Defining Agile and Planned Method Fragments for Situational Method Engineering

Guilherme Vaz Pereira, Lisandra Manzoni Fontoura

Laboratório de Computação Aplicada – Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brasil

{guigavazpereira,lisandramf}@gmail.com

Abstract. *Situational Method Engineering (SME) aims to build a specific software development method/process according to a project situation from reusable building blocks called method fragments. This paper proposes a metamodel for definition of method fragments. The proposed metamodel was elaborated from the integration between the RUP and ISO/IEC 24744 metamodels. It allows the fragment definition according to practices recommended by agile and planned methodologies.*

1. Introduction

There is no unique software process model appropriate to all projects. Each project requires particular actions [Alegría 2011] thus a specific process according to the requirements of each project is desirable.

Situational Method Engineering (SME) approach focuses on the project specific method/process building according to the situation at hand. This happens from reusable method fragments stored in a repository called method base. These can be extracted, e.g., from best practices and process models [Henderson-Sellers 2010].

The situation assists on determining which fragments are appropriate according to a project [Henderson-Sellers 2010]. It is described by project features and in this paper is represented by the Octopus Model [Kruchten 2010].

Some issues about SME are about how to create fragments and how to formalize, store and retrieve them [Henderson-Sellers 2010]. This paper proposes the M^2F – Metamodel for Method Fragments to define method fragments for SME based approaches.

This paper is organized as follows: Section 2 presents the concepts of method fragments. Section 3 describes the proposed M^2F metamodel. Section 4 presents examples of applying M^2F metamodel. Section 5 and 6 describe related works, and conclusions and future works, respectively.

2. Method Fragments

The notion of method fragments used in this paper is from [Henderson-Sellers 2008]. It is widely supposed that a method fragment is an element generated from a metamodel.

There is a distinction between *process-fragments* (e.g., an activity) and *product-fragments* (e.g., an artifact) thus must be an association between process and product fragments to capture appropriate dependences between them. Furthermore, the

information about the fragments use-situations is distributed in classes of the metamodel [Henderson-Sellers 2008].

The *process fragments* are focus in this paper. Thus the term "method fragments" refers to this concept.

3. M²F – Metamodel for Method Fragments

M²F describes elements to express method fragments and their concepts and relationships. It is flexible enough to represent method fragments from agile, planned and hybrid approaches.

The proposed metamodel was developed from the integration of RUP metamodel [Bencomo 2005] and the ISO/IEC 24744 metamodel [ISO/IEC 2007] following guides to integrate conceptual schemes proposed by Batini (1992). The RUP metamodel describes the Rational Unified Process (RUP) [Rational 2003] elements. ISO/IEC 24744 is an international standard which defines a metamodel to defining methodologies for software development. Due to the limited space we are not able to describe details about these metamodels.

The goal of the M²F is to be as simple as possible to represent only the needed concepts to define method fragments. For this reason not all the classes and attributes from RUP and ISO/IEC 24744 metamodels are relevant for our purposes. The ISO/IEC 24744 metamodel is very abstract and complex because it aims to conceive a model in multiple domains (components as documented and application domain), and extensive (sixty-eight classes). This kind of representation is not necessary because the scope of this work is only related to the definition of method fragments and do not consider the application domain.

Figure 1 shows the M²F. We choose to keep the RUP nomenclature (version 7) [Shuja 2007] for classes with similar concepts, e.g., between "Artifact" and "WorkProduct" the first was chosen.

The "Action" and "TaskTechniqueMapping" classes are from ISO/IEC 24744. The "Discipline" and "LifeCycle" classes are from RUP. The "Artifact", "Phase" and "Worker" classes are from RUP and correspond to "WorkProduct", "Stage" and "WorkProducers" ISO/IEC 24744 classes, respectively.

In the M²F, the "Activity" class represents a method fragment (*process-fragment*) while the "Artifact" class represents a *product-fragment*.

In M²F, the classes from the ISO/IEC 24744 are semantically equivalent to the "Kind" classes (represent elements as documented in a methodology) even though they do not use this suffix, e.g., the M²F "Task" class corresponds to the ISO/IEC 24744 "TaskKind" class. This has been determined to favor the metamodel understanding through a nomenclature similar to acknowledged models such as SPEM [OMG 2011].

However the proposed metamodel uses concepts from the ISO/IEC 24744, e.g., "Action" class, and mechanisms for fragments retrieval according to a situation at hand through "TailoringGuide" class. This class represents the fragment use-situation, i.e., situations in which the method fragment can be used successfully. The "TailoringCriteria" attribute corresponds to one or more criteria to tailor a software process through a SME approach, e.g., the project risks, or quality or security

requirements of the project, etc. For example, if one of the tailoring criteria is the risks which the situational process should prevent thus the attribute "TailoringCriteria" should contain the risks which the fragment can avoid through its tasks.

The "InfoSituational" attribute represents the fragment use-context according to the Octopus Model. It is about the project context in which the fragment use is appropriated.

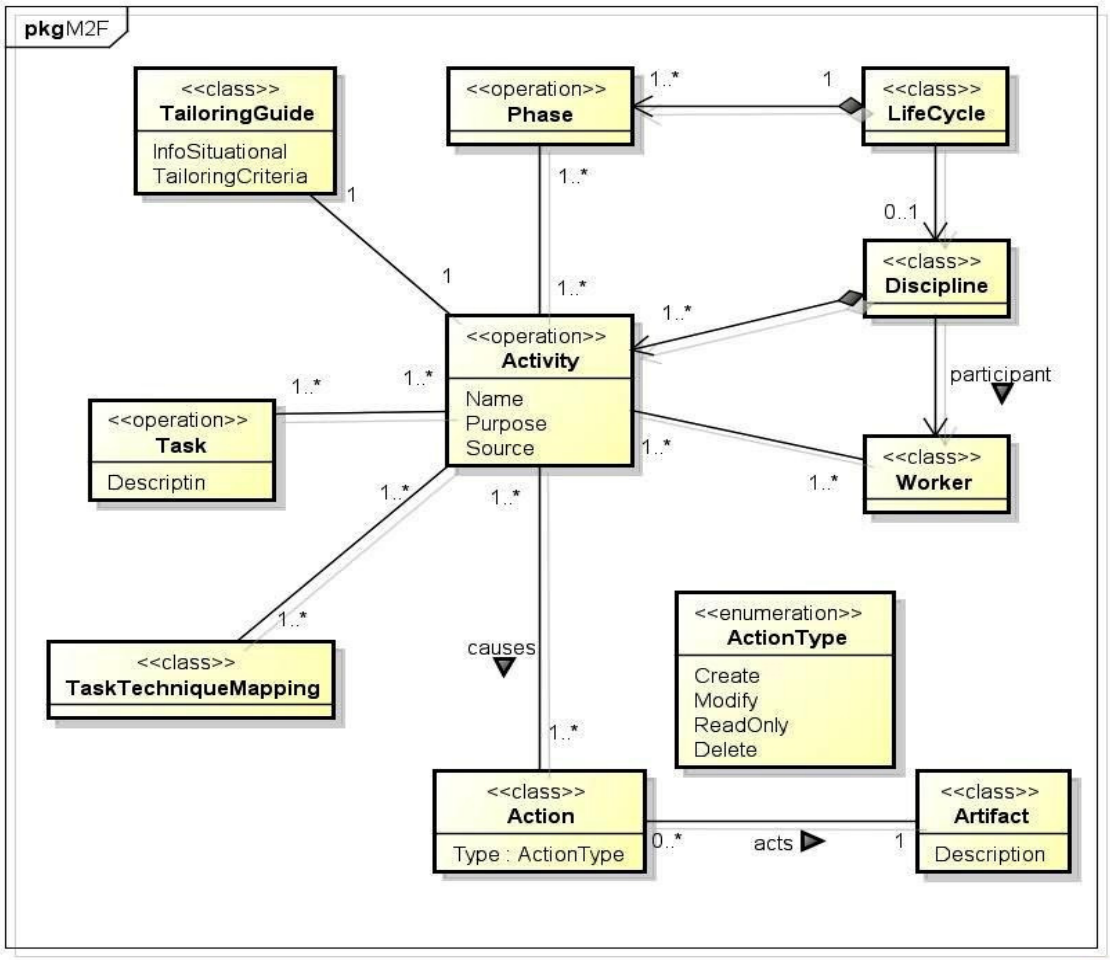

Figure 1. M²F – Metamodel for Method-Fragments.

The "TaskTechniqueMapping" class is about guides related to the fragment's tasks. The "Action" class corresponds to how a method fragment acts upon a *product-fragment*. Thus a artifact is not only input or output, e.g., it consider that the "x" task is related to the "y" artifact with a "create", "readOnly", "modify" or "delete" action.

3.1. TailoringGuide – Octopus Model

Octopus Model [Kruchten 2010] is a model for contextualizing the software development through eight factors which affect significantly the software development process.

It defines the "agile sweet spot" which means conditions where agile development practices they are most likely to succeed. So we consider that for situation where agile practices are not advised, practices from planned approaches are recommended.

The factors defined by the Octopus Model with possible values (some are proposed by Kruchten (2010) while others by us) are shown in Figure 2. The model allows defining the project context and the use-context of the method fragments through its criteria and obtain an agile, planned or hybrid context, i.e., if it is most appropriated to be used in a project with the agile, planned or hybrid characteristics.

Factors	***Possible Values***		
	Agile Characteristics → *Planned Characteristics*		
Size (in relation to team)	Small	Medium	Large
Stable Architecture	Stable	Changed	New
Business Model	In house – bespoke for a customer	Commercial	Large System Component
Team Distribution	Collocated	Different teams	Geographic
Rate of Change (% in a month)	More than 50	From 10 to 30	Less than 10
Age of System	New development	Maintenance	Legacy evolution.
Criticality	Comfort loss	Essential money loss	Deaths
Governance	Dynamic/flexible	Simple rules	Mechanic/formal.

Figure 2. Octopus Model factors and possible values.

4. DEFINING METHOD FRAGMENTS WITH M^{2}F

In this section method fragments are defined from M^{2}F. Figure 3 presents a method fragment example extracted from RUP [Rational 2003]. It has a planned use-context according to the Octopus Model.

The fragments can be defined or modified according to the organization's needs or experiences through the M^{2}F. They can be extracted, e.g., from process models, process patterns, reference models, best practices, etc.

Figure 3 shows a planned fragment. Through the values presented in the Figure 2 is possible to define agile and hybrid fragments use-context ("TailoringInfo"). For example, to define an agile use-context should be used values corresponding to agile characteristics (Figure 2).

Due to the limited space we are not present more examples here.

5. Related Works

There are other approaches to method fragments to be used in SME approaches. The metamodel of the FIPA [Seidita 2009] is based on SPEM. However it is specific for an application or technology domain, software development processes oriented to agents.

Agile method fragments are proposed in [Abad 2010] such elements are more related to the concept of method chunks and are limited to the agile methods context. Still, they do not present a mechanism for the definition of new fragments, as hereby proposed with the M^{2}F metamodel.

An approach to represent method components and tailor them onto a situational method is proposed in [Aharoni 2008]. It is based only on the ISO/IEC 24744 metamodel. As proposed in this paper, it includes information to retrieval method components. However, it limits to four attributes to characterize the project. Besides this, it uses a slightly known language to describe method components.

Fragment	Analyze the Problem	
Purpose	*The purpose of this process-fragment is to gain agreement on the problem being solved. Analysis of the problem involves identify the stakeholders…*	
Source	*RUP.*	
Discipline	*Requirements.*	
Phase(s)	*Inception; Elaboration.*	
Tasks	✓ *Capture a Common Vocabulary;* ✓ *Find Actors and Use Cases;* *etc.*	
Roles	✓ *System Analyst ;* ✓ *Customer;* ✓ *Others stakeholders.*	
Task-Technique Mapping	✓ *Brainstorming;* ✓ *Requirements Workshop;* *etc.*	
Action	✓ *Create "Requirements Management Plan";* ✓ *Read "Software Development Plan";* *etc.*	
TailoringGuide	***InfoTailoring:***	***Size:*** *large.* ***Stable Architecture:*** *new.* ***Business Model:*** *large system component.* ***Team Distribution:*** *geographic distribution.* ***Rate of Change:*** *less than 10.* ***Age of System:*** *legacy evolution.* ***Criticity:*** *money loss.* ***Governance:*** *mechanic/formal.*
	TailoringCriteria: *One or more tailoring criteria*	

Figure 3. Octopus Model factors and possible values.

6. Conclusions and Future Works

This paper is about to define and formalize method fragments through a metamodel called M^2F. It was developed through the RUP and ISO/IEC 24744 metamodels following the guide proposed by Batini (1992) to integrate them. In M^2F, the RUP nomenclature favors the understanding of it.

To favor the SME application the metamodel provides means to retrieve and tailoring fragments through the TailoringGuide class. Furthermore M^2F allows defining method fragments with agile, planned and hybrid use-contexts.

Using the proposed metamodel each organization can define method fragments to store in a method base according to yours specific needs or experiences from past projects.

In future works we plan develop a SME approach to prevent risks in software projects from this model of method fragments using a tool support to select fragments according project risks and put them in relevance order in relation to the project context.

Acknowledgements

We want to express our gratitude to CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the financial support.

References

Abad, Z. S. H., Sadi M. H. and Ramsin, R. (2010) "Towards Tool Support for Situational Engineering of Agile Methodologies", In: Proceedings of the 2010 Asia Pacific Software Engineering Conference (APSEC '10), IEEE Computer Society, USA, p. 326-335.

Aharoni, A. and Reinhartz-Berger, I. (2008) "A Domain Engineering Approach for Situational Method Engineering", In: Proceedings of the 27th International Conference on Conceptual Modeling (ER '08), Springer-Verlag, Berlin, p.455-468.

Alegría, J. A. H., Bastarrica, M. C., Quispe, A. and Ochoa, S. F. (2011) "An MDE approach to software process tailoring", In: Proceedings of the 2011 International Conference on Software and Systems Process (ICSSP '11), ACM, USA, p. 43-52.

Batini, C., CERI, S. and Navathe, S. B. (1992) Conceptual Database Design An Entity-Relationship Approach, The Benjamin/Cummings Publications, USA.

Bencomo, A. (2005) Extending the RUP, Part 1: Process Modeling. <http://www-128.ibm.com/developerworks/ rational/library/05/r-3320/>. Accessed March 2011.

Henderson-Sellers, B. and Ralyte, J. (2010) "Situational Method Engineering: State-of-the-Art Review", In: Journal of Universal Computer Science, p. 424-478.

Henderson-Sellers, B., Gonzalez-Perez, C., Ralyté, J. (2008) "Comparison of Method Chunks and Method Fragments for Situational Method Engineering", In: Proceedings 19th Australian Software Engineering Conference.IEEE Computer Society, USA, p. 479-488.

ISO/IEC (2007) Software Engineering - Metamodel for Development Methodologies International Organization for Standardization / International Electrotechnical Commission.

Kruchten, P. (2010) "Contextualizing Agile Software Development", In: Proceedings of the EuroSPI 2010 Conference, França, p. 1-12.

OMG - Object Management Group Inc. (2011) Software Process Engineering Metamodel (SPEM) 2.0.

Rational Software Corporation (2003) Rational Unified Process: Version 2003.06.12, USA.

Seidita, V. et al. (2009) "Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies", In: Agent-Oriented Software Engineering IX, Michael Luck and Jorge J. Gomez-Sanz (Eds.). Lecture Notes In Computer Science, Vol. 5386. Springer-Verlag, Berlin, Heidelberg, p. 46-59.

Shuja, A., Krebs, J. (2007) "RUP Reference and Certification Guide: Solution Designer (RUP)", IBM Pres, USA.