ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2011, Vol.40, No.1

AN APPROACH FOR THE METAMODEL OF THE FRAMEWORK FOR A PARTIAL AGILE METHOD ADAPTATION

Gytenis Mikulėnas, Rimantas Butleris, Lina Nemuraitė

Kaunas University of Technology, Department of Information Systems
Studentu St. 50-308a, LT-51368 Kaunas, Lithuania
e-mail: gytenis.mikulenas@ktu.lt, rimantas.butleris@ktu.lt, lina.nemuraite@ktu.lt

crossref http://dx.doi.org/10.5755/j01.itc.40.1.194

Abstract. Today, such information system development methods as Extreme Programming, Scrum, Dynamic Systems Development Method, Crystal family, Agile modeling, OpenUP and others are being positioned as proven alternatives to the more traditional plan-driven approaches. However, although there are a variety of agile methods to choose from, the formal methods for their partial adaptation and customization are lacking. The main aim of this paper is to present a metamodel of the framework for a partial agile method adaptation. The paper presents the process of metamodel construction from the concepts that are both direct and indirect solutions to the sub-problems of the partial agile method adaptation. The presented paper extends some of our earlier and more fragmented findings that have been described in our previous work.

Keywords: agile method, partial adaptation, framework, metamodel.

1. Introduction

Although only eight years have passed since the first publishing of the Agile Manifesto [4], the concept of Agile development has gained strong positions within the field of Information Systems Development (ISD). Such approaches as Extreme Programming (XP) [9], Scrum [35], Dynamic Systems Development Method (DSDM) [13], Crystal [14], Agile modeling [6], OpenUP [22] and others are now being positioned as proven alternatives to the more traditional plan-driven approaches [3, 10].

According to the latest CHAOS Report published by Standish Group, only about 32 % of software projects can be called successful, i.e. they reach their goals within a planned budget and on time [25]. Despite the availability of numerous new approaches, companies tend not to take drastic risks instantly switching from their methodological know-how to the agile methods. Instead, companies usually use their in-house know-how based ISD methods that are combinations of various methods and that evolved through the lifetime of the company [11, 32]. This happens because of the uniqueness of every ISD project and its environment. Despite the promises of the benefits from agile methods, practitioners are rarely faced with the need to adapt an entire agile method. Companies usually do not want to rebuild their methods and processes from scratch. Instead, the current demand is to extend their existing in-house methods by implementing some useful parts of certain agile methods [5]. The problem is that current agile methods are often presented as monolithic solutions without a formal roadmap how to customize and configure them for a partial adaptation [3]. Also, although there are a variety of agile methods to choose from, the formal methods for their adaptation and customization are lacking [30, 31, 37]. In order to resolve this problem, we present a framework for a partial agile method adaptation.

The paper builds on and extends some of the ideas presented in our previous research [27, 28]. The main aim of this paper is to present the metamodel of the framework for a partial agile method adaptation. The supplementary aims are to give an integral view of the problem and to extend the scope of the adaptation of agile methods.

The reminder of the paper is structured as follows. Section 2 reveals how the main problem was decomposed into more specific sub-problems. Section 3 presents the framework as an integral solution. Sections 4 and 5 describe the process of construction and the resulting metamodel. A case study of applying the framework is presented in Section 6, followed by the conclusions in Section 7.

2. The decomposition of a partial agile method adaptation problem

Majority of researchers of agile method adaptation and customization concentrate on presenting success stories or lessons learned by organizations that have partly adopted agile methodologies for specific

projects [15, 18, 23, 24]. Others propose adopting individual Agile practices only [5, 21]. Finally, there is a group of considerably more practical approaches [8, 12, 27, 28, 31] that could be described as techniques for the partial agile method adaptation. The problem with these approaches is that each of them proposes a solution for different aspects of the partial agile method adaptation and does not cover the whole problem. For example, Cockburn proposes to choose agile methods according to the number of people involved and criticality criteria when selecting a project's methodology [12] but only from the Crystal family methods, with no formal roadmap on how actual decision making could be done. Attarzadeh and Hock distinguish a dipole between Agile and traditional plan-oriented methods with some directions for method selection criterions [8] but without a guide for selecting a method and assessing its suitability. Though Mirakhorli et al. propose how to tailor an XP method for partial adaptation [31], their idea is based only on the informal brainstorming and expert judgment decision making techniques while being tailored for XP method only. Breaking and structuring agile methods is a promising direction but there is a need for defining the concepts and metamodel for this purpose as well as a process guide.

Although we have addressed these issues to some extent in our previous researches [27, 28] the further analysis revealed the need for decomposing the problem. Therefore, we propose viewing the problem of the partial agile method adaptation as a generalization of six sub-problems (Figure 1). Such a decomposition model serves as the basis for the possible solution that is discussed in the remainder of this paper. It could also be used by other researchers or practitioners willing to address the problem of partial agile method adaptation. The summarized results of the analysis of these sub-problems are presented in Table 1.

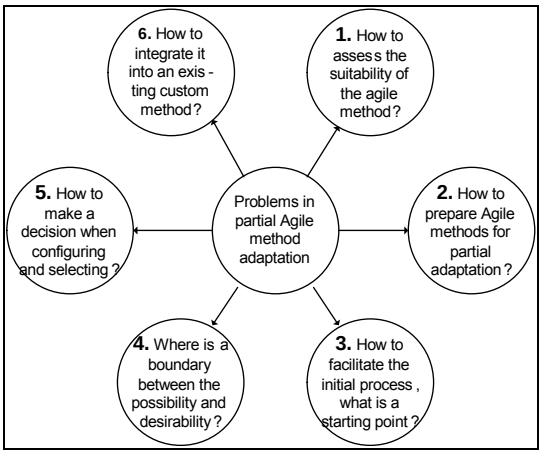

Figure 1. The decomposition of a partial agile method adaptation problem

Table 1. The sub-problems of a partial agile method adaptation problem

Sub-problem	Description
1. How to asses the suitability of the agile method?	Agile methods are specific methods often being positioned as alternatives to the more traditional plan-driven methods. There are both organizational and project level restrictions and requirements that must be met in order to succeed using agile methods at IT company. There is a need for a formal way to make an assessment of agile method suitability.
2. How to prepare agile methods for partial adaptation?	Companies usually have their own know-how and do not want to rebuild processes from scratch. Current demand is to extend these existing methods by supplementing them with some useful parts of certain agile methods. Existing agile methods are often presented as monolithic solutions without a formal roadmap how to combine and configure the methods for their partial adaptation. Therefore, there is a need for an approach for breaking down agile methods into a set of elements for their partial adaptation.
3. How to facilitate the initial process, what is the starting point?	IT market is very dynamic. Any additional method upgrade is a very costly activity for an IT company due to the risk and efforts for their personnel training. In the world of non-stop competition any approach needs to be presented in a way that facilitates and accelerates the process of learning and adapting. That is why a need for a guiding wizard arise, especially during the initial steps.
4. Where is a boundary between the possibility and desirability?	It is a well-known fact that ISD methods are not used as they ought to be in an actual ISD development projects. The same is true for elements (practices, techniques, etc.) of these methods. Done in ad-hoc manner this leads to a number of various modifications. There is a need to define the levels for the adaptation of such method elements when balancing between the possibility and desirability.
5. How to make the decisions when configuring and selecting?	The problem of a partial agile method adaptation brings forward a related question of how to select and construct fragments from the concrete agile methods. Moreover, this construction is a customization of agile methods, it uses decision making when selecting appropriate elements for the fragment. In most Agile adaptation approaches, researchers propose to perform the appropriate decisions using expert judgment or brainstorming techniques. In our opinion, the actual user is left on his own in such cases. There is a need for a guidance during the customization of the agile method.
6. How to integrate it into the existing in-house method?	Companies usually have their own know-how and do not want to rebuild processes from scratch. Instead, the current demand is to extend these in-house methods by involving some useful parts of certain agile methods. After customization and construction of the fragment of the agile method, the next step is its implementation. There is a need for the process guidance for making this implementation.

The sub-problems in Figure 1 reveal a wider picture surrounding the partial agile method adaptation at the same time offering some outlook into a possible solution. It becomes clear that due to the scope of this problem, coming up with just an informal technique would not be enough.

3. The framework for a partial agile method adaptation

Existing agile methods are often presented as monolithic solutions without a formal roadmap of how to configure a method for a partial adaptation. The basic definition of the partial agile method adaptation implies that agile methods must be broken down into a set of elements. An implementation of a subset of these elements is a partial implementation of an agile method. Aiming to provide the comprehensive solution for the partial adaptation, we have proposed general guidelines and concepts for building such a framework [28]. During the further research of the analysis results, the scope of our framework has been extended and classes for the metamodel were defined. The illustration of applying the framework is presented in Figure 2.

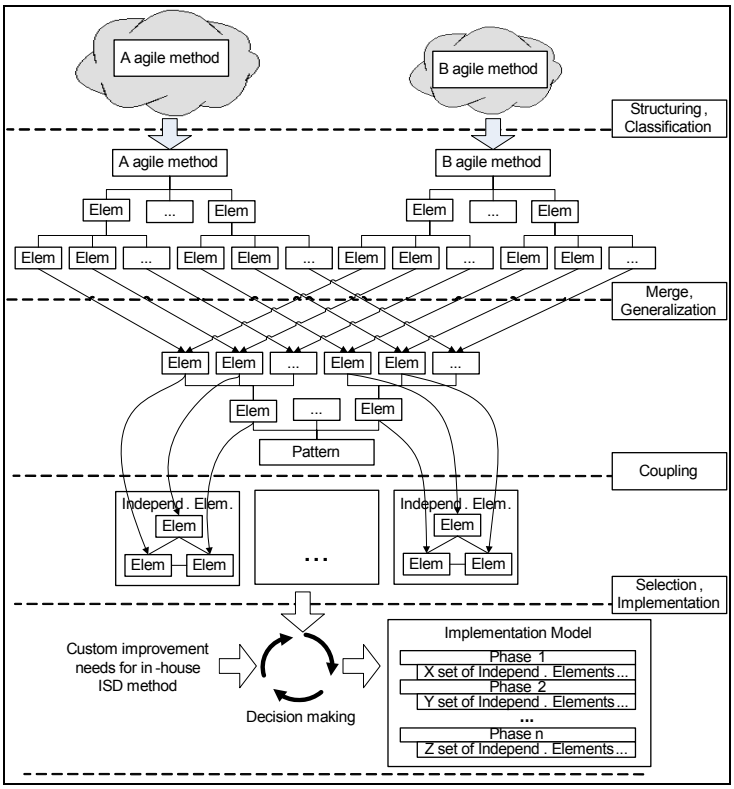

Figure 2. The illustration for applying the framework

Different agile methods can be decomposed into elements by using a common structure that is a part of the metamodel of the framework for a partial agile method adaptation. This common structure makes it possible to merge and generalize similar elements from different agile methods. This brings up the possibility to combine the implementations of different agile methods by applying patterns – similarly as in other areas of software engineering, e.g. in generating a program code [1, 2]. This process should be performed by an agile method engineer who can create different patterns by using decomposed elements. The pattern can be defined as a variant of some unified agile method that combines elements from different agile methods. Coupling is the process of defining new independent elements from those elements that are closely related to each other. It gives the end user the possibility to facilitate the element selection and composition when creating a partial agile method implementation model. Finally, those independent elements are prioritized according to the appropriate criterions using the formal decision making technique AHP (Analytic Hierarchy Process). Those prioritized independent elements are used for creating partial implementation of agile methods. The implementation model can be described as a plan for partial implementation. The process of using proposed framework has been divided into 3 tiers (Figure 3).

Each tier represents a process performed by an agile method engineer or the end user. The first tier of the framework is about managing the structure of the metamodel. On the second tier, patterns are created from the structured agile methods. The pattern consists of elements that are derived by the process of structuring agile methods. The structuring of the metamodel and creating a pattern should be performed by agile method engineers because they require the

appropriate knowledge and skills. Finally, the usage of such patterns is performed by the end user who may have any role in the project. Most of the method improvements at IT companies are initiated by managers but most of the implementations of new practices and techniques are performed by enthusiasts. That is why the usage of the tier 3 is wide opened.

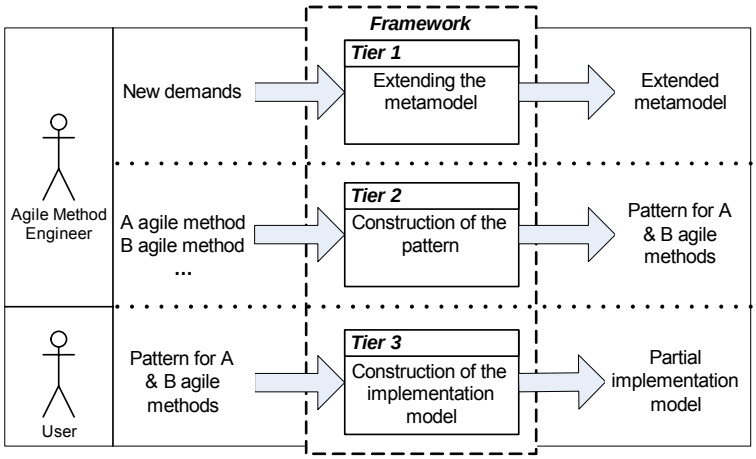

Figure 3. The conceptual usage model of the framework

4. Construction of the metamodel for the framework

The framework of the partial agile method adaptation requires a metamodel. This metamodel should serve as the basis for creating and using patterns during the partial agile method implementation. The framework is an integral solution and covers all the sub-problems defined in Section 2. By using the term "concept", we define a concrete direct or indirect solution for the related sub-problem defined in Section 2. Both groups of concepts are presented in Tables 2 and 3.

Table 2. The concepts that are directly related to the sub-problems

Direct concept	Related sub-problem
Agility requirements	How to asses the suitability of the agile method?
Method decomposition	How to prepare agile methods for partial adaptation?
Application areas	How to facilitate the initial process, what is the starting point?
Element levels	Where is a boundary between the possibility and desirability?
Criterions and prioritization	How to make the decisions when configuring and selecting?
Implementation model	How to integrate it into the existing in-house method?

Table 3. The concepts that are indirectly related to the sub-problems

Indirect concept	Related problem
Abstraction levels	How to split the metamodel into parts?
Structuring	How to prepare a new agile method using the common structure?
Merge and generalization	How to combine the elements from the different agile methods?
Coupling	How to facilitate the process of element selection?
Pattern	How to allow creating multiple preparations of agile methods?
Extensible metamodel	How to create a flexible structure of the metamodel?

The need for direct concepts is described in our previous research [28]. Indirect concepts were defined during the construction of the metamodel (Section 4.2). In this research, both direct and indirect concepts were scrutinized. We also used classes from existing metamodels from the domain of Situational Method Engineering. As a result of this concept development, we have got the classes that compose the proposed metamodel of the framework of a partial agile method adaptation. The initial results of deriving classes from existing metamodels are presented in Section 4.1. The final results (including proposed concepts and derived classes) are presented in Section 4.2.

4.1. Classes derived from existing metamodels

The concept of "Method decomposition" is the key subject of research in the field of Situational Method Engineering (SME) [19]. There are three most used standard metamodels in this domain. They are Open Process Framework (OPF) [16, 34],

Software Process Engineering Metamodel (SPEM) [33] and Software Engineering Metamodel for Development Methodologies ISO/IEC 24744 [20]. Each of them has both overlapping and unique parts. The approach for the development of the concept "Method decomposition" was to define the core common and unique classes of those three metamodels that should be used in the proposed metamodel. The results are shown in Figure 4.

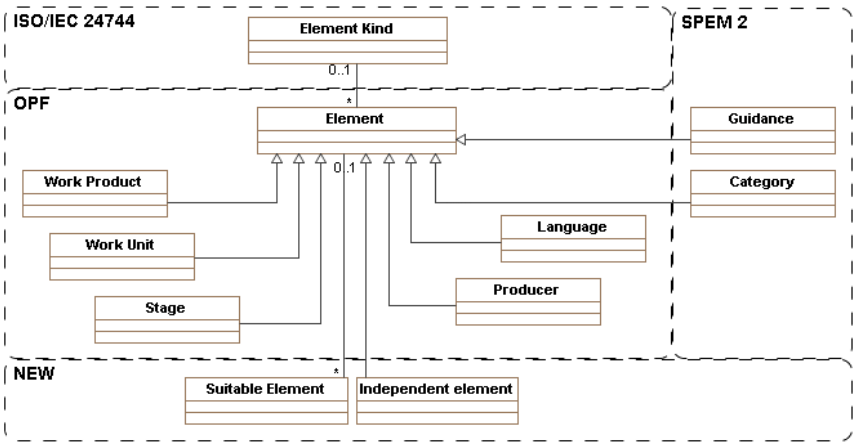

Figure 4. The combination of used core element classes

Any ISD method, including agile methods, could be decomposed into a set of related elements. The main classes of OPF, ISO/IEC 24744 and SPEM metamodels have the same purposes but their names are different. We decided to use the names of the most common classes from the OPF metamodel. "Work Product" is a kind of method element that defines anything valuable that is produced by the "Producer" performing the "Work unit" activities during the "Stage" process. The "Language" is used when the "Work product" is a code. Any additional information that is related to an element is described as the "Guidance". The specific parts of agile methods (philosophy, values, etc.) that do not fit to any of these classes may be set to a kind of "Category". A group of closely related elements should be related to a kind of "Independent element". In addition, we used a separation of the element from its kind according ISO/IEC 24744 [20] and added the "Suitable element" class that represents the usage of an element in the lower level.

4.2. Proposed concepts and classes

Now we will describe the proposed concepts that are directly or indirectly related to the sub-problems defined in Section 3. Each concept was used deriving the classes that are needed to implement that concept. Notice that three different shading styles are used to denote the tier the particular class belongs to.

Abstraction levels. The agile methods can be structured into a set of elements using the predefined structure of classification (element kinds and their relation kinds). The classes that reflect the structured agile methods are defined in the second tier. Hence, using the predefined structure, it is possible to create a plenty of patterns from structured agile methods, and a plenty of partial implementation models using these patterns (Figure 5).

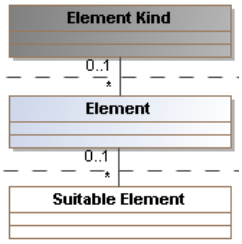

Figure 5. Abstraction levels

Extensible metamodel. The metamodel is oriented towards the structuring of so called "lightweight ISD agile methods". Therefore, the predefined structure must be straightforward and flexible enough to extend the metamodel in a case of such a necessity. This can be achieved by the hierarchy of "Element Kind" and "Relation Kind Use" (Figure 6).

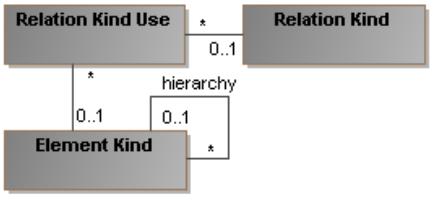

Figure 6. Elements allowing extensibility

Agility requirements. Due to the specific nature, agile methods are not universally suitable [10, 17]. It is possible to distinguish a set of the environment (organization, project) characteristics, where agile methods are most suitable. We prefer calling these characteristics "agility requirements". The default set of such agility requirements can be used to identify an IT organization and its project environment's suitability for agile methods. Each agility requirement is described by the pair of two kinds. "Measurement Kind" class indicates whether the requirement is quantitative or qualitative. "Content Kind" class defines different types of requirements, such as

technical, social, business, psychological, etc [36] (Figure 7).

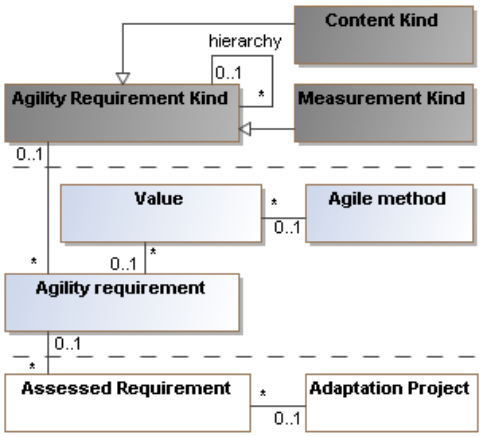

Figure 7. Agility requirements

Application areas. It is important to facilitate the initial process of the partial implementation model creation for the end user. Application areas can be described as a set of domains and disciplines of ISD engineering. Relating elements with these application areas allows the end user to facilitate the process of element selection (Figure 8).

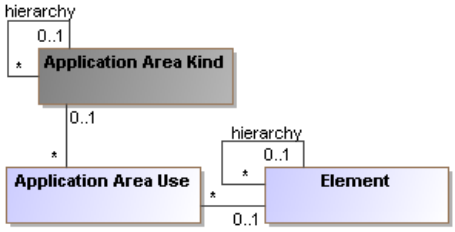

Figure 8. Application areas

Element levels. Most elements (techniques, artifacts, practices, etc.) are presented with the complete static content or dynamic usage descriptions in the sources on agile methods. It is a well-known fact that in actual development projects ISD methods are not used as they ought to be. For each element we propose to define three levels of its implementation: minimal, balanced, full. "Minimal" level represents minimal steps needed for using that element. "Balanced level" is an intermediate level between "Minimal" and "Full" (Figure 9).

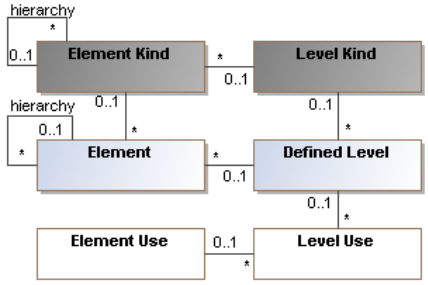

Figure 9. Element levels

Merge and generalization. More than ten ISD agile methods emerged since the publishing the Agile Manifesto in 2001 [4]. Their evolution raised the problem of element overlapping. The classes of this concept provide a possibility to merge and generalize similar or complementary elements [7] (Figure 10).

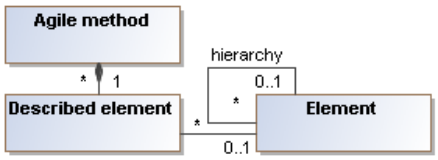

Figure 10. Elements allowing merge and generalization

Coupling. Breaking down methods into elements gives a possibility to merge and generalize similar or complementary elements. Hence, it becomes possible to combine elements from different agile methods. However, having a lot of small elements burdens the process of selecting elements in the third tier. The coupling allows grouping of closely related small elements into bigger (independent) elements (Figure 11).

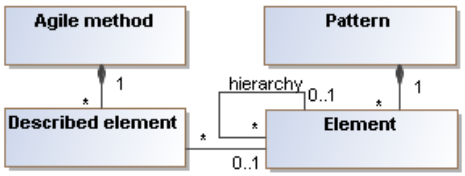

Figure 11. Coupling the elements

Pattern. The result of the work of the agile method engineer is in the second tier. It consists of structured, merged and generalized elements from agile methods; their levels; relations to application areas and internal elements; agility requirements that are defined for decomposed methods. Multiple patterns may be created by different engineers. The concept gives the possibility to maintain different versions of patterns for the same agile methods [29] (Figure 12, Figure 11).

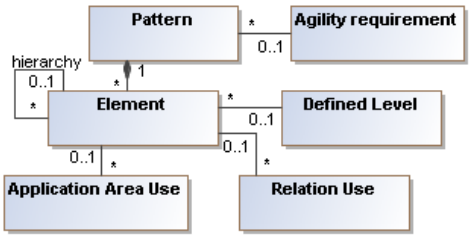

Figure 12. Pattern concept classes

Prioritization. The prepared patterns are used for creating an implementation model of a partial agile method adaptation in the third tier. Several prioritizations may be performed during the creation of a partial implementation model (Figure 13).

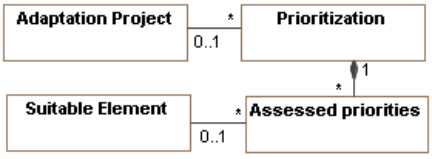

Figure 13. Priority concept classes

Criteria. The prioritization is performed using a subset of criteria from a predefined set (Figure 14). Criteria reflect the custom needs that are identified by the end user [26, 38]. Examples include "Easy to learn", "Easy to install", "Easy to use", "Low risk", "Required efforts", "Available resources" etc.

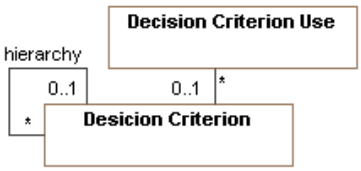

Figure 14. Criteria concept classes

Implementation model. The result of the third tier process is the implementation model (Figure 15).

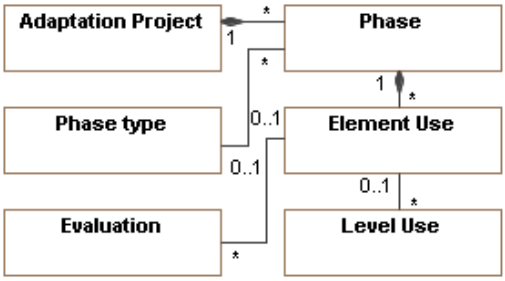

Figure 15. Implementation concept classes

The partial implementation can be described as an initialized adaptation project that consists of several phases (initial, intermediate, final). During the first phase, the user identifies concerns as application areas, selects the related elements and their minimal levels. The initial phase can be described as a trial. During the intermediate phases, user configures the use of selected elements (removes non-profitable elements, selects new elements or their higher levels).

5. The final metamodel

This section presents the summarized view of the metamodel of the framework for the partial agile method adaptation. The architecture of related element kinds, elements and selected elements is presented in 0. The classes used for the first, second and third tiers are filled in a dark grey, light grey and white color respectively. The core classes used for element classification are defined as "Element Kinds" (classes in a dark grey). The classes used for pattern composition of structured agile methods are defined as "Elements" (classes in light grey). The classes used for an implementation model are defined as "Suitable Elements" (classes in white). This architecture defines how the main classes are related and arranged through the tiers that can be used for method decomposition and their element classification.

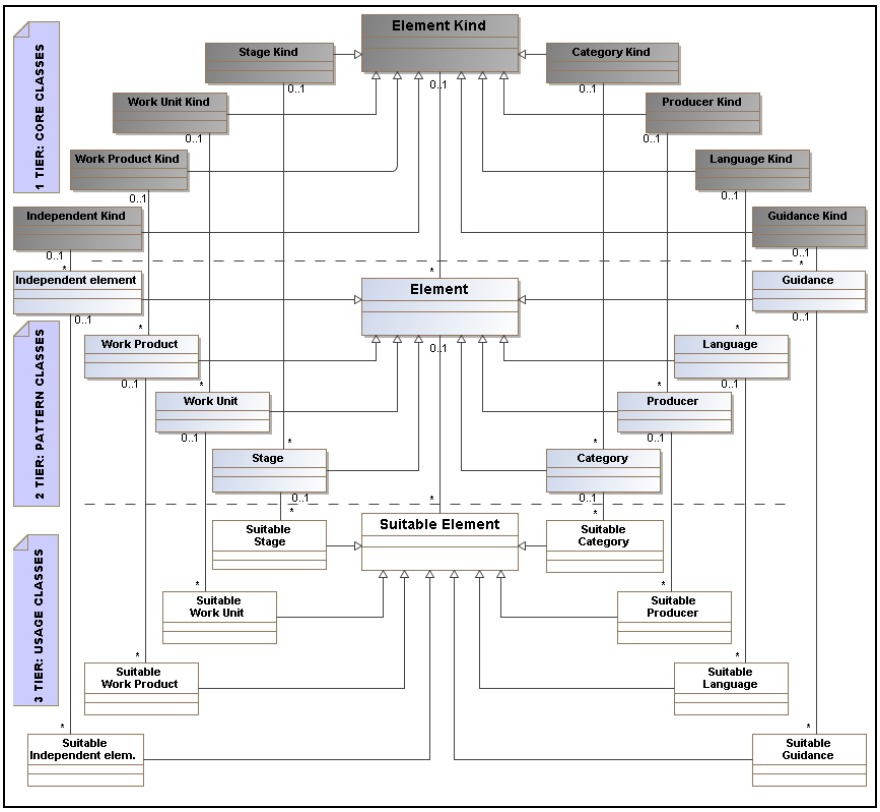

Figure 16. The architecture of the method decomposition

Using this architecture, it is possible to structure agile methods into patterns. Furthermore, it is possible to have many implementation models using these patterns. The three tier architecture distinguishes the metamodeling of the element classification structure from the modeling and implementing the actual agile methods. The relations between the different kinds of elements are shown in Figure 17.

The proposed metamodel of the framework has eight classes of Element Kind. Any agile method can be structured into a set of elements of these kinds. They are “Guidance Kind”, “Stage Kind”, “Producer Kind”, “Work Unit Kind”, “Work Product Kind”, “Language Kind”, “Category Kind” and “Independent Kind”. The stage (process) consists of producers that are responsible for producing assigned work products. If a work product is a code, then the class “Language” is used. The producer performs the work units (activities, steps) that manipulate related work products. Any element may have a related guidance (example, checklist, supporting material). Any fragment of agile method that does not fit to other kinds is defined using the “Category Kind”. Related elements can be grouped into the independent elements that facilitate the implementation process in the third tier. The composite metamodel is presented in Figure 18.

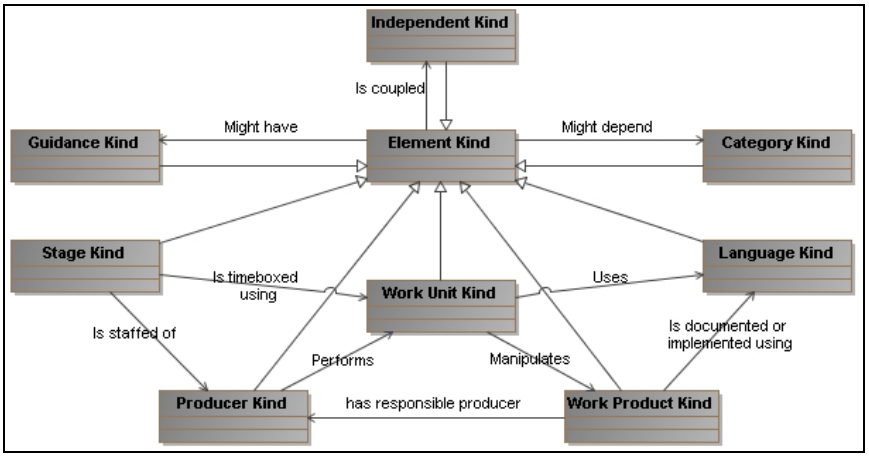

Figure 17. The core classes used for the method decomposition

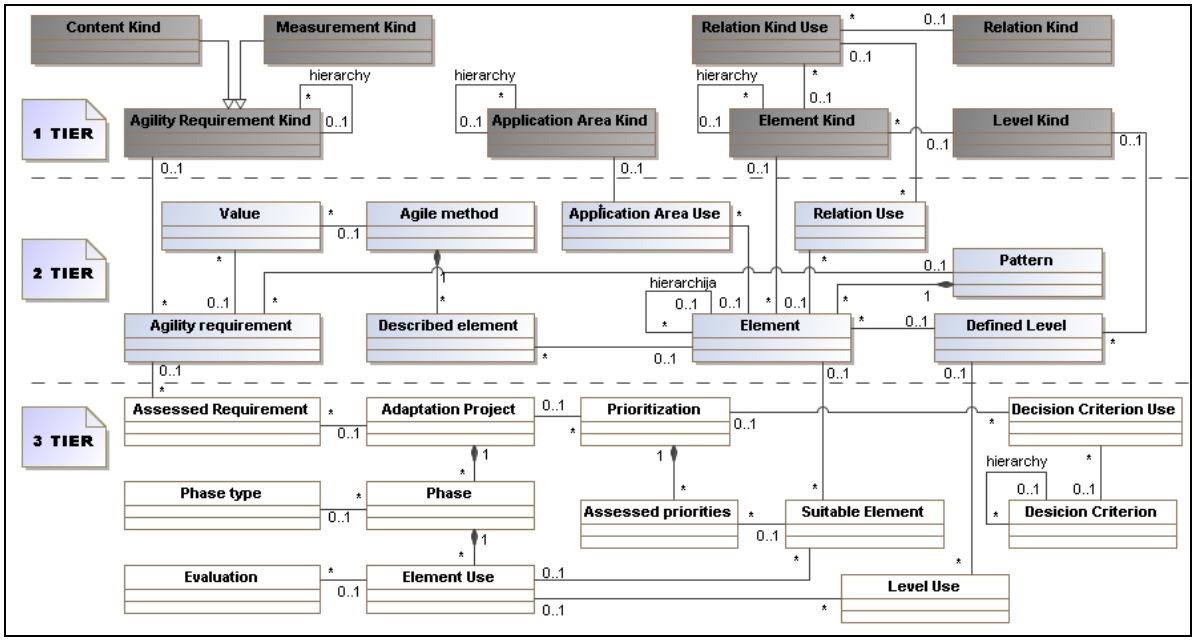

Figure 18. Composite metamodel of the framework for the partial agile method adaptation

The framework is divided into three tiers. Therefore, the classes are grouped with respect to these tiers. An agile method engineer can extend the structure used for agile method decomposition. This means he can add new element or relations kinds, define new internal relations or element levels, supplement the list of application area kinds or define new types of agility requirements in the first tier. Using this structure (first tier, dark grey classes), many agile methods may be structured into patterns (second tier, light grey classes) composed from elements and related agility requirements. Many different patterns can be created by different agile method engineers due to their different skills or experience. The pattern is the result of composition of decomposed different agile methods and is built using the predefined structure (first tier, dark grey classes). These patterns may be used for constructing many implementation models (third tier, white classes) for the selected partial agile method adaptation.

6. Case study

In this section, we will overview the application of the framework by presenting steps that are performed during all three tiers. Due to a lack of space, we will present only fragments of each step using illustrative scenarios. The tasks that will be covered are presented in Table 4.

Table 4. The tasks of using the framework for the partial adaptation

S1	Extend the metamodel
S1.1	Add new children classes to "kind" classes
S2	Create the pattern of XP and Scrum agile methods
S2.1	Structure the XP method
S2.2	Structure the Scrum method
S2.3	Merge and generalize similar or complementary elements
S2.4	Couple elements into independent elements
S2.5	Relate elements with corresponding application areas
S2.6	Define adaptation levels for independent elements
S3	Create an implementation model for the partial implementation of XP and Scrum methods
S3.1	Assess environmental suitability for the agile methods
S3.2	Select suitable elements
S3.3	Perform the prioritization of the independent elements with respect to criterions
S3.4	Build the implementation model for the selected elements

Table 5. Extended kind classes with the classes from OPF metamodel

Core classes	Detailed classes
Stage Kind	Cycle Kind, Phase Kind, Build Kind , Milestone Kind
Work Unit Kind	Activity Kind, Task Kind, Work Flow Kind, Technique Kind
Producer Kind	Organization Kind, Team Kind, Role Kind, Project Kind, Tool Kind
Language Kind	Constraint Kind, Implementation Kind, Modeling Kind, Natural Kind, Specification Kind, Database Kind, Interface Kind, Scripting Kind
Work Product Kind	Application Kind, Architecture Kind, Component Kind, Diagram Kind, Document Kind, Metric Kind, Model Kind, Requirement Kind, Database Kind, Convent. Kind

S1. Extend the metamodel

S1.1. Add new children classes to kind classes. The proposed metamodel has a flexible structure that allows to extend it with the classes from existing metamodels. For example, an agile method engineer can extend the proposed metamodel by adding new classes from the OPF metamodel [34].

S2. Create the pattern of XP and Scrum agile methods

S2.1. Structure the XP method. Using the original source of the XP method [9], it is possible to extract such elements as XP team (Team Kind), customer (Role Kind), programmer (Role Kind), architect (Role Kind), tester (Role Kind), interaction designer (Role Kind) and others. They are responsible for producing such elements as metaphor (Architecture Kind), user stories (Requirement Kind), iteration plan (Document Kind), code (Work Product Kind) using pair programming (Technique Kind) performing estimate iteration (Task Kind), etc. In addition, corresponding agility requirements such as real customer involvement, friendly environment, the policy of the company can be defined.

S2.2. Structure the Scrum method. Using the original source of the SCRUM method [35], it is possible to extract such elements as SCRUM team (Team Kind), SCRUM master (Role Kind), Product Owner (Role Kind), developer (Role Kind) etc. They are responsible for producing such elements as Product backlog list (Requirement Kind), Sprint backlog list (Requirement Kind) performing Daily Scrum Meeting (Task Kind), Sprint Backlog task (Task Kind), Sprint review meeting (Task Kind) during Sprint (Phase Kind).

S2.3. Merge and generalize similar or complementary elements. XP and SCRUM methods are often described as complementary agile methods. Therefore, SCRUM is oriented towards the process while XP emphasizes supplementing techniques and practices. However, the overlapping elements also exist (see examples presented in Table 6.

Table 6. Overlapping or complementary elements from XP and SCRUM

XP	SCRUM
Iteration	Sprint
User stories	Product backlog list
User stories selected for iteration	Sprint backlog list
Metaphor	Architecture

Also, the common elements such as developer, tester, customer, process lifecycle exist in most ISD methods and they are a subject for merging when creating a pattern.

S2.4. Couple elements into independent elements. If we take a closer look at the descriptions of elements in the agile methods, we will find that most of them are closely related to each other. For example, a user story (Work Product Kind → Requirement Kind) is closely related to such tasks as derive requirements (Task Kind), analyze requirements (Task Kind), estimate requirements (Task Kind), and to such producers as developer (Role Kind), customer (Role Kind) and project manager (Role Kind). The independent element that consists of these smaller elements simplifies the use of these elements later, working in the third tier.

S2.5. Relate elements with corresponding application areas. The relations of elements and application areas also facilitate the selection of elements by the end user. Such elements as "Pair Programming", "Refactoring" may be related to such application area as "Code quality". The elements "Ten-Minute Build", "Continuous Integration" can be related to "Early testing". The elements "Test-First Programming", "Incremental development" may be related to "Get close to business values". The elements "Open workspace", "Energized work" may be related to "Tuning work performance".

S2.6. Define adaptation levels for independent elements. Sometimes, it is not desirable to follow all the steps, sections or adaptation levels when adapting an element. For example, if a template is used for requirement specification, then sometimes only the major sections are used for capturing the requirements due to some restrictions on time or available resources. Another example is selecting the duration of iteration. "Weekly cycle" is defined as an element describing the duration of the iteration. It may be impossible to perform a weekly iteration. The possible definitions of the element adaptation levels would be full = weekly, minimal = bi-monthly, balanced = one week during a month.

S3. Create an implementation model for the partial implementation of XP and Scrum methods.

S3.1. Assess environmental suitability for the agile methods. Agile methods are not suitable everywhere due to their specific nature. The extraction of agility requirements gives the possibility to perform an approximate assessment of the method suitability. For example, if an IT company is performing a project where requirements are clear, unambiguous, and non-changing, but there is a poor customer involvement, project manager distrusts the stakeholders, then it is likely that the more rigorous plan-driven ISD methods should be used instead of the agile methods.

S3.2. Select suitable elements. From the end user's point of view, the framework facilitates the process of element selection. For example, customer identifies his needs by selecting application areas "Code quality", "Tuning work performance" and "Get close to business values" first. Then, considering the existing relations, corresponding elements such as "Pair Programming", "Refactoring", "Test-First Programming", "Incremental development", "Open workspace", "Energized work" can be proposed as the most suitable solutions.

S3.3. Perform the prioritization of the independent elements with respect to criteria. There is always a balance between the elements needed and resources available to apply them. Whenever there is a need for an optimal decision, it is wise to try the proven decision making techniques. The prioritization is the process of ranking the elements with respect to criteria such as the ease of use, ease of learning, cost, benefit and etc. There is a formal decision making technique such as Analytic Hierarchy Process (AHP) that should be used for element prioritization [36].

S3.4. Build the implementation model for the selected elements. Let us assume that elements "Refactoring", "User stories", "Sprint", "Metaphor", "Open workspace", "Working conventions", "Application Refactoring", "Daily sprint meeting", "Pair programming", "Continuous Integration" were ranked as top ten elements having highest cost/value ratio among other suitable elements during prioritization. The implementation model for the partial agile method adaptation can be described as a plan. The selected top ten elements should be implemented incrementally during the phases.

Table 7. Implementation of selected elements

Do (Phase 1): try elem1(lvl 1), elem2(lvl 1), elem3(lvl 1), elem4(lvl 1), elem5(lvl 1),…, elem10(lvl 1);
Evaluate (Phase 1): dismiss elem2, neutral elem5, elem10, eager for elem1, elem3, elem4;
Do (Phase 2): try elem1(lvl 2), ~~elem2(lvl 1)~~, elem3(lvl 2), elem4(lvl 2), elem5(lvl 1),…, elem10(lvl 1);
Evaluate (Phase 1): dismiss elem10, neutral elem5, eager for elem3, elem4;
Do (Phase 3): try elem1(lvl 1), ~~elem2(lvl 1)~~, elem3(lvl 3), elem4(lvl 1), elem5(lvl 1),…, ~~elem10(lvl 1)~~;
Evaluate (Phase 1) dismiss elem5, eager for elem1;

7. Conclusions

An in-depth analysis of the problem of a partial agile method adaptation revealed that this problem can be described as a composition of the several sub-problems. Subsequently, the required solution for a partial agile method adaptation must cover all of the sub-problems. We defined a set of concepts, where each concept has direct or indirect relation to these sub-problems. The integral solution has been achieved by developing these concepts, deriving their classes and organizing them into the framework of the partial agile method adaptation. The constructed metamodel for the framework serves as a structure for the decomposition of the agile methods. It is a guide for creating patterns and developing models for the partial implementations of the agile methods from these patterns.

A contribution of this paper is manifold. First, it combines classes from the OPF, ISO/IEC 24744 and SPEM metamodels along with the new proposed classes. The developed metamodel also has a straightforward structure oriented towards the decomposition of the lightweight agile methods. Therefore, the metamodel can be extended by adding new element or relations kinds, defining new internal relations or element levels, application area kinds or new types of agility requirements for emergent custom needs. Moreover, the proposed metamodel implements such concepts as agility requirements, levels of element adaptation, application areas, criteria and prioritization that are used for decision making when building an implementation model. Note that existing Situational Method Engineering metamodels are lacking of such concepts.

References

[1] **L. Ablonskis, L. Nemuraitė**. Discovery of complex model implementation patterns in source code. *Information Technology and Control*, 2010, 39(4), 291–300.

[2] **L. Ablonskis, L. Nemuraitė**. Discovery of model implementation patterns in source code. *Information Technology and Control*, 2010, 39(1), 68–76.

[3] **P. Abrahamsson, J. Warsta, M.K. Siponen, J. Ronkainen.** New directions on agile method A comparative analysis. *In proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society*, 2003, 244 – 254.

[4] **Agile Alliance**. Principles behind the Agile Manifesto. *Available from: http://agilemanifesto.org /principles.html* [*Accessed* 20 *September* 2009].

[5] **S.W. Ambler.** Agile Adoption Rate Survey: March 2007. *Available from: http://www.ambysoft.com /downloads/surveys/AgileAdoption2007.ppt* [*Accessed* 15 *April* 2009].

[6] **S.W. Ambler.** Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. *John Wiley & Sons*, 2002.

[7] **A. Armonas, L. Nemuraitė.** Using attributes and merging algorithms for transforming OCL expressions to code. *Information Technology and Control*, 2009, 38(4), 283–293.

[8] **I. Attarzadeh, O.S. Hock.** New direction in project management success: Base on smart methodology selection. *In proceedings of Information Technology Symposium, Springer*, 2008, 1–9.

[9] **K. Beck.** Extreme Programming Explained: Embrace Change, Second Edition. *Addison Wesley Professional*, 2004.

[10] **B. Boehm, R. Turner.** Using Risk to Balance Agile and Plan-Driven Methods. *Computer*, 2003, 36(6), 57–66.

[11] **J. Charvat.** Project Management Methodologies – Selecting, Implementing, and Supporting Methodologies and Processes for Projects. *John Wiley & Sons*, 2003.

[12] **A. Cockburn**. Selecting a Project's Methodology. *IEEE Software*, 2000, 7(4), 64–71.

[13] **A. Cockburn, J. Highsmith.** DSDM Business Focused Development, *Addison-Wesley*, 2003.

[14] **A. Cockburn.** Crystal Clear A Human-Powered Methodology for Small Teams. *Addison Wesley Professional*, 2004.

[15] **J. Drobka, D. Noftz, R. Raghu.** Piloting XP on Four Mission-Critical Projects, *IEEE Computer*, 2004, 21(6), 70–75.

[16] **D.G. Firesmith, B. Henderson-Sellers.** The OPEN Process Framework, an introduction. *Addison-Wesley*, 2002

[17] **E. Georgiadou, K.V. Siakas, E. Berki.** Agile quality or depth of reasoning: applicability versus suitability respecting stakeholders' needs. *In proceedings of Agile software development quality assurance, Information Science Reference*, 2007, 23 – 55.

[18] **D. Greer, G. Ruhe.** Software release planning: An evolutionary and iterative approach. *Information and Software Technology*, 2004, 46(4), 243–253.

[19] **B. Henderson-Sellers, C. Gonzalez-Perez, J. Ralyte.** Comparison of Method Chunks and Method Fragments for Situational Method Engineering. *In proceedings of Software Engineering ASWEC 2008, IEEE Computer Society*, 2008, 479–488.

[20] **ISO/IEC**. Software Engineering - Metamodel for Development Methodologies. ISO/IEC 24744:2007 (E), 2007.

[21] **R.E. Jeffries, A. Anderson, C. Hendrickson.** Extreme Programming Installed, *Addison-Wesley*, 2000.

[22] **P. Kroll, B. MacIsaac.** Agility and Discipline Made Easy: Practices from OpenUP and RUP. *Addison Wesley Professional*, 2006.

[23] **C. Lan, K. Mohan, X. Peng, B. Ramesh.** How Extreme does Extreme Programming Have to be? Adapting XP Practices to Large-scale Projects. *In proceedings of the 37th Hawaii International Conference on System Sciences, IEEE Press*, 2004, 342 – 250.

[24] **L. Layman, L. Williams, L. Cunninghan.** Exploring extreme programming in context: An industrial case study. *In proceedings of the Agile Development Conference, IEEE Computer Society*, 2004, 32–41.

[25] **J. Lynch.** New Standish Group report shows more project failing and less successful projects. Available from: *http://www.standishgroup.com/newsroom /chaos_2009.php* [*Accessed* 2 *September* 2009].

[26] **G. Mikulenas, R. Butleris.** An approach for modeling technique selection criterions. *In proceedings of the 15th International Conference on Information and Software Technologies, IT* 2009, *Kaunas University of Technology*, 2009, 207–216.

[27] **G. Mikulėnas, K. Kapočius.** An Approach for Prioritizing Agile Practices for Adaptation. *In proceedings of 18th International Conference on Information Systems Development, Springer*, 2010, 485–498.

[28] **G. Mikulėnas, K. Kapočius.** A Framework for Decomposition and Analysis of Agile Methodologies during their Adaptation. *In proceedings of 18th International Conference on Information Systems Development, Springer*, 2010, 547–560.

[29] G. Mikulėnas, R. Butleris. An approach for constructing evaluation model of suitability assessment of agile methods using analytic hierarchy process. *Electronics and Electrical Engineering*, 2010, 10(106), 99–104.

[30] E. Mnkandla, B. Dwolatzky. Agile methodologies selection toolbox. *In proceedings of the International Conference on Software Engineering Advances ICSEA* '07, *IEEE Computer Society*, 2007, 72 – 72.

[31] M. Mirakhorli, A.K. Rad, F.S. Aliee, A. Mirakhorli, M. Pazoki. RDP Technique: Take a Different Look at XP for Adoption.Software Engineering. *In proceedings of the* 19*th Australian Conference on Software Engineering, ASWEC* 2008. *IEEE Computer Society*, 2008, 656–662.

[32] I. Mirbel. Method chunk federation. *In proceedings of workshops on Exploring Modeling Methods for Systems Analysis and Design, Namur University Press*, 2006, 407-418.

[33] OMG. Software Process Engineering Metamodel Specification. *OMG Document Number: formal*/2002-11-14, 2002.

[34] OPEN Process Framework Repository Organization (OPFRO). OPF repository. Available from: *http://www.opfro.org/* [*Accessed* 9 *February* 2010]

[35] K. Schwaber. Agile Project Management with Scrum. *Microsoft Press*, 2004.

[36] D. Silingas, R. Butleris. Towards implementing a framework for modeling software requirements in MagicDraw UML. *Information Technology and Control*, 2009, 38(2), 153-164.

[37] A. Smaizys, O. Vasilecas. Business Rules Based Agile ERP Systems Development. *Informatica*, 2009, 20(3), 439 – 460.

[38] L. Tutkutė, R. Butleris, T. Skersys. An approach for formation of leverage coefficients-based recommendations in social network. *Information Technology and Control*, 2008, 37(3), 245 – 254.

Received September 2010.