Enhancing Tool Support for Situational Engineering of Agile Methodologies in Eclipse

Zahra Shakeri Hossein Abad, Anahita Alipour, and Raman Ramsin

Abstract. In recent years, with the growth of software engineering, agile software development methodologies have also grown substantially, replacing plan-driven approaches in many areas. Although prominent agile methodologies are in wide use today, there is no method which is suitable for all situations. It has therefore become essential to apply Situational Method Engineering (SME) approaches to produce agile methodologies that are tailored to fit specific software development situations. Since SME is a complex process, and there is a vast pool of techniques, practices, activities, and processes available for composing agile methodologies, tool support–in the form of Computer Aided Method Engineering (CAME) environments–has become essential. Despite the importance of tool support for developing agile methodologies, available CAME environments do not fully support all the steps of method construction, and the need remains for a comprehensive environment. The Eclipse Process Framework Composer (EPFC) is an open-source situational method engineering tool platform, which provides an extensible platform for assembly-based method engineering in Eclipse. EPFC is fully extensible through provision of facilities for adding new method plug-ins, method packages, and libraries. The authors propose a plug-in for EPFC which enables method engineers to construct agile methodologies through an assembly-based SME approach. The plug-in provides facilities for the specification of the characteristics of a given project, selection of suitable agile process components from the method repository, and the final assembly of the selected method chunks, while providing a set of guidelines throughout the assembly process.

Keywords: Agile methodology, Situational Method Engineering, Eclipse Process Framework Composer, Computer-Aided Method Engineering.

Zahra Shakeri Hossein Abad · Anahita Alipour · Raman Ramsin
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: z_shakeri@ce.sharif.edu, alipour@ce.sharif.edu, ramsin@sharif.edu

R. Lee (Ed.): Software Engineering Research, Management and Appl. 2012, SCI 430, pp. 141–152.
springerlink.com

1 Introduction

The simplicity and development speed of agile methodologies are the main reasons for their popularity. Although prominent agile methodologies are available, there is no general-purpose agile methodology which fits all situations. This has led to the application of Situational Method Engineering (SME) approaches to produce project-specific methodologies that are tailored to fit specific development situations. Like all engineering disciplines, efficient application of SME methods is dependent on the availability of adequate tools; Computer Aided Method Engineering (CAME) environments have been developed for this purpose [1, 2].

There are three main SME approaches [3]: *Assembly-based* SME, in which a method is constructed from reusable method components that are extracted from existing methodologies and stored in a repository called the "method base"; *Extension-based* SME, in which existing methods are extended and enriched by applying extension patterns [1]; and *Paradigm-based* SME, in which a new method is constructed by instantiating a metamodel or applying abstraction to existing methods. Among the different approaches to SME, *Assembly-based* SME is the most commonly used and has become the basis of method construction in CAME tools. Method development in these tools consists of three distinct stages: Specifying the method requirements based on the situation of the project, selecting the appropriate method fragments, and assembling the fragments into a coherent methodology.

In assembly-based engineering of agile methodologies, CAME tools are expected to provide the necessary means for performing the following four stages [2]: (1) Specification of a set of methodology requirements by characterizing the project at hand; (2) Development of an agile method base (repository) by extracting a set of method fragments from existing agile methodologies; (3) Matching the extracted method chunks with the methodology requirements, thereby forming a cohesive set of method chunks; and (4) Supplementing the method chunks with guidelines on how they can be assembled into a coherent process. The main shortcoming of existing CAME tools is that they only partially cover these stages [1].

The Eclipse Process Framework Composer (EPFC) [4] is the single most prominent CAME tool currently used by method engineers. EPFC already provides support for the instantiation of XP and Scrum methodologies, but this support is partial; EPFC represents these methodologies as general methods, and does not support assembly-based SME stages for constructing bespoke methodologies from their components. In order to address the shortcomings of existing CAME tools, we propose ASEAME (Assembly-based Situational Engineering of Agile Methodologies in Eclipse) as a plug-in for EPFC which enables method engineers to construct agile methodologies through an assembly-based SME approach. ASEAME provides facilities for the specification of the characteristics of a given-project, selection of suitable agile process components from the method repository, and the final assembly of the selected method chunks, while providing a set of guidelines throughout the assembly process.

The rest of this paper is organized as follows: Section 2 briefly reviews the literature related to this work and highlights the contributions of this research; Section 3 explains the details of ASEAME; Section 4 evaluates ASEAME according to the

ISO/IEC 9126 quality model and compares it to other CAME tools; and Section 5 presents the concluding remarks and suggests ways for furthering this research.

2 Related Research

Since the early years of method engineering, several academic prototypes of CAME environments have been introduced [1], and different versions of them have been developed. Since our main focus is on CAME tool support for *agile* methodologies, we have divided this research work into the following categories:

- *Research conducted on CAME environments in general:* None of the research efforts in this category has resulted in CAME tools that provide adequate support for the method engineering process. Method Base [5] is one of the primary tools in this area, which is focused on helping the method engineer in selecting the appropriate method for the project at hand. This tool does not support some of the features of assembly-based SME, but provides facilities for method customization. Other CAME tools in this category, including Decomerone [1], MENTOR [6], Method Editor [7], MERU [8], and METAEdit+ [9] provide partial support for the assembly-based approach, but only MENTOR and MERU cover method requirements analysis; others just support the design and implementation stages. MENTOR and MERU, in turn, have other problems: In MERU, methods are considered only in the product part; therefore, method fragments defined in this tool are not comprehensive; since method fragments collection is a prerequisite for method fragments selection, we regard this is as a major flaw. MENTOR, on the other hand, supports both assembly-based and paradigm-based approaches, and the assembly-based approach is not its main focus.

- *Research conducted on CAME-tool support for agile methodologies*: Among the few studies that have been conducted in this area, EPFC is the only tool introduced that represents two agile methodologies (SCRUM and XP) in their entirety, and that provides the means to instantiate these methodologies. However, this cannot be considered as adequate support for assembly-based SME [2]. In [10], a toolbox is introduced for agile methodology selection which assists the method engineers in classifying the projects, selecting agile methodologies, and selecting agile practices. However, the method classification factors provided in this tool are very limited. Moreover, the tool proposes a limited collection of agile practices for the project at hand, and even these limited practices are provided separately for each agile methodology. Furthermore, the ultimate assembly of the practices is not supported; in other words, this toolbox does not support the assembly-based approach.

3 Particulars of ASEAME

ASEAME is an Eclipse plug-in for EPFC which supports the comprehensive implementation of assembly-based SME, and enhances situational method engineering of agile methodologies. As shown in Fig. 1, EPFC provides extension facilities for adding new method plug-ins, method packages, and method libraries [1]. In

this section, we first study the coverage of the generic SME process in EPFC, and then introduce the architecture of ASEAME.

3.1 Coverage of the Generic SME Process in EPFC

Although EPFC is being used extensively by methodology engineers for ME and SME purposes, this environment does not provide full coverage of SME stages, and requires a high level of involvement by the method engineer. In this section, we investigate the mapping between the generic SME stages (as defined in [4]) and the EPFC, and present ASEAME with the purpose of providing complete coverage of SME stages, so that the deficiencies of EPFC are properly addressed.

EPFC is an open-source Eclipse project which has been created using the Eclipse Integrated Development Environment, and which supports a large number of Eclipse plug-ins. This CAME environment enables process engineers and managers to implement, deploy, and maintain situation-specific methodologies, and is intended to provide the following two facilities:

- A knowledge base to help developers learn their responsibilities in SME projects. This knowledge base includes external content as well as the users' own content, and can be used for educational purposes.
- A catalog of predefined processes which helps method engineers learn how to perform their responsibilities in a process, and understand how the different tasks in a process relate to one another. Some of these processes are complete "deliverables" that can be adapted to individual situations. Other processes (called "capability patterns") are building blocks for other complete processes, and represent the best development practices for specific disciplines, technologies, or management styles.

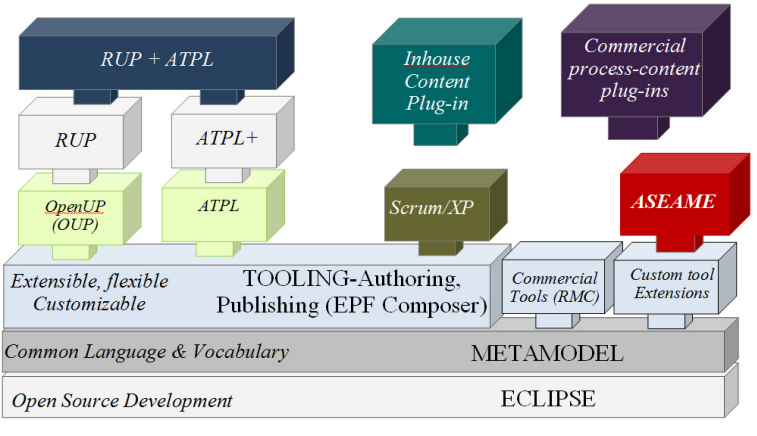

Fig. 1 Position of ASEAME in the EPFC architecture

In [4], a generic process is proposed for SME which consists of three main steps:

- *Situation characterization:* In which project situations are distinguished by using a set of factors called "situational factors." Project managers and method engineers can specify the situation of the project at hand by assigning values to these factors.
- *Method fragments selection:* In which the method fragments that correspond to the project characterization are selected from the method base (repository).
- *Method assembly:* In which the selected method fragments are assembled to form a coherent situational method. Method engineers need proper guidelines and rules to develop consistent methods.

EPFC does not support the "situation characterization" stage. It only supports two stages: Selection of method fragments, and method assembly. However, this support is only partial, and requires a high level of involvement on the part of the method engineer. Since characterizing the situation of the project at hand is a key step in determining method requirements, ignoring this step reduces the quality and efficiency of the methodologies produced.

Considering the vast amount of activities, tasks, and techniques available for agile methodologies, the method engineer's deep involvement in fragments selection not only significantly increases the complexity of the task, but also turns it into an error-prone process. Moreover, if method engineers do not have sufficient knowledge of all agile methodologies, necessary and useful method fragments may be ignored. ASEAME is specifically intended to address these issues.

3.2 ASEAME Architecture

EPFC uses the System Process Engineering Metamodel (SPEM 2.0) standard for method decomposition [17]. In SPEM 2.0, method content is made up of reusable components that compose processes. Elements are of three types: roles, work products, and tasks. To organize the components and define them at different levels of abstraction, and also to delimit the sequence of the activities performed, SPEM 2.0 incorporates the concepts of lifecycle, phase, activity, task, and technique (in descending order of granularity). As mentioned in [2], we have adopted SPEM 2.0 in defining the proposed agile method base (used in ASEAME's method repository).

Throughout the rest of this section, we present a complete description of ASEAME, based on the stages of assembly-based SME that it addresses. Fig. 2 and Fig. 3 show the ASEAME screens corresponding to these stages.

3.2.1 Situation Characterization

ASEAME characterizes the project at hand through defining a series of situational factors for agile methodologies. As shown in Table 1, these factors are organized in three groups: Application Domain, Project Organization, and Environment.

A default value is defined for each group. Discussion on how to select these factors, however, is out of the scope of this paper. The interested reader is referred to [1, 18, 19, 20].

After these factors have been initialized, the method requirements will be determined, and the input for the next step (method fragments selection) is provided.

Table 1 Situational Factors for Agile Methodologies

	Decision Factors	Possible Values in ASEAME
Environment	Degree of financial constraints	Normal / High
	Diversity of end-users	Wide / Narrow
	Time pressure imposed on the project	Yes / No
	Degree of importance of the project to the environment	Yes / No
Application Domain	Degree of formalism required in the methodology	Low / High
	Criticality of methodology quality factors	Normal / High
	Size of the target system	Normal / Large
	Criticality level of the target system	Normal / High
	Technology innovation level of the target system	Normal / High
	User-interface dependency of the target system	Low / High
Project Organization	Degree of developers' business knowledge	Adequate/ Inadequate
	Degree of developers' technical expertise	Adequate/ Inadequate
	Geographical distribution of development teams	Yes / No
	Distribution of skills among teams and members	Even / Uneven
	Degree of teams' acquaintance with agile methodologies	Adequate/ Inadequate

3.2.2 Method Fragments Selection

Once the situational factors are initialized by the method engineer, ASEAME provides facilities for selecting the appropriate method fragments.

Several approaches exist for method fragments selection: The Map [21] approach selects method fragments by measuring the similarity between the requirement's map and method fragments. However, such calculations may not provide sufficient distinction between method fragments, and the selected fragments might be similar. Therefore, choosing the appropriate method fragments may require a higher degree of involvement by the method engineer [20].

An alternative approach uses the Deontic matrix [22] for method fragments selection. This matrix is a two dimensional array of process elements, spanning activities, tasks, and techniques. These matrices can be developed at different process component levels, such as task/activity, task/technique, activity/technique,

and work-product/activity. Cell values specify the connection among these process components. The matrix can easily become huge if a large number of components are involved; filling the matrix can therefore become a time-consuming task for the method engineer.

The activity diagram approach [23, 24, 25] uses UML activity diagrams for selecting method fragments. Due to its low level of formalism, and the fact that diagram development heavily depends on the knowledge of the designer, implementing this approach in CAME-tools has proved difficult [23].

A multicriteria approach is also available [20], which resembles a modified version of the original assembly approach. Multicriteria techniques are commonly used in decision making for determining priorities based on the available alternatives. This is the approach used in ASEAME for method fragments selection.

3.2.3 Method Assembly

In [26], Brinkkemper presents techniques for assembling method fragments at both product and process levels. This approach formalizes the assembly process by defining rules and guidelines. In [5], Harmsen uses a set of rules in the form of mathematical axioms and derived corollaries and theorems for assembling method fragments. These rules focus on situation-independent factors such as completeness, consistency, efficiency, soundness, and applicability. Learning and implementing these rules can be very time-consuming for the method engineer. In addition, correct assembly is dependent on the method engineer's knowledge of mathematical formalisms, which might not be adequate. ASEAME supports this step, which has a strong impact on method consistency, through examining the selected method fragments from different aspects. ASEAME also provides guidelines in order to resolve the conflicts arising among rules, and also to address nonfunctional requirements in the final method.

3.3 Examples of ASEAME Screens

In this section, examples of ASEAME screens are shown. Fig. 2 shows the screen on which the method engineer will specify the *organization situational factors* of the project. ASEAME provides screens for entering other types of situational factors, including *environment situational factors* and *application domain situational factors.* Fig. 3 shows an example of the final results of ASEAME, consisting of assembly guidelines and a coherent agile methodology composed of phases, activities, tasks, techniques, roles, and work products.

4 Analysis of the Proposed Plug-In

The ISO/IEC 9126 quality model [27] will be used in this section for evaluating ASEAME. This model evaluates software systems by inspecting six main features. Considering that this model spans a huge number of standards, not all of which are applicable to CAME tools, an adaptation of ISO/IEC 9126 (introduced in [1]) is used for evaluating ASEAME.

Fig. 2 Screen for specifying "project organization" situational factors in ASEAME

Fig. 3 Final agile method constructed by ASEAME for a specific situation

These characteristics are organized into three main groups: Functionality, Usability, and Portability. Each characteristic consists of sub-characteristics. Results of this assessment are presented in Table 2. The evaluation results indicate that ASEAME adequately provides the features that are expected in a CAME tool.

Table 2 Evaluation of ASEAME based on ISO/IEC 9126

Characterization	Sub-characterization	Support in ASEAME	Explanation
Functionality	Suitability	✓_	Does not cover all SME approaches.
	Accuracy	✓	Uses exact algorithms for selection and assembly of method fragments.
	Functionality compliance	✓	Utilizes SPEM 0.2 as a standard for method development.
Usability	Understandability	✓	Uses intelligible interfaces.
	Learnability	✓_	There is adequate documentation on the EPFC environment, but formal documentation has not been produced for ASEAME yet.
	Operability	✓	EPFC supports method development graphical notations.
	Attractiveness	✓	Graphical user interfaces are designed to enhance ASEAME's attractiveness.
Portability	Installability	✓	Based on Eclipse
	Adaptability	✓	Based on Eclipse

✓ : Adequately supported
✓_ : Weakly supported

Fig. 4 depicts ASEAME's coverage of the generic SME process, and shows that ASEAME significantly augments the EPFC tool.

We have also compared ASEAME to existing CAME tools. In comparison, ASEAME offers these advantages:

- ASEAME supports the comprehensive implementation of assembly-based SME, which is its most important contribution.
- ASEAME significantly reduces the method engineer's manual burden, as compared to other CAME tools, through enhancing automation in all the stages of SME, including requirements engineering, selection of appropriate method fragments, and method fragments assembly. Additionally, the final assessment of method fragments consistancy, (which is a very time-consuming undertaking if performed manually) is executed automatically in ASEAME.

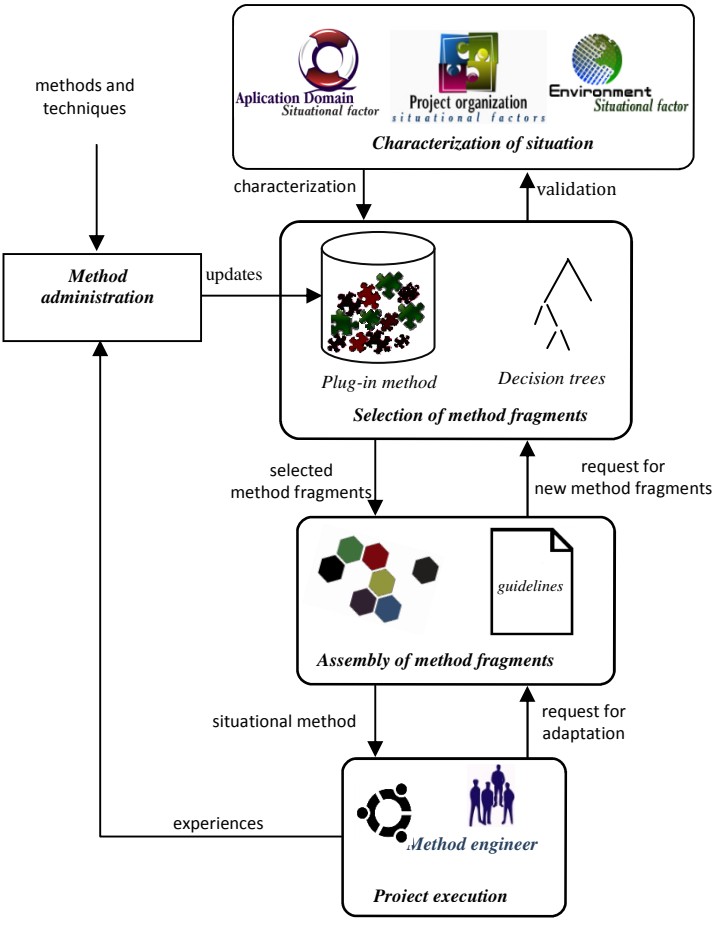

Fig. 4 Coverage of the generic SME process in ASEAME

- ASEAME provides a comprehensive set of agile method fragments derived from prominent agile methodologies, including Crystal Clear [11], DSDM [12], FDD [13], ASD [14], Scrum [15], and XP [16]. In comparison, other CAME tools lag far behind. The method fragments provided by ASEAME cover the *process* aspect as well as the *product* aspect. ASEAME supports the process aspect of methods by defining method fragments in terms of activities, tasks, and phases. The product aspect is supported by defining the related concepts in accordance with SPEM 2.0, in terms of artifacts, outcomes, and deliverables. Support for the definition of *roles* is another important feature of ASEAME.

5 Concolusions and Future Work

Our proposed CAME tool, ASEAME, enhances support for situational method engineering of agile methodologies in EPFC. Existing tools are limited in their support for comprehensive implementation of assembly-based SME, and they need a high degree of manual engagement by method engineers throughout the method construction process. ASEAME covers all the stages of SME (as defined in [1]) and reduces the method engineer's manual involvement through providing a high degree of automation. Thus, the complexities encountered in method engineering are adequately managed, production time and costs are reduced, and accuracy is enhanced.

ASEAME is based on *assembly-based* SME, and does not support other approaches, such as *paradigm-based* and *extension-based*. Therefore, the next step in this research is to extend ASEAME with support for other SME approaches. Method verification and enhancement of the agile method base are other research tasks that should be taken up.

References

1. Niknafs, A., Ramsin, R.: Computer-Aided Method Engineering: An Analysis of Existing Environments. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 525–540. Springer, Heidelberg (2008)
2. Shakeri Hossein Abad, Z., Hasani Sadi, M., Ramsin, R.: Towards Tool Support for Situational Engineering of Agile Methodologies. In: Proc. Asia-Pacific Software Engineering Conference (APSEC 2010), pp. 326–335 (2010)
3. Ralyté, J., Brinkkemper, S., Henderson-Sellers, B.: Situational Method Engineering: Fundamentals and Experiences. Springer (2007)
4. Haumer, P.: Eclipse Process Framework Composer. Eclipse Foundation (2007)
5. Harmsen, A.F.: Situational Method Engineering. Moret Ernest & Young (1997)
6. Si-Said, S., Rolland, C., Grosz, G.: MENTOR: A Computer-Aided Requirements Engineering Environment. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 22–43. Springer, Heidelberg (1996)
7. Saeki, M.: CAME: The first step to automated method engineering. In: Proc. OOPSLA 2003 Workshop on Process Engineering for Object-Oriented and Component-Based Development, pp. 7–18 (2003)
8. Heym, M., Osterle, H.: A semantic data model for methodology engineering. In: Proc. Workshop on Computer-Aided Software Engineering, pp. 143–155 (1992)
9. Meta Case Consulting: Method Workbench User's Guide (2005), `http://www.metacase.com/support/40/manuals/mwb40sr2a4.pdf/`
10. Mnkandla, E., Dwolatzky, B.: Agile methodologies selection toolbox. In: Proc. International Conference on Software Engineering Advances (ICSEA 2007), pp. 72–72 (2007)
11. Cockburn, A.: Crystal Clear: A Human-Powered Methodology for Small Teams. Addison Wesley (2004)
12. Stapleton, J.: DSDM: Business Focused Development, 2nd edn. Addison Wesley (2003)

13. Palmer, S.R., Felsing, J.M.: A practical guide to feature-driven development. Prentice Hall (2002)
14. Highsmith, J.: Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House (2000)
15. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall (2001)
16. Beck, K., Andres, C.: Extreme programming explained: Embrace change, 2nd edn. Addison Wesley (2004)
17. Bendraou, R., Combemale, B., Cregut, X., Gervais, M.: Definition of an executable SPEM 2.0. In: Proc. Asia-Pacific Software Engineering Conference (APSEC 2007), pp. 390–397 (2007)
18. Slooten, K.V., Hodes, B.: Characterizing IS Development Projects. In: Proc. IFIP TC8, WG8.1/8.2 Working Conference on Method Engineering, pp. 29–44 (1996)
19. Henderson-Sellers, B., Ralyté, J.: Situational Method Engineering: State-of-the-Art Review. Universal Computer Science 16(3), 424–478 (2010)
20. Kornyshova, E., Deneckere, R., Salinesi, R.: Method Chunks Selection by Multicriteria Techniques: An Extension of the Assembly-based Approach. In: Ralyte, J., Brinkkemper, S., Henderson-Sellers, B. (eds.) Situational Method Engineering: Fundamentals and Experiences, pp. 64–78. Springer (2007)
21. Rolland, C., Prakash, N., Benjamen, A.: A Multi-model View of Process Modeling. Requirements Engineering 4(4), 169–187 (1999)
22. Seidita, V., Ralyté, J., Henderson-Sellers, B., Cossentino, M., Arni-Bloch, N.: A comparison of deontic matrices, maps and activity diagrams for the construction of situational methods. In: Proc. CAiSE 2007 Forum, pp. 85–88 (2007)
23. Cossentino, M., Seidita, V.: Composition of a New Process to Meet Agile Needs Using Method Engineering. In: Choren, R., Garcia, A., Lucena, C., Romanovsky, A. (eds.) SELMAS 2004. LNCS, vol. 3390, pp. 36–51. Springer, Heidelberg (2005)
24. Saeki, M.: Embedding Metrics Into Information Systems Development Methods: An Application of Method Engineering Technique. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 374–389. Springer, Heidelberg (2003)
25. Van De Weerd, I., Brinkkemper, S., Souer, J., Versendaal, J.: A situational implementation method for web-based content management system-applications: Method engineering and validation in practice. Software Process: Improvement and Practice 11(5), 521–538 (2006)
26. Brinkkemper, S., Saeki, M., Harmsen, F.: Assembly Techniques for Method Engineering. In: Pernici, B., Thanos, C. (eds.) CAiSE 1998. LNCS, vol. 1413, pp. 381–400. Springer, Heidelberg (1998)
27. International Organization for Standardization (ISO), International Electrotechnical Commission (IEC): ISO/IEC 9126: Software engineering-Product quality. ISO (2004)