCoRe.

3.2 General Behavior of SCoRe

3.2.1 Prerequisite

Some prerequisites are necessary for the execution of SCoRe. Actually, in order to help the fragments selection in SCoRe, the user has to fill in two forms about its objectives and its knowledge. Firstly, the user completes its knowledge that will be mainly used

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

to select fragments. Secondly, the user characterizes the wished system. From the ticked fields, the initial and final MMMEs are extracted. Actually, according to user's knowledge and the initial and final phase selected, MMMEs are included in the running process by SCoRe.

3.2.2 Starting of the Running Process Construction

When the user provides the prerequisite items, Waiting Fragment agents and their Context agents are created. They know each other; however the Waiting Fragment don't know the others Waiting Fragment, and their associated Context don't know each others.

Besides, the MMMEs corresponding to users' problem are created with the acquaintance of the available Waiting Fragments. They are then included in the running process. The construction of the running process starts therefore with the MMMEs. These MMMEs are said *initial* and *final*. While a MMME aims at being linked to at least two fragments, i.e. one which produces it and one which consumes it, an *initial*, respectively *final*, MMME aims at being linked to at least one fragment which consumes it, respectively produces it. The *initial* and *final* MMME start the running process construction by interacting with all the waiting fragments.

3.2.3 Running Process Construction

As it is shown in figure 2, an agent communicates as follow. The first agent interactions correspond to the *initial* and *final* MMME looking for a fragment. When a waiting fragment receives a message from a MMME agent, it solicits their own contexts. The context self-evaluates itself. Then it answers by giving its relevance to the waiting fragment which sends it to the MMME in turn. According to the different answers, one of the requesting MMME selects a waiting fragment. The selected waiting fragment is then ready to create a running fragment. The created running fragment is added in the running process. Being related to input and output MMMEs, when a running fragment is added in the running process, it links itself to the MMMEs already present in the running process. If one of its MMMEs is missing, the running fragment creates it. Then, the created MMME agent is added in the running process. The MMME agents request the waiting fragment agents until to be satisfied.

Next sections will provide a more detailed description of these agents behaviors and interactions.

3.3 General Behavior of the Agents

An agent that intervenes in an AMAS is composed of different parts that produce its behavior: skills, aptitudes, the interaction language, world representations, Non Cooperative Situations, and criticality and/or confidence.

For an agent, criticality represents the degree of non-satisfaction of its own goal. It enables an agent to determine the relative difficulty of agents in its neighborhood. Evaluation methods and calculation of the criticality are specific to each type of agent. The confidence of an agent is an internal measure that provides information on the reliability of the decision on actions intended.

These notions guide the SCoRe agents behavior and will be presented in the following subsections.

3.3.1 The MMME Agents

The MMME agents represent the links between running fragment agents. Their individual goal is to be incorporated in the running process. The MMMEs behavior 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 respectively produces or consumes it. In this state, it is looking for a relevant fragment to which it can be linked by requesting all waiting fragments. The relevant waiting fragments answer it by giving their confidence. The confidence of the waiting fragment agent provides information on its reliability proposition. The most relevant fragment will be chosen by the MMME agent for being a potential fragment to be added in the running process and the MMME agent updates its confidence.

The MMME agents evaluate their own criticality and cooperate to choose the most relevant fragment according to the criticality of the ones suggested.

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

3.3.2 The Waiting Fragment Agents

Each waiting fragment agent has an associated set of context agents. Their goal is to be integrated in a process once it is in an adequate situation. For that, when a waiting fragment agent receives requests from MMMEs which are looking for a fragment, it forwards the request to its context agents, if the waiting fragment agent considers itself as a potential solution. A waiting fragment agent considers itself to be solution if the requesting MMME belongs to its own required or provided MMME. Then, the waiting fragment agent waits the answer from its context agents. It updates its confidence and sends it to the MMME.

Moreover, a waiting fragment agent can be selected to be inserted in the running process. Informed by the MMME agent, the waiting fragment agent creates a running fragment agent and sends the information to its context agents as a feedback.

3.3.3 The Running Fragment Agents

A running fragment agent is created by a waiting fragment agent and is introduced on time in the process. Its aim is to be incorporated in the running process. Its behavior changes according to its current state and its 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 its 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. Moreover, when the running fragment agent is inserted in the running process, it adds the required or produced MMMEs which are missing in the running process.

3.3.4 The Context Agents

The goal of the context agents is to represent a situation leading to a specific method process. They 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. Besides, a context agent possesses a set of forecasts, which describes the impact of the action proposed on the satisfaction of the both user and system characteristics. Moreover, a context agent possesses a set of metrics, which describes the impact of the action proposed on the running process (Bonjean et al., 2012). Those input values are modified during the life of a context agent. According to its behavior, from different feedback that it receives, a context agent adjusts its confidence.

Finally, the behavior 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 it 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.

4 RESULTS AND ANALYSIS

The conducted tests focus on the functional adequacy and the dynamic adaptation to specific situation. The first test has been carried out with known method processes to show the correctness of SCoRe. We show that Score enables to compose a known method back from its fragments. Concerning adaptability test, we conducted them with fictive processes. Combining known methods are very complex task. Until now, the inter-operability and semantic matching of fragments from different known methods stay an important challenge. Actually, in this problem, some works base on standardisation of fragments notion and of their description. For this reason, we simulate methods. Therefore we used fictive processes which have been simplified. Actually, they are defined in the following way. They are composed by four fragments while a known method is made up of approximately twenty fragments. These tests show how SCoRe is able to adapt itself: on the one hand, how SCoRe adapts itself to a context and on the other hand, how SCoRe adapts itself to design a new tailored method process.

5 RELATED WORKS

Apart from application field, several recent works exploit the lessons of adaptive self-organizing natural and social system to enforce self-awareness, self-adaptability, and self-management features in distributed system.

Some approaches of components agentification have been developed. They aim at allocating components to agent properties such as autonomy and interaction. (Hara et al., 2000) proposes an extension of ContractNet protocol for finding components in libraries. A function which deals with request as message is added in the components stored in libraries as an agent. The agents have knowledge about specificity of the component and its ability to answer a need. Our proposition has the same goal with a view to distribute fragments research and their adaptation and their composition. This allocation of reuse process enables some assembling strategies for instance.

Besides, Web Services represent today's reference standard technology for the set up of distributed systems that need to support machine-to-machine interaction among heterogeneous applications distributed over a network. The automatic composition and adaptation of services has been explored using a variety of AI planing engines (Rao and Su, 2005).

In (Thomas et al., 2009), a set of workflow fragments are composed in ad hoc wireless mobile environments. This approach designs dynamically construction of custom, context-specific workflows in response to unpredictable and evolving circumstances by exploiting the knowledge and services available within a given context. For that, a graph made up of all workflow fragments is built up before exploring and pruning it. As presented approaches, ours is based on current data base of fragments and on MAS metamodel elements. The way to integrate the method fragment in the process is different because in running development, our approach can take into account process adaptation according to development context.

6 CONCLUSIONS AND FUTURE WORKS

This paper presents SCoRe architecture, an adaptive multi-agent system, which designs a tailored process by combining fragments together. Each agent composing the adaptive multi-agent systems follows a local and cooperative behavior, driven by the use of their confidence. The four different kinds of agents, composing the SCoRe system, were defined in order to self-design and self-combine a tailored method process without relying on the method engineer. The resulting behavior of the SCoRe system is the ability to design process and adjust the proposed process according to the characteristics of application domain and users profile. This first prototype allowed to enhance our experience on practical problems such as metamodel compatibility, parameters composition or fragments adaptation to specific field.

However, there is still room from improvements for incoming interoperability and for evaluating the designed process.

REFERENCES

Bergenti, F., Gleizes, M., and Zambonelli, F. (2004). *Methodologies And Software Engineering For Agent Systems: The Agent-oriented Software Engineering Handbook.*

Bernon, C., Camps, V., Gleizes, M.-P., and Picard, G. (2005). Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology . In *Agent-Oriented Methodologies* , pages 172–202.

Bonjean, N., Chella, A., Cossentino, M., Gleizes, M.-P., Migeon, F., and Seidita, V. (2012). Metamodel-Based Metrics for Agent-Oriented Methodologies. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Brinkkemper, S., Saeki, M., and Harmsen, F. (1998). Assembly Techniques for Method Engineering.

Capera, D., Georgé, J.-P., Gleizes, M.-P., and Glize, P. (2003). The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In *International Workshop on Theory And Practice of Open Computational Systems*, pages 389–394.

Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., and Russo, W. (2008). Passim: A simulation-based process for the development of multi-agent systems. *International Journal on Agent Oriented Software Engineering (IJAOSE)*.

Hara, H., Fujita, S., and Sugawara, K. (2000). Reusable software components based on an agent model. In *Proceedings of the Seventh International Conference on Parallel and Distributed Systems: Workshops*, ICPADS '00.

Henderson-Sellers, B. and Giorgini, P. (2005). *Agent-oriented methodologies*.

Henderson-Sellers, B. and Ralyté, J. (2010). Situational method engineering: State-of-the-art review. *J. UCS*, pages 424–478.

Ralyté, J. (2004). Towards situational methods for information systems development: Engineering reusable method chunks. *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education*, pages 271–282.

Rao, J. and Su, X. (2005). A Survey of Automated Web Service Composition Methods. pages 43–54.

Thomas, L., Wilson, J., Roman, G.-C., and Gill, C. (2009). Achieving coordination through dynamic construction of open workflows. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09.