Experience-Based Approach for Adoption of Agile Practices in Software Development Projects

Iva Krasteva, Sylvia Ilieva, and Alexandar Dimov

Sofia University St.Kliment Ohriski, 65 Akad. J.Boucher str.,
Sofia, Bulgaria
iva.krasteva@rila.bg, sylvia@acad.bg, aldi@fmi.uni-sofia.bg

Abstract. The agile approach for software development has attracted a great deal of interest in both academic and industry communities in the last decade. Nevertheless the wide adoption of agile methods in ever growing number of software development projects, shifting the development process of an organization to an agile one is not straightforward. Certain considerations for the applicability of agile practices should be taken into account when this transition is performed. In this paper, an approach for situational engineering of agile methods is proposed. The approach is based on the experience gained in adopting agile practices in both internal and external projects of organizations. A knowledge-base supporting the selection of agile practices that are suitable for certain project is introduced. Automated generation of appropriate software development process is included as well. Particular realization of the approach supported by SPEM-based tools is also presented in the paper.

Keywords: agile practices, method engineering, project situation, context, SPEM.

1 Introduction

The agile approach for software development has attracted a great deal of interest in both academic and industry communities in the last decade. Based on common sense values and principles of collaboration, trust and the potential of the individuals, a number of agile practices have been proposed and a good number of agile methods have been developed. The agile software development has arisen as an alternative to the traditional development practice. A lot of success stories of its adoption in continuously growing number of domains and projects have been published. Nevertheless its wide adoption, shifting the development process of an organization to an agile one is not straightforward [1] [2] and certain considerations of the applicability of agile practices should be taken into account [3] [4]. In addition, the agile approach encourages adaptation and customization of the development method throughout the execution of the project, which makes the adoption process a continuous and interactive activity [5] [6] [7].

The objective of this research is to support organizations in introducing agile practices in their development methods and in further adoption and improvement through the whole project. We propose a situational method engineering approach which is based on the experience gained in adopting agile practices in both internal and external projects. The suggested approach incorporates knowledge for agile practices applicability and suggests different mechanisms for knowledge acquisition. There are two major activities in the approach. The first one is *selection of agile practices* that are suitable for particular project. *Creation of a new agile development method* is the second one. Due to space limitations the paper presents the overall approach and describes in details the selection process, while the creation of the development method is discussed briefly.

The approach we propose supports iterative and incremental process for methodology creation and adaptability through a number of method engineering milestones during the execution of the development process. Our approach is built on an instance of a Software Process Engineering Meta-Model (SPEM) 2.0 meta-model [8] and its execution is supported by several tools implemented as extensions to Eclipse Process Framework (EPF) Composer [9]. Models that are used as input and output for the method engineering process are fully compatible with the standardized instance of SPEM 2.0 Meta-Model- the SPEM 2.0 Base Plug-in [8], and thus can be freely distributed among different SPEM-based tools.

The rest of the paper is organized as follow. Section 2 makes an overview of the related work and comparison between other approaches and this study is made. In Section 3 the method engineering process is presented as well as basic concepts and models used. Organization of the knowledge-base and the selection of applicable agile practices are described in details in Section 4. How the generation of new agile method is specified and realized is presented in Section 5. Section 6 proposes validation of the approach through comparison with an external source and a case study. Section 7 concludes the paper and makes suggestions for future work.

2 Related Work

Method engineering is an approach for conceptualization, construction and adaptation of methods and tools for information systems development [10]. Situational method engineering (SME) deals with the creation of project-specific method or tailoring existing ones to a particular project situation. Project situation is defined [11] as a combination of the (external) context of the project and the project type. We are adhering to this terminology throughout the whole paper.

A recent overview of the existing approaches for method engineering can be found in the works of Ralyte *et al.* [12] and Nehan *et al.* [13]. Depending on the method construction techniques a number of approaches can be distinguished. Assembly-based approach is based on the reuse of preexisting method components (in different approaches referred to as fragments [10] or chunks [12] with difference in the meaning). Components are selected to suit particular situation and are assembled using appropriate technique to form new development methods. The extension-based approach combines instantiation and extension techniques to form new methods by applying extension patterns. Another approach, the paradigm-based approach lies on the abstraction strategy by either abstracting from existing method or by instantiating a metamodel.

Two approaches which focus on method tailoring are developing recently. The practice-driven approach [14] enforces particular rules to configure company's development method. In the Method for Method Configuration (MMC) approach [15] particular process is customized by applying appropriate configuration patterns. As outlined in [14], different SME approaches differ in their ability to execute by providing sufficient details on how the approach actually works. In the presentation of our approach we describe in parallel its specification as well as its realization. Of course, due to space limitation the realization doesn't reveal all the details. Our approach is realized on SPEM 2.0 Meta-Model specification and the EPF Composer tool. By utilizing standardized and open source tools and specifications, we believe that the applicability and extendibility of the approach is greatly enhanced. In [16] [17] SPEM is used to represent agent oriented methodologies following the method engineering approach. They extend SPEM in appropriate way to support modeling of method fragments and agent design processes. Our approach, however, is different in the way the realization model is instantiated from SPEM 2.0 Meta-Model and the design of the development process.

The value of introducing a knowledge-base to SME approaches is recognized by others [18] [19]. Keeping track on how the method has been previously applied in different project situations gives the possibility to execute various analyses based on particular metrics. Klooster *et al.* [18] have introduced eighteen performance indicators to measure the success of a project which is determined by the dependencies among situation factors. Qumer *et al.* [19] argue on the importance of introducing a knowledge-base when creating agile methods and state that agile knowledge engineering and management approach should be integrated with an agile software development approach. Our knowledge-base is especially designed to support introduction of agile practices to a given project situation based on the experience of practices applicability from both outside and inside of an organization.

Comprehensive studies on the applicability of the agile approach by utilizing OPEN framework [22] have been published by Qumer *et al.* [19] [20] and Hendersson-Seller *et al.* [21]. In [19] a complete framework to assist managers in assessing the degree of agility they require and how to identify appropriate ways to introduce agility into their organization is presented. The Agile Software Solution Framework (ASSF) consists of agile conceptual aspect model and tools. The agile conceptual aspect model represents the aspects of knowledge, governance and method core; which are linked to business via a bridge that aligns business goals and agile software development goal. In addition to the framework, Agile Adoption and Improvement Model (AAIM) is suggested to support method adoption and further software process improvement efforts. Our approach shares some common points in the assessment of agile practices. However, while the ASSF focuses on assisting managers to decide on appropriate transition and the way to execute it, we are only focusing on method engineers whose objectives are to design and further tailor the method through its execution to best suit the people involved in the project and the particular project situation. It should support method engineers who are not experts in agile adoption to select appropriate practices and guide them in the creation of the new agile method.

3 Description of the Approach

The section makes an overview of the new experience-based situational method engineering approach suggested by us. First, the method engineering process is described. The concepts and models that are involved in the formal specification of the approach are presented in the second subsection.

3.1 The Method Engineering Process

The introduction of agile practices to the development methods of an organization should involve iterative and incremental process of adoption while continuously reviewing and customizing the practices to the particular project [6] [5] [19]. This process is based on two types of adaptation [7] - adaptation to the characteristics of the project and self-adaptation which is guided by the way the team responds to different development practices used in the project. For that reason our method engineering process is interleaved with the software development process and thus supporting the method engineer to adapt the process continuously either by manually tailoring it or by executing the method engineering process for any process iteration. The projects are described by a set of factors that characterizes particular project situation.

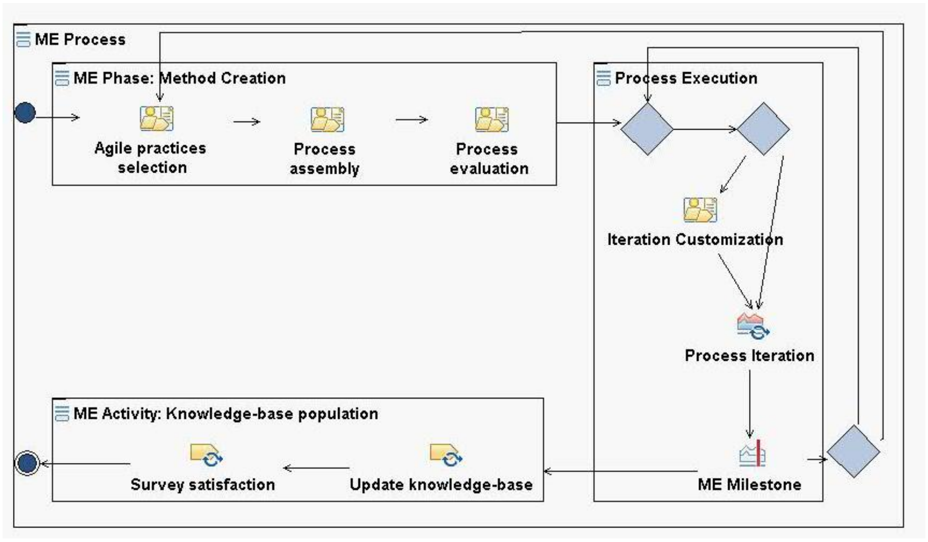

Fig. 1. The method engineering process

Three major parts constitutes the method engineering process proposed by our approach - method creation phase, process execution phase and knowledge-base population activity. The method creation phase begins with selection of agile practices that are applicable to the given project situation. The activity involves a number of steps for evaluation of practices applicability based on the information in a knowledge-base. The knowledge-base stores data for theoretical and empirical adoption of agile practices when certain project and environmental factors are present. Appropriate practices are then assembled with the current development process in a number of possible processes which are further evaluated and one particular is chosen. The adoption is performed in a number of process iterations each of which ends with a method engineering milestone. If the iteration is not the last one, decides whether the next iteration should be changed or not. The method engineer can optionally customize some of the elements of the iteration or execute the method engineering process with a new set of practices. After the last iteration two additional method engineering tasks are performed- the information for the project is added to the knowledge-base and stakeholders' satisfaction is surveyed. The method engineering process is presented in Figure 1.

3.2 Concepts, Models and Tools

Our situational method engineering approach is based on SPEM 2.0, which is defined as a MOF 2.0-based Meta-Model as well as a UML 2 Superstructure-based Profile. SPEM is designed to provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes [8]. In SPEM distinction between reusable method components (called Method Content) and instantiation of such components in particular processes is drawn. The method content provides the 'building blocks' such as tasks, work products, roles, tools, guidance, etc., while the process elements specify parts or the whole development process as a work breakdown structure (including activities as well as work products and roles) by implementation and further customization of the elements from the method content. In this sense a development method consists of different method content elements and process elements. Methods can be specified by reusing, extending and customizing method and process elements from other already defined methods by means of Method Configurations. The repository for methods is called Method Library. Figure 2 illustrates these concepts and their relationships.

Our method engineering approach is based on four metamodels. Two of them are standardized models- the SPEM 2.0 Meta-Model and the SPEM 2.0 Base Plug-in. We define the other two models- the Specification and the Realization metamodels. These models are derived from the SPEM metamodel by extension and/or instantiation.

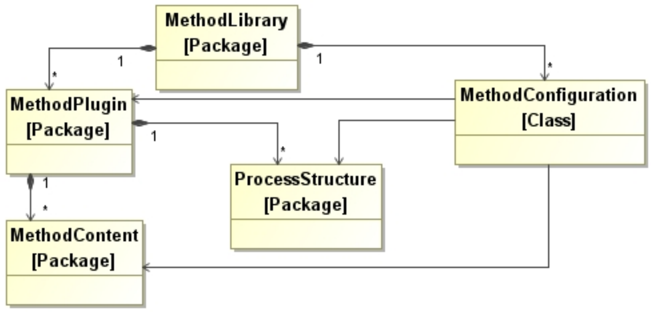

Fig. 2. Definition of SPEM 2.0 basic concepts

Most of method engineering concepts and reuse mechanisms can find their corresponding ones in SPEM elements. One such mapping assumed by us is presented in Table 1. The knowledge-base that we introduce in the approach is modeled using standard UML. However, it imports certain process and method content elements from our SPEM-based metamodels. The other way relation is not provided explicitly.

Table 1. Mapping of method engineering concepts to SPEM 2.0 Meta-Model elements

Method Engineering concepts	SPEM 2.0 Meta-model elements
Method component	Method content; Process
Method repository;	Method library

Aggregation;
Configuration;
Instantiation
	Variability types;
	extension mechanisms

Specification metamodel presents the basic concepts of our approach and relationships between them. In order to execute our approach we have specified three tools that are realized as extensions to EPF Composer. EPF Composer utilizes the SPEM 2.0 Base Plug-in which is a standardized instantiation of SPEM 2.0 Meta-Model and provides commonly used instances for many SPEM concepts for the domain of Software Engineering [8]. For that reason we specify another model- the Realization metamodel, which imports the concepts form the Base Plug-in as well as certain design rules and validations to ensure that the model conforms to the Specification model. By introducing a second metamodel, the modeled elements that come as an input or output of our approach can be transferred to any SPEM-based tools for modeling software development processes. Figure 3 presents the relationships between different models used in our approach.

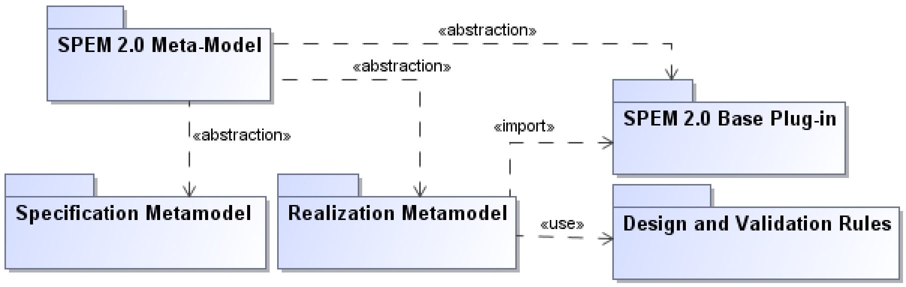

Fig. 3. Models introduced in the approach and their relationships

The three tools that are realized by us as extensions to EPF Composer are: the APA tool, the ConfCheck tool and the EBAGen tool. The Agile Practices Applicability (APA) tool supports all the operations with the knowledge-base of agile practices- feeding up data, editing data, configuring the situation parameters, analyzing the data. The tool also exposes a wizard-like interface to help the method engineer in the selection of agile practices appropriate for a particular project. The (Experience-Based

Approach method GENerator) EBAGen tool guides the generation of the new agile process from particular set of agile practices and a number of requirements specified by a method engineer. The Conformity Check (ConfCheck) tool is used mainly by the other tools to check the conformity of input method elements with the specification model.

4 Method Content Selection

As discussed in the introductory section the applicability of agile practices is heavily dependent on characteristics of the projects. Certain practice can be applied in one context while other cannot. For that reason in our approach we maintain a knowledge-base which stores information for applicability of particular agile practice when given project situation is present. The organization of the knowledge-base as well as different mechanisms that are used to analyze it and are used in the selection of appropriate agile practices for the project in hand are described in the subsections below.

4.1 Organization of the Knowledge-Base

Currently, there is no standard taxonomy of projects in the software industry that can be used to identify and categorize projects based on common factors. A lot of studies are published but there is no consistent and complete set of factors that are used to describe particular project situation. Jones [23] identifies 36 important context factors have been documented by Software Productivity Research (SPR) and used for benchmarking projects. However, this classification lacks some important factors that are relevant for agile development while introducing others that are not so important. Thus we use this project characterizing set of factors as a starting point and *map, adjust, add* and *remove* factors that are identified in related studies considering agile practices applicability [18] [20] [3] [24] [25]. As an outcome of we have identified 45 factors to be included as characteristics of different projects situations [26].

Once we have identified the relevant factors, the knowledge-base is designed to support the analysis of the applicability of agile practices according:

- Theoretical inapplicability
- Experience reports of other companies
- Experience of the company

The *theoretical inapplicability* analysis is based on the characteristics of a practice and makes assumptions about how appropriate is to apply certain practice when particular factor is present. We have defined 7-point scale {-5,-3,-1, 0, 1, 3, 5} to measure the levels of applicability- from highly inapplicable (-5) to highly applicable (5). Zero means that applicability of certain practice is not affected by the presence of a factor. In previous works of ours [27] [28] we have studied theoretical applicability of agile practices when particular social, technological and business factors are present as well as their adoption in the development and usage of reusable components. In order to start populating the knowledge-base we are examining and synthesizing relevant theoretical research by us and others.

However deep and systematic those theoretical analysis are, the assumptions drawn by them can be validated through further *empirical studies*. The knowledge-base stores information of projects in which agile practices have been applied. Relevant data is taken from published experienced reports on agile adoption through systematic review of literature. The projects are described by means of the set of situational factors we have identified, and practice applicability is recorded in a 5-point scale {N/A, 0, 1, 3, 5} showing the extent to which the practice is used ('N/A' stands for 'Unknown'). The evidence of the success or failure of practice usage is also recorded in the knowledge-base. The information in the knowledge-base can be further analyzed and conclusions based on the experiences of other companies of the applicability of agile practice when certain factor is present can be made. We are currently researching appropriate statistical methods to analyze the data. Although based on empirical reports, there are a couple of issues that should be considered when this type of analysis is made. Such issues consider the high level of uncertainty and subjectivity of the information, as well as the quality of the reports. The presence of certain situational factors might not be explicitly revealed in the report however they can influence applicability of given practice in the real situation. Mapping project descriptions to situational factors, deciding the level of applicability and the evidence of success or failure involves speculative reasoning and its effect on the results should be taken into account. The last consideration is about the quality of the empirical reports. In a systematic review [29] of empirical studies up to and including 2005 year, out of 270 studies only 36 of them satisfied certain quality criteria. In order to mitigate the impact of such issues the statistical significance and quality of the data in the sample is examined as part of the statistical analysis. Furthermore, as an additional step a statistical analysis on a set of situational factors to a set of practices is introduced. In this way influences among practices are also taken into consideration and sets of practices are suggested for application. For the purpose of comparing how close one project situation is to the other, we introduce the measure of situation proximity. Situation proximity of two situations is measured as a sum of:

- '1' for each factor that are the same in both situations
- '0' for each factor that is different
- when a factor is not known, the value of the probability this factor to match the one in the other situation (this is dependent on the cardinality of the scale for particular factor).

An additional feature of the knowledge-base is the information about incompatible agile practices, which is based on theoretical research of the characteristics of agile practices like the one made by us [27]. Such information is useful when the set of practices is evaluated.

The knowledge-base contains also information for applicability of agile practices in the *past projects of the organization*. This is the most reliable information however not available for first-time agile adoption. The information from internal projects contains complete project situations and thus the uncertainty level is not an issue. The records in this part of the knowledge-base can contain customized practices as well as general ones. Also for this type of information we can collect additional information on results of practice application. In [18] eighteen performance indicators are suggested to measure the success of particular practice applicability. We would like to keep our information repository light and measure the success of a given practice by the satisfaction of project stakeholders.

4.2 The Selection Process

The knowledge acquisition and analysis is supported by the APA tool. The selection of agile practice to be used in the project is iterative procedure and is based on a number of parallel analysis steps. A *precondition* to the selection procedure is the method content that contains the agile practices we want to be analyzed. As a *first optional step* of the selection procedure the state of the knowledge-base is set – it could be either fed up with data, or the data can be updated, or just reviewed. This data, however, is not stored in some dedicated repository. Once the information about theoretical and empirical applicability of practices is specified by the means of the tool, it is serialized as text (according predefined by us format) and attached to a dedicated text attribute in each agile practice. In such a way the information about agile practices applicability is attached to them and can be read by appropriate parser tool. On a *second step* the method engineer specifies the project by the set of situational factors.

The third step includes several parallel or sequential analyses:

- Theoretical analysis per practice per situational factor
- Empirical analysis per practice per situational factor
- Empirical analysis per project situation for external projects (according to situation proximity)
- Empirical analysis per project situation for internal projects (according to situation proximity)

The method engineer selects several sets of practices which are checked for incompatibilities. The data for theoretical incompatibility of two practices is stored in the knowledge-base. Some of the practices are substituted with custom practices from the internal records. The method engineer can decide to run some of the analyses again and change the preliminary sets of practices. At the end, one set is selected to be used in creating the development method. The output of the selection procedure is a new method configuration in EPF Composer that contains selected agile practices. The description of the project situation is attached to an appropriate text attribute in the new development process, which is to be populated in the next step.

5 Development Process Creation

Once the appropriate set of agile practices is selected they can be adopted with the existing practice of the organization in a particular development process. How the new development process is generated is presented in the current section. We first present the specification metamodel.

5.1 The Specification Metamodel

The metamodel we define specifies the basic concepts and the relationships between them to support our approach. The basic concepts that are introduced in the metamodel are presented in Figure 4. We introduce AgilePractices and CustomAgilePractices elements as SPEM Guidance. Any custom agile practice customizes one of the general agile practices. AgileValues are SPEM Metrics that present agile values such as collaboration and simplicity [30] [7] [19] associated with each agile practice. Agile practices are categorized further depending on their purpose in the development method- either they are part of the development or project management activities in the software process, or they support the collaboration [7].We introduce three breakdown elements to support the method engineering process- the MEPhase and MEActivity as SPEM Activities and MEMilestone as SPEM Milestone.

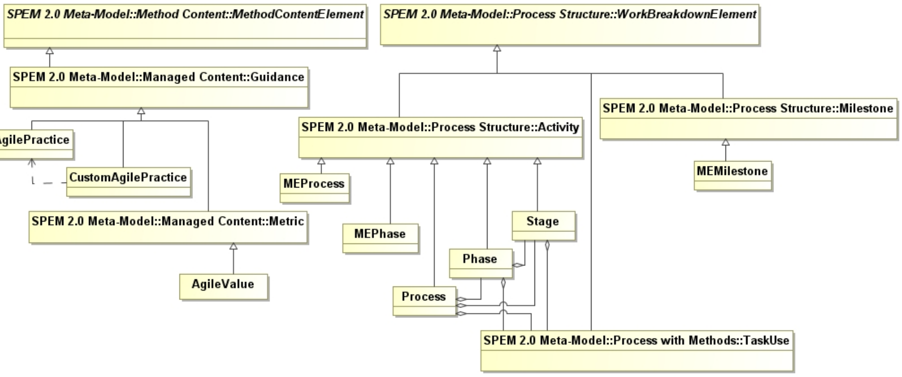
SPEM 2.0 Meta-Model::Method Content::MethodContentElement
SPEM 2.0 Meta-Model::Process Structure::WorkBreakdownElement
SPEM 2.0 Meta-Model::Managed Content::Guidance
AgilePractice
CustomAgilePractice
SPEM 2.0 Meta-Model::Process Structure::Activity
SPEM 2.0 Meta-Model::Process Structure::Milestone
MEProcess
MEMilestone
SPEM 2.0 Meta-Model::Managed Content::Metric
MEPhase
Stage
Phase
AgileValue
Process
SPEM 2.0 Meta-Model::Process with Methods::TaskUse

Fig. 4. Basic concepts in the experienced-based approach**-** specification metamodel

We use the process patterns as described in [31] to present and build our agile development process, which are modeled with Process, Phase, Stage and Task Description elements. The Task Description element is reused as-is from SPEM specification. A particular pattern of higher level can only contain patterns of the lower levels. An association of process pattern with AgilePractice guidance specifies an agile development process pattern. Each pattern of a higher level is associated with one input and one output Work Product Description. The Task Descriptions are associated with at least one Role Description and one Work Product Use. The Task Descriptions, Role Descriptions and Work Product Uses that are used in the process are instances of elements in the Method Content. Qualifications of Roles are specified with predefined enumeration of skills and competences, which is serialized in a text attribute of a Describable Element. In this way they are compatible with the model but are reused in our knowledge-base to model persons as part of the project situation. Work Product Definitions are categorized as Formal Work Product or Informal Work Products.

We also introduce the notion of Abstract Patterns and abstract Work Products. An abstract pattern is an activity of higher level- Process, Phase or Stage, in whose breakdown structure there is a leaf element which is also abstract. An abstract work product is a more general work product which needs to be instantiated in the process. Such general relationship among Work Products Definitions is expressed by A Work Product Definition Relationship in the SPEM Meta-Model.

5.2 Generation of Development Method

As discussed earlier, in order to provide compatibility of our approach with the SPEM 2.0 Base Plug-in implemented in EPF Composer, a number of design and validation rules are introduced. One such design rule is the use of specific custom categories instead of new stereotypes. The ConfCheck tool is used to check whether method and process elements that are inputs to the generation procedure conform to the specified design and validation rules. The EBAGen tool guides the method engineer in the generation of a new development process by introducing certain set of agile practices to the current development process. Two types of inputs are expected- the method and relevant process content for a set of agile practices and for the development method used in the organization. In addition, the method engineer should specify abstract patterns for the current development method as well as a number of disciplines that are domain specific and utilizes some of those patterns. The particular instance of the current development method should provide instantiation for these abstract patterns. Process generation algorithm uses a number of instantiations and assembly mechanisms to create new processes.

In the beginning, the method engineer specifies the number of method engineering phases and milestones. Then for each of them an abstract pattern to be instantiated is specified. The abstract patterns are available from the current method as well as from agile practices. Abstract patterns can be specified by the method engineer and can be included in the configuration to be used in the method engineering process. At the next step, the method engineer specifies abstract work products. If any tasks or work products should be preserved in the new method they are specified as mandatory. The method engineer also defines the roles to be used in the process by either selecting them from the predefined roles based on the available skills or by specifying new roles. Based on the skill set during process generation new roles are assigned to the tasks. Different variants of non-abstract methods are generated after selecting those roles, work products, tasks and lower level abstract patterns that can instantiate abstract patterns and their abstract work products. The method engineer evaluates the generated processes according:

- abstraction level- depends on the extent to which abstract patterns are instantiated. More abstract processes are liable to customization [7]
- business values of the included agile practices
- coverage of the disciplines of the domain
- lightness of the process from the involved types of work products

The method engineer can select one process or start the generation from the beginning with different initial parameters. When the method engineer chooses one particular process a new method configuration is created and project characteristics are copied to the delivery process. From any method engineering milestone the following ME phases can be regenerated starting from the initial configuration. After the regeneration the method configuration is updated accordingly.

6 Validation

We have designed and implemented our approach having in mind the methodology design principles and concepts identified by Cockburn [6].Cockburn mention four things that should be taken into account when designing a methodology:

- Variations in people
- Variations across projects
- Long debug cycles
- Changing technologies, techniques, and cultures

With the exclusion of the long debug cycles, the other three considerations are addressed in our approach by identification of project situation and people as part of it, and by supporting adaptation of the methodology throughout the project execution. Most of the methodology elements identified by Cockburn are either explicitly included in our approach or can be easily derived. Table 2 presents how the methodology elements are supported in our approach.

Table 2. Methodology elements and their support by the experience-based approach

Methodology Elements [Cockburn]	**Supported by:**
Process	Metamodel element Process
Milestone	Metamodel element Milestone
Activity	Metamodel element Stage
Technique	Metamodel element Task with associated Guidance
Skills	Metamodel element Skills and knowledge-base element
Role	Metamodel element Role
Team	Metamodel element Team profile
Tool	Metamodel element Tool
Standards	Work products and tools that are precondition for the creation of the development process
Product	Metamodel Element Work Product Description, Work Product Use
People	Knowledge-base- situation factors
Quality	Knowledge-base – satisfaction

Currently we are in a process of verifying our approach in a case study of the applicability of agile practices in the domain of component-based development. The case study we are currently completing examines the adoption of agile development in the implementation of both components and systems based on components. We have previously studied the theoretical adoption [28] by systematic analysis of the applicability of the agile approach in the development processes of components and systems. As a second step, we conducted an empirical research [32] in which the assumptions made in the first study were verified by an industry survey. As we had expected, some of the assumptions was confirmed by the answers of the professional, while others were not. Most of the situational factors were included in the structure of the survey and the data is suitable to feed up the knowledge-base with a number of relevant experience reports. We have received 43 complete answers, from which 33 are related to development of components and component-based systems. The structure of the questionnaire as well as the data is available as technical report [33]. The preliminary analysis of the results has shown that most of the agile practices are applicable in projects for implementation of components and systems based on components. In addition, the practitioners have expressed strong preference towards using the agile practices more rigorously.

7 Conclusion and Future Work

The paper presented an experience-based approach for situational engineering of agile methods. The approach supports systematic adoption of agile practices to the development method of organizations by analyzing experience reports on applicability of agile practices when particular situation factors are present. It introduces iterative and incremental method engineering process, adaptable to project characteristics and customizable through the execution of the project. Its realization is supported by a number of tools which are specified as extensions to EPF Composer tool. Method and process elements, used as input and produces as an output of the method engineering process, are fully compatible with the standardized SPEM 2.0 Base Plug-in and can be further managed by any other SPEM-based tool.

We are currently implementing the tools by the specifications, described briefly in the paper. As well, we are developing appropriate criteria for systematic review of available empirical and theoretical studies on agile practices adoption, which is to be used for knowledge-base initialization. In future, we want to add more formal presentation and evaluation of methodology conceptual terms such as ceremony, and weight. Introduction of simulations for evaluation of generated development processes is another extension to our approach planned for the near future.

Acknowledgments. The work is funded by Bulgarian Ministry of Education and Science, National Science Fund. The authors want to thank the anonymous reviewers for their valuable feedback.

References

1. Hodgetts, P.: Refactoring the development process: experiences with the incremental adoption of agile practices. In: Agile Development Conference, Salt Lake City, pp. 106–113 (2004)
2. Krasteva, I., Ilieva, S.: Adopting an Agile Methodology - Why It Didn't Work. In: ICSE: International workshop on Scrutinizing agile practices or shoot-out at the agile corral, pp. 33–36 (2008)
3. Kruchten, P.: Keynote: Situated Agility. In: 9th International Conference on Agile Processes and eXtreme Programming in Software Engineering (2008)
4. Koch, A.: Agile Software Development Evaluating the Methods for Your Organization. Artech House Publishers (2004)
5. Beck, K.: Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley Professional, Reading (2004)

6. Cockburn, A.: Agile Software Development: The Cooperative Game, 2nd edn. Addison-Wesley Professional, Reading (2006)
7. Highsmith, J.: Agile Software Development Ecosystems. Addison-Wesley Professional, Reading (2002)
8. Software process engineering metamodel. Version 2.0. formal/2008-04-01, OMG (2008)
9. EPF Composer. In: Eclipse Process Framework, `http://www.eclipse.org/epf/`
10. Brinkkemper, S.: Method engineering: engineering of information systems development. Information and Software Technology 38(7), 275–280 (1996)
11. Butcher, T., Klesse, M., Kurpjuweit, S., Winter, R.: Situational Method Engineering on the Differentiation of "Context" and "Project Type". In: Ralyte, J., Brinkkernper, S., Henderson-Sellers, B. (eds.) IFIP International Federation for Information Processing, Situational Method Engineering: Fundamentals and Experiences, vol. 244, pp. 33–48 (2007)
12. Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 95–110. Springer, Heidelberg (2003)
13. Nehan, Y.-R., Deneckere, R.: Component-based Situational Methods A framework for understanding SME. In: IFIP International Federation for Information Processing, Situational Method Engineering: Fundamentals and Experiences, vol. 244, pp. 161–175 (2007)
14. Bajek, M., Vavpotic, D., Krisper, M.: Practice-driven approach for creating project-specific software developmnet methods. Information and Software technology (49), 345–365 (2007)
15. Karlsson, F., Agerfalk, P.: Towards structured flexibility in information systems development: Devising a method for method configuration. Journal of Database Management 20(3), 51–75 (2009)
16. Seidita, V., Cossentino, M., Gaglio, S.: Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 46–59. Springer, Heidelberg (2009)
17. Cossentino, M., Gaglio, S., Henderson-Sellers, B., Seidita, V.: A Metamodelling-based Approach for Method Fragment Comparison. In: International Workshop on Exploring Modeling Methods in Systems Analysis and Design at CAISE 2006, pp. 419–432 (2006)
18. Klooster, M., Brinkkemper, S., Harmsen, F., Wijers, G.: Intranet facilitated knowledge management: a theory and tool for defining situational methods. In: Conference on Advanced Information Systems Engineering, Barcelona, Spain, pp. 303–317 (1997)
19. Qumer, A., Henderson-Sellers, B.: A framework to support the evaluation, adoption and improvement of agile methods in practice. The Journal of Systems and Software (81), 1899–1919 (2008)
20. Firesmith, D.G., Henderson-Sellers, B.: The OPEN Process Framework. Pearson Education, London (2002)
21. Qumer, A., Henderson-Sellers, B.: Agile software solution framework: An analysis of practitioners' perspectives. In: Information Systems: Modeling Development and Integration: Third International United Information Systems Conference, UNISCON 2009, pp. 41–52 (2009)
22. Henderson-Sellers, B., Serour, M.K.: Creating a Dual-Agility Method: The Value of Method Engineering. Journal of Database Management 4(14), 1–16 (2005)
23. Jones, C.: Software Assessments Benchmarks and Best Practices. Addison-Wesley Professional, Reading (2000)

24. Turk, D., France, R., Rumpe, B.: Assumptions underlying agile software-development processes. Journal of Database Management 16(4), 62–87 (2005)
25. Cockburn, A., Highsmith, J.: Agile software development, the people factor. Computer 34(11), 131–133 (2001)
26. Krasteva, I., Ilieva, S.: Characterizing Agile Projects. Internal Report, Sofia Uniresity 'St. Kliemnt Ohridski', Sofia (2009)
27. Krasteva, I., Ilieva, S.: Rush into Agile- Analytical Framework for Agile Practices Applicability. In: IET Conference Publications (528 CP), Durham, pp. 229–237 (2007)
28. Krasteva, I., Branger, P., Land, R.: Challenges for agile development of COTS components and COTS-based systems A theoretical examination. In: International Conference on Evaluation of Novel Approaches to Software Engineering, Funchal, Madeira, pp. 99–106 (2008)
29. Dyba, T., Dingsyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology 50(9-10), 833–859 (2008)
30. Manifesto for Agile Software Development, `http://agilemanifesto.org/`
31. Tasharofi, S., Ramsin, R.: Process Patterns for Agile Methodologies. In: IFIP International Federation for Information Processing, Situational Method Engineering: Fundamentals and Experiences, vol. 244, pp. 222–237 (2007)
32. Krasteva, I., Land, R., Sajeev, A.S.M.: Being Agile when Developing Software Components and Component-Based Systems- Experience from Industry. In: EuroSPI Conference, vol. Industrial Proceedings, pp. 8.7-8.17 (2009)
33. Causevic, A., Krasteva, I., Land, R., Sajeev, A.S.M., Sundmark, D.: An Industrial Survey on Software Process Practices, Preferences and Methods. Malardalen University, Sweden (2009)