Online Method Engine: a Toolset for Method Assessment, Improvement, and Enactment

Kevin Vlaanderen, Fabiano Dalpiaz, Geurt van Tuijl, Sandor Spruit, Sjaak Brinkkemper
Utrecht University, the Netherlands

ABSTRACT

Software companies keep evolving their methods for software production, due to the continuous changes in the organizational, technological, and societal context. Implementing changes to existing methods is a complex activity, which depends not only on understanding 'what' to alter, but also on defining 'how' to apply the changes. Approaches in the literature are mainly focus on the 'what', provide only partial answers to the 'how' question, and offer no concrete toolset to support change.

In this paper, we propose and discuss a software toolset that realizes the online method engine (OME) concept and provides a concrete answer to the 'how' question by supporting the iterative and incremental assessment, improvement, and enactment of methods for software production. We describe the technical architecture of our toolset that concretely realizes our previously proposed OME concept, provide details on the method enactment mechanism, and report on a qualitative evaluation based on interviews with experienced industrial practitioners in process improvement.

Keywords: Software Product Management, Software Process Improvement, Knowledge Management, Experience Management, Incremental Method Engineering

Introduction

The methods that organizations employ for software production are in constant change, as companies strive to adapt their production processes to new technologies, business models, and development paradigms. Implementing changes in existing methods is a time- and cost-consuming activity, and its success is threatened by multiple factors, including resistance to change, fear of ineffectiveness, and resource constraints (Baddoo, 2003). Unsurprisingly, method change and enactment are important subjects of research in Software Process Improvement (Brocke & Sinnl, 2011; Feiler & Humphrey, 1993; Gonzalez-Perez & Henderson-Sellers, 2008).

Much research has been conducted on determining '*what*' changes to introduce in order to improve process, and thus software, quality. This has resulted in a wide range of frameworks for assessing and improving processes, including the CMMI (CMMI Product Team, 2002) and SPICE (El Emam, 1997) frameworks. While useful, these approaches stay at an abstract level and do not answer the important question of '*how*' to evolve existing methods to accommodate the needed changes.

This research provides an answer to the '*how*' question from a process-oriented perspective, i.e., by investigating how to support the planning and enactment of changes in the software production method. We do not consider here the equally important cultural or human aspects, such as informing personnel about the change, delivering training sessions, etc. We are motivated by the analysis of software process improvement (SPI) success in small and medium software enterprises by Pino, García, & Piattini (2008), which reveals that two crucial factors are (1) the conduction of an iterative and incremental improvement process that introduces changes based on their priority, and (2) the usage of technical tools and platforms that support the improvement process.

In this paper, guided by these two factors, we describe the realization of a knowledge-based tool that supports the generation of process improvement plans for a specific context. Such plans link the process areas that need to be improved to the available process alternatives and the order in which they are implemented.

Our baseline consists of incremental process improvement (J. P. Tolvanen, 1998; van de Weerd, Brinkkemper, & Versendaal, 2007)—a paradigm that promotes the step-wise improvement of process development processes—and the online method engine (OME) (Vlaanderen, van de Weerd, & Brinkkemper, 2013), the architecture of a knowledge management tool that facilitates incremental process improvement by supporting knowledge dissemination as well as method assessment, improvement, and enactment.

We make the following contributions:

1. We detail the technical toolset that realizes the OME conceptual architecture. To do so, we use the well-known the 4+1 architectural view model (Kruchten, 1995).
2. We illustrate how our developed toolset is a sound realization of the OME concept.
3. We detail how we support method enactment through the usage of a widely used, off-the-shelf requirements management tool that includes a workflow engine.
4. We report on a qualitative evaluation of our technical toolset that we obtained through interviews with product managers that have expertise in process improvement.

The remainder of the paper is structured as follows. We first discuss related work, showing how a technical toolset for method improvement is needed by researchers and practitioners. We present our baseline and illustrate the challenges to be addressed. We describe the realization of the OME concept by showing structural aspects, the process view, and usage scenarios. We illustrate how we support method enactment through a non-trivial mapping to a workflow engine. We report on empirical evaluation of our approach. Finally, we present conclusions and outline future directions.

Related Work

In software product management (SPM), the main domain of this paper, the skills and practices of a product manager are part of the intellectual capital of a software vendor. However, a problem with this intellectual capital is, as Rus & Lindvall (2002) put it, “it has legs and walks home every day”. This belief is shared by García, Amescua, Sánchez, & Bermón (2011), who tackle the problem by developing a software process knowledge repository developed as a wiki. Similarly, Mishra, Aydin, & Mishra (2013) have developed a basic knowledge management tool that facilitates the capturing of method knowledge. However, these approaches focus on knowledge dissemination only, leaving the issues related to the *use* of that knowledge unresolved.

In addition, many current approaches fail to gain traction within industry. While web-based knowledge bases such as the ones described above can help, the knowledge embedded within them is often too rigid and does not allow feedback. Within the area of knowledge management, a prime objective is to support not only the creation and organization, but also the development and leverage of knowledge (Wiig, de Hoog, & van der Spek, 1997). Mirbel (2007) attempts to improve this by integrating the solution with Communities of Practice. However, the focus remains on knowledge dissemination.

In the field of method engineering, several proposals related to the modelling, modification, and dissemination of standardized method fragments using automated tools exist, such as MERET (Heym & Osterle, 1992), MethodBase (Saeki & Iguchi, 1993), Decamerone (Harmsen & Brinkkemper, 1995), MENTOR (Si-Said, Rolland, & Grosz, 1996), MetaEdit+ (Kelly, Lyytinen, & Rossi, 1996), MERU (Prakash & Gupta, 1998), and Method Editor (Saeki, 2003). Unfortunately, most of these tools are used sparsely in practice, as users are often reluctant to use and trust method engineering techniques; something that is supported by the results of our evaluation interviews (Vlaanderen, Brinkkemper, & van de Weerd, 2012). In addition, there is a strong focus on the modelling aspect, and little on method on method enactment and improvement, with the exception of Method as a Service (Rolland, 2009)

The ability to construct, deconstruct, combine, and alter method fragments is not enough to provide adequate support during process improvement (Mirbel & Ralyté, 2006). According to several authors, including Rossi, Ramesh, Lyytinen, & Tolvanen (2004) and Ocampo & Munch (2006), method rationale is one of the key components of successful method engineering. Process owners are not only interested in *what* the process should look like, but also in *why* they should look like that (Karlsson, 2008). Current solutions are limited in their ability to combine the two views. This can result in misguided process evolution, that foregoes the original basic philosophy of a method (Karlsson, 2012).

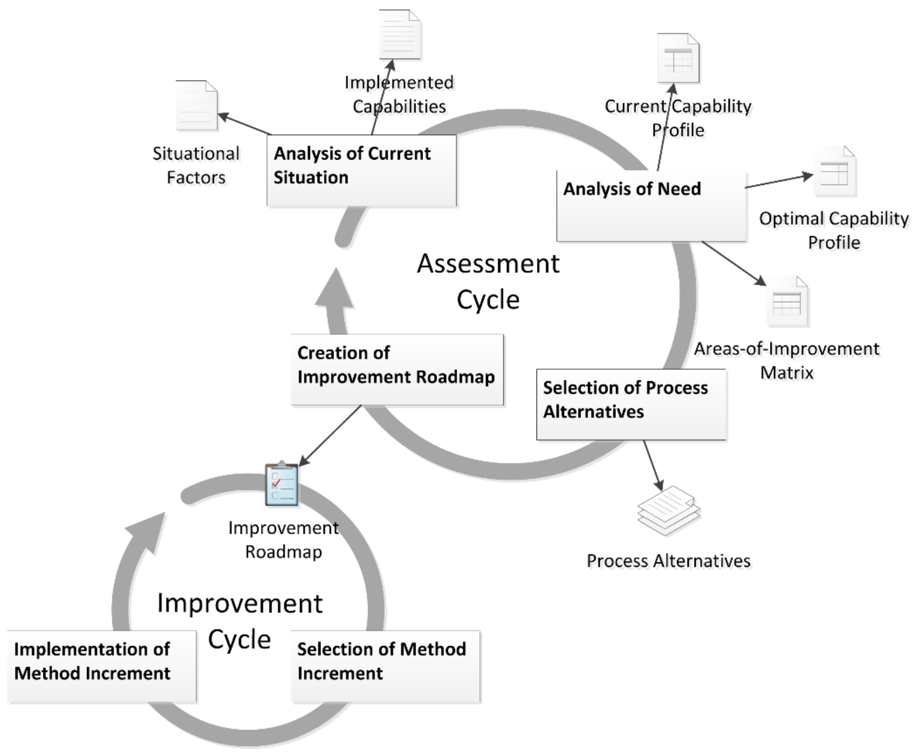

Implemented Capabilities
Situational Factors
Analysis of Current Situation
Current Capability Profile
Analysis of Need
Optimal Capability Profile
Assessment Cycle
Areas-of-Improvement Matrix
Creation of Improvement Roadmap
Selection of Process Alternatives
Process Alternatives
Improvement Roadmap
Improvement Cycle
Implementation of Method Increment
Selection of Method Increment

Figure 1. Incremental Process Improvement (Vlaanderen, van de Weerd, et al., 2012)

The need for this evolutionary aspect is becoming increasingly recognized. While traditional frameworks for assessing and improving processes, including the CMMI (CMMI Product Team, 2002) and SPICE (El Emam, 1997) frameworks, are useful, these approaches stay at an abstract level and do not answer the important question of '*how*' to evolve existing methods to accommodate the needed changes. However, Pino, García, & Piattini (2008) reveal that the conduction of an iterative and incremental improvement process that introduces changes based on their priority is crucial to successful process improvement. Unsurprisingly, new iterative SPI approaches have sprouted lately, such as the SPI-LEAM approach (Petersen & Wohlin, 2010).

Most of the recent work acknowledges the fact that method construction should always take into the account the specific characteristics of the situation in which it is to be applied. Reuse of existing knowledge is bound to factors such as cost, the existing process paradigm, and organizational characteristics such as team composition and the availability of skills within the organization (Becker, Janiesch, & Pfeiffer, 2007). The results from previous evaluations support this strongly, and these factors are guiding current and future design cycles of the OME.

Research Approach

Baseline

Situational method engineering (SME) (Becker & Knackstedt, 2007; Brinkkemper & Harmsen, 1995; Ralyté, 2004) suggests that development methods are constructed and configured depending on the specific situation (characteristics, environment) of the project under consideration. Methods are assembled starting from existing reusable components, which are tailored to the situation and integrated with the other methods that are already in place in the organization.

A key success factor for SME is that process improvement—i.e., the integration or replacement of components—shall be conducted *incrementally* (Harmsen, Brinkkemper, & Oei, 1994; Pino et al., 2008), through a series of small steps that fit well with the situation at hand. Our research over the past few years has focused on supporting evolutionary process improvement through incremental method evolution (van de Weerd et al., 2007). The starting point was the definition of the atomic types of adaptations, called increments (van de Weerd et al. (2007)).

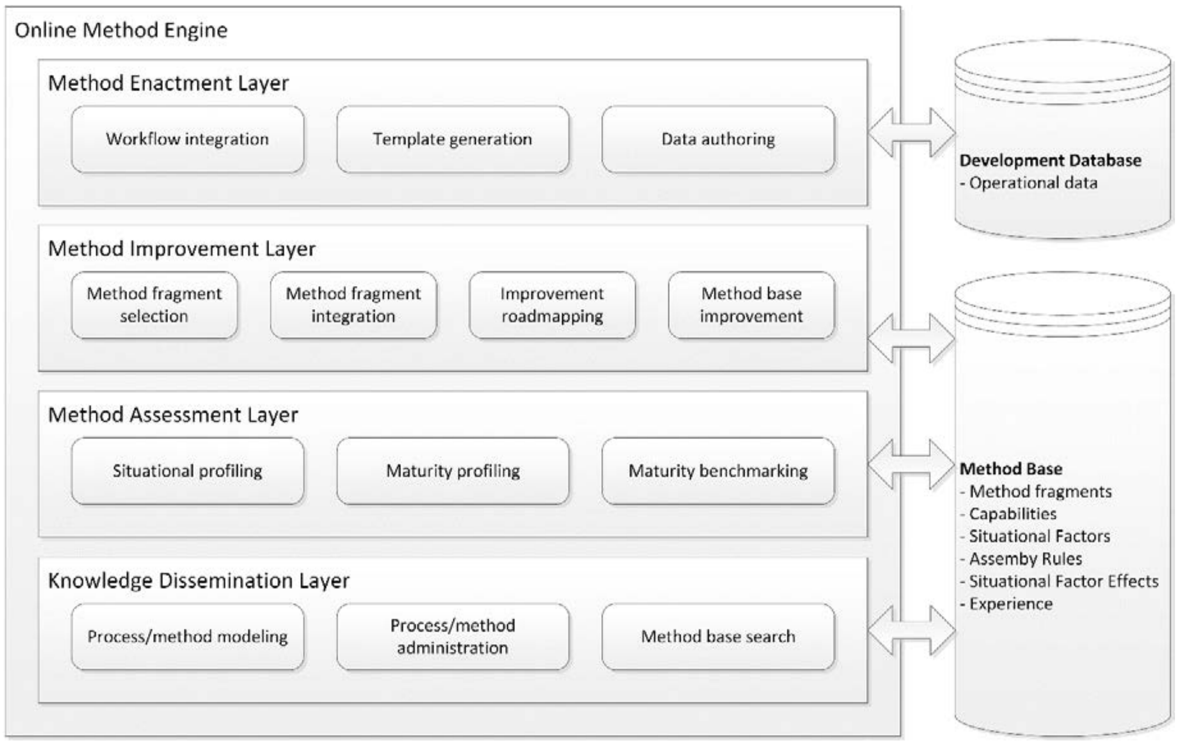


Figure 2. Conceptual Model of the Online Method Engine (Vlaanderen, van de Weerd, et al., 2012)

Vlaanderen, van de Weerd, & Brinkkemper (2012) have proposed a high-level approach (Figure 1) that supports incremental method evolution. This is based on the situational assessment method by Bekkers, Spruit, van de Weerd, van Vliet, & Mahieu (2010), influenced by process improvement approaches such as the Deming cycle of Plan, Do, Check and Act (Deming, 2000), and the Quality Improvement Paradigm (Basili, 1993).

The approach consists of an outer process improvement cycle (*assessment*) that triggers an inner cycle (*improvement*). Process improvement starts with the analysis of the current processes, which outputs in a maturity profile. The organization's situational context (Bekkers, van de Weerd, Brinkkemper, & Mahieu, 2008) is used to determine an optimal maturity profile to reach. The process improvement effort aims to bridge the gap between the current maturity level and the optimal one.

While doing so, suitable method fragments are combined into a new method that is integrated with existing processes in the organization.

The online method engine (Vlaanderen et al., 2013) is a knowledge management system that supports incremental process improvement. The proposal of OME was motivated by the need of a framework that makes available the necessary resources for incremental process improvement (methods, techniques, tools, and templates). The conceptual architecture of the OME is shown in Figure 2. A more detailed description is provided as part of the research contribution presented in this paper.

Problem Statement

The design process of the OME can be described in terms of the Systems Development Research Process (SDRP) by Nunamaker Jr., Chen and Purdin (1990) (Figure 3). The first design has been presented in the form of the Product Software Knowledge Infrastructure (PSKI) that was proposed by Brinkkemper (1996) and van de Weerd, Versendaal, & Brinkkemper (2006). The PSKI included activities that, starting from the analysis of current situation, result in the implementation of changes in the existing process. The conceptual architecture resulted from a new design cycle, incorporating the results from a qualitative evaluation round. This evaluation has been described in an earlier paper (Vlaanderen, van de Weerd, et al., 2012). This resulted in a set of observations that were used to revise the original conceptual design.

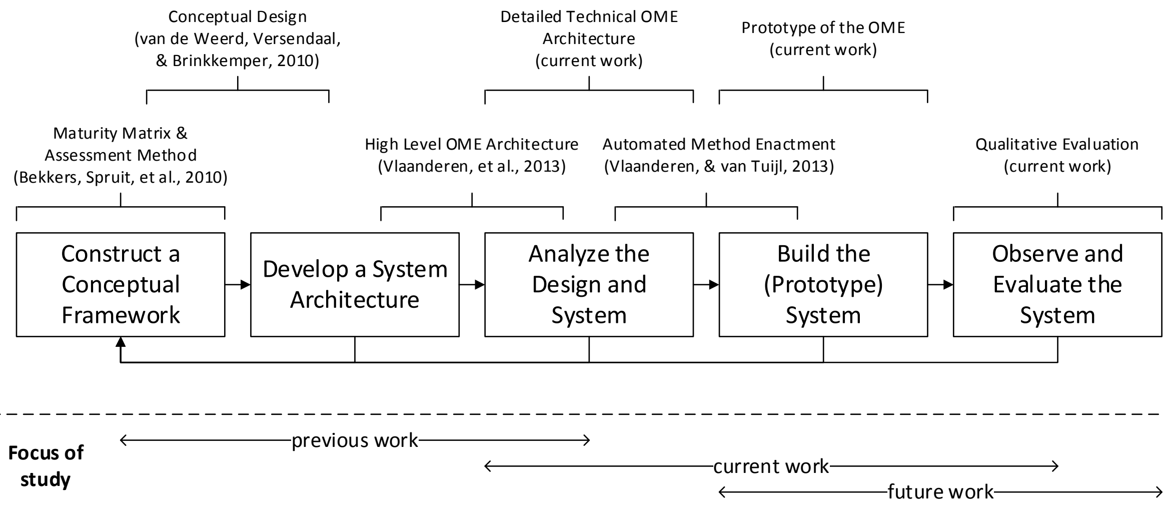

Conceptual Design (van de Weerd, Versendaal, & Brinkkemper, 2010)
Detailed Technical OME Architecture (current work)
Prototype of the OME (current work)
Maturity Matrix & Assessment Method (Bekkers, Spruit, et al., 2010)
High Level OME Architecture (Vlaanderen, et al., 2013)
Automated Method Enactment (Vlaanderen, & van Tuijl, 2013)
Qualitative Evaluation (current work)
Construct a Conceptual Framework
Develop a System Architecture
Analyze the Design and System
Build the (Prototype) System
Observe and Evaluate the System
Focus of study
previous work
current work
future work

Figure 3. Systems Development Research Process, annotated with earlier and current work

While the conceptual architecture was considered as promising (Vlaanderen, Brinkkemper, et al., 2012), our empirical experience has identified a number of major obstacles that affect the field in general, and the applicability of the OME in particular:

- **Process Improvement framework mismatch:** process improvement frameworks are too heavyweight and require very experienced analysts to be effective.
- **Science/Practice gap:** existing approaches are not turned into practical solutions by practitioners, which do not access the literature and deem it as too complex.
- **Low SPM maturity:** most organizations score low on the SPM maturity matrix, and are thus unprepared to develop their own process improvement tools.

The problem that we address here is to overcome the listed obstacles, for they prevent the adoption of method improvement initiatives in (small- and medium-sized) industries. Our proposed solution is to describe a realization of the OME concept, which can be used as such by practitioners, or can be taken as a reference for developing custom frameworks by larger organizations.

Realization of the OME: Static View

In this and the next two sections, we use the 4+1 architectural view model (Kruchten, 1995) to illustrate our realization of the OME concept. In this section, we start with the static view of the architecture, which includes the logical, development, and physical views.

Logical View

This view corresponds to the original conceptual architecture in Figure 2. The OME contains four functional layers. Each layer includes several functional components, shown by the rounded boxes. The approach integrates techniques for sharing knowledge, and assessing, improving, and implementing methods.

The bottom layer is related to knowledge gathering and sharing. In the current approach, methods can be modeled in the form of process-deliverable diagrams using MetaEdit+ (Tolvanen & Rossi, 2003). Any meta-information can be added through the method administration module. In this manner, a method base is built that can be searched for relevant method fragments.

The method assessment layer deals with qualitative analysis of methods using situational profiling and maturity profiling. These profiles can be used for direct benchmarking against other organizations. Based on the results of the assessment, new method fragments can be selected from the method base. These method fragments can be integrated into the current method, and in case of large improvements, an improvement roadmap can be generated.

The top layer—method enactment—includes workflow integration to synchronize method descriptions with actual process data, template generation to update existing tools such as spreadsheets and requirements databases (development database in Figure 2) according to the updated method, and data authoring to manage company-specific process data.

Development View

The OME is implemented on top of the web content management system GX WebManager[1] (Souer & Mierloo, 2008). Its feature set contains all the content management functionality required, e.g. web-based forms, authorization, versioning, and workflow management. At the same time, it is flexible and extensible, for it consists of components and services that are connected at startup time.

Table 1. Data types within the OME

OME Concepts	Data type
Process-Deliverable Diagrams	XML-Documents
Situational Profiles, Maturity Profiles	Relational Data
Alternative Metamodel Formats (e.g., Visio)	Binary Files

Functionality related to the OME is developed as a set of hot-pluggable components and services based on the concept of an open service gateway initiative (OSGi) bundle[2]. These components and services add a set of custom user interface components to the content management system, which allow one to create and maintain a specialist website containing situational method knowledge.

The technical architecture of the OME is based on components and services because, even though the design and implementation are currently guided by the needs of software product

[1] https://www.gxdeveloperweb.com/
[2] http://www.osgi.org/

management research, we aim to ensure generality. The solution is designed to be applicable to other domains by replacing, adding or extending components that are domain-specific, such as the capability model, the list of situational factors, and the available tool connectors.

The individual components' design follows the model-view-controller design pattern and is based on the Spring MVC framework[1]. Most component types have multiple views, so we can provide customizable presentations for the knowledge in the repository, where editors configure component instances in 'edit view' before they get published.

The research data is a mixture of XML-documents, objects, relational data and possibly binary files (see Table 1 for an overview of the main data types currently handled by the OME). The Java Content Repository (JSR-170)[2] implementation in WebManager can handle these different types of data through a single, unified API.

WebManager (Souer, Joor, Helms, & Brinkkemper, 2011; van Berkum, Brinkkemper, & Meyer, 2004) builds on standard technologies and products (e.g., Java, Apache Tomcat, Apache Felix, MySQL) that can be scaled easily. This enables the gradual extension of the OME from a small research prototype to a publicly available resource.The OME depends on various WebManager services, merges with its user interface, and builds on the underlying repository.

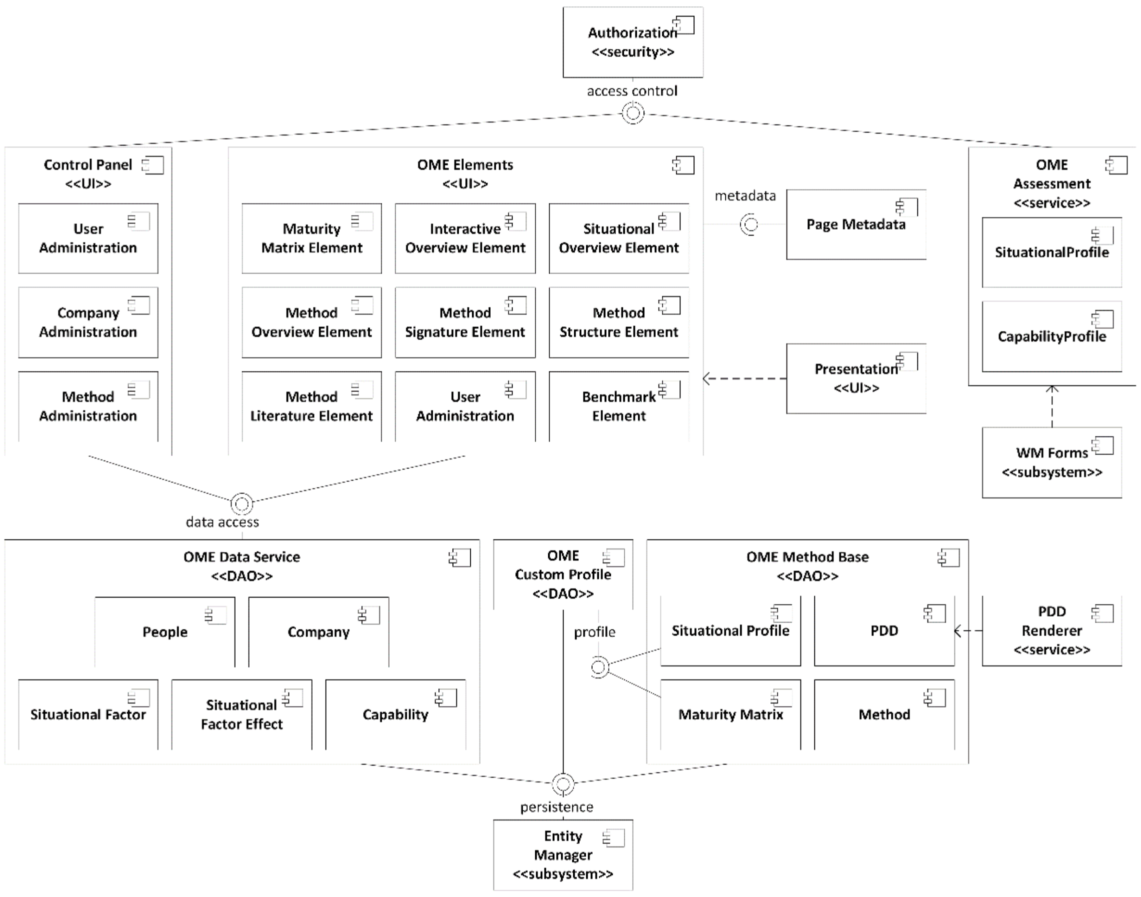


Figure 4. Component diagram of the OME

The users, organizations, and methods can be maintained through a set of UI components. These components access the data layer through a set of Data Access Objects (DAOs) that is connected to an entity manager. Another set of UI components is used to implement all functionality

[1] http://www.springsource.org/
[2] http://jcp.org/en/jsr/detail?id=170

related to the scenarios described below. This set can be used by a website editor to publish selected information from the method base on the website in a suitable form. The assessment results are handled by a set of services that link the data in the method base to an extended user profile that is capable of maintaining all data relevant to a specific user or organization. The UML 2 component diagram (ObjectManagementGroup, 2005) in Figure 4 shows all relevant components and services. The diagram includes both generic WebManager specific components, as well as OME specific components.

Physical View

The implementation of the OME conceptual architecture addresses issues that go beyond the domain of process improvement. An important goal of our implementation is to provide a repository of method fragments and assessment results that serves to improve the overall quality of software product management (SPM) process and increase the understanding of best practices and the characteristics of process improvement efforts.

The conceptual architecture does not detail where this data resides and who has access to it. Our implementation differentiates between data in the public domain and data in the private/corporate domain. The former category includes generic fragments from the literature, analytical tools (e.g., capabilities and situational factors), and rules originating from empirical research that relate these components (e.g., situational factor effects). The latter category consists of organization-specific data, such as enacted methods and assessment results.

By design, the OME's functionality goes beyond mere process improvement support. An important goal is the acquisition of empirical data that can be used to infer hypotheses related to successful combinations of situational context and method fragments. This introduces privacy- and competition-related issues. Although many organizations are willing to provide confidential data for academic purposes, there are concerned with safeguarding their competitive position. This concern is reflected in the physical design of the OME.

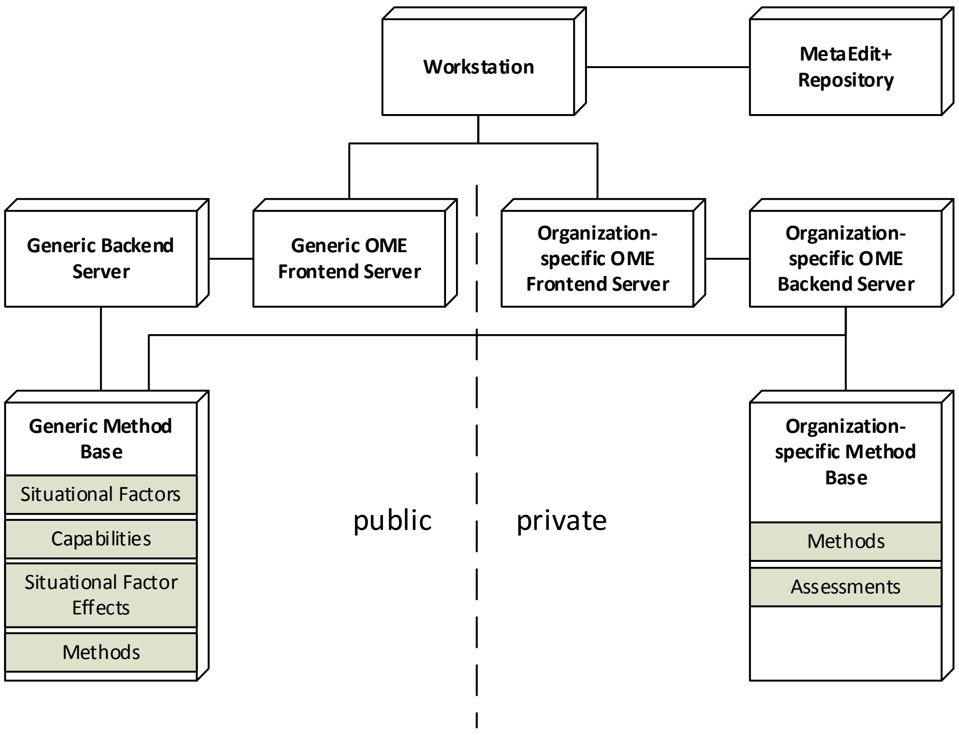

Workstation
MetaEdit+ Repository
Generic Backend Server
Generic OME Frontend Server
Organization-specific OME Frontend Server
Organization-specific OME Backend Server
Generic Method Base
Situational Factors
Capabilities
Situational Factor Effects
Methods
public
private
Organization-specific Method Base
Methods
Assessments

Figure 5. Deployment diagram of the OME

The deployment diagram in Figure 5 shows our implementation, with the major nodes and their relationships. The dashed vertical line determines the boundary between the publicly available nodes (left-hand side) and the private organization-specific nodes (right-hand side). There are two types of frontend and backend servers, public and private. The organization-specific backend server connects not only to the organization-specific method base (from retrieving private methods and assessments), but also to the public method base, that contains generic situational factors, capabilities, situational factor effects, and methods.

Mapping to Conceptual Architecture

As described above, one of the goals of this paper is to demonstrate how we implemented the conceptual architecture shown in Figure 2. The static views detailed in the previous sections provide a thorough overview of the various techniques we used and the design choices we made. In order to link the previously designed conceptual architecture to its technical counterpart, we have outlined each conceptual component to the relevant technical components in Table 3.

The left-hand side of the tables lists all conceptual components of Figure 3. The right-hand side shows the techniques that we employed to satisfy the requirements for that component. While some of these techniques link directly to components shown in Figure 4, others refer to external tools or techniques from the method engineering domain. For example, several components refer to the Situational Assessment Method, described by Bekkers et al. (2010).

Table 2: Mapping the conceptual architecture to our prototype realization

Conceptual component	Implemented through
Method Modeling	MetaEdit+ / Visio
Method Administration	Method Administration Panel
Method Base Search	Method Overview Method Base Facetted Search
Situational Profiling	Situational Profile (SAM)
Maturity Profiling	Capability Profile (SAM)
Maturity Benchmarking	Capability Profile with Aggregated Data
Method Fragment Selection	Areas of Improvement Matrix (SAM) Method Base Facetted Search
Method Fragment Integration	Method Assembly Techniques
Improvement Roadmapping	Method Increment Planner
Method Base Improvement	Situational Profile (SAM) Capability Profile (SAM) Feedback Form
Workflow Integration	Method Enactment Mechanism Link with CASE tools
Template Generation	Method Enactment Mechanism
Data Authoring	Not implemented yet

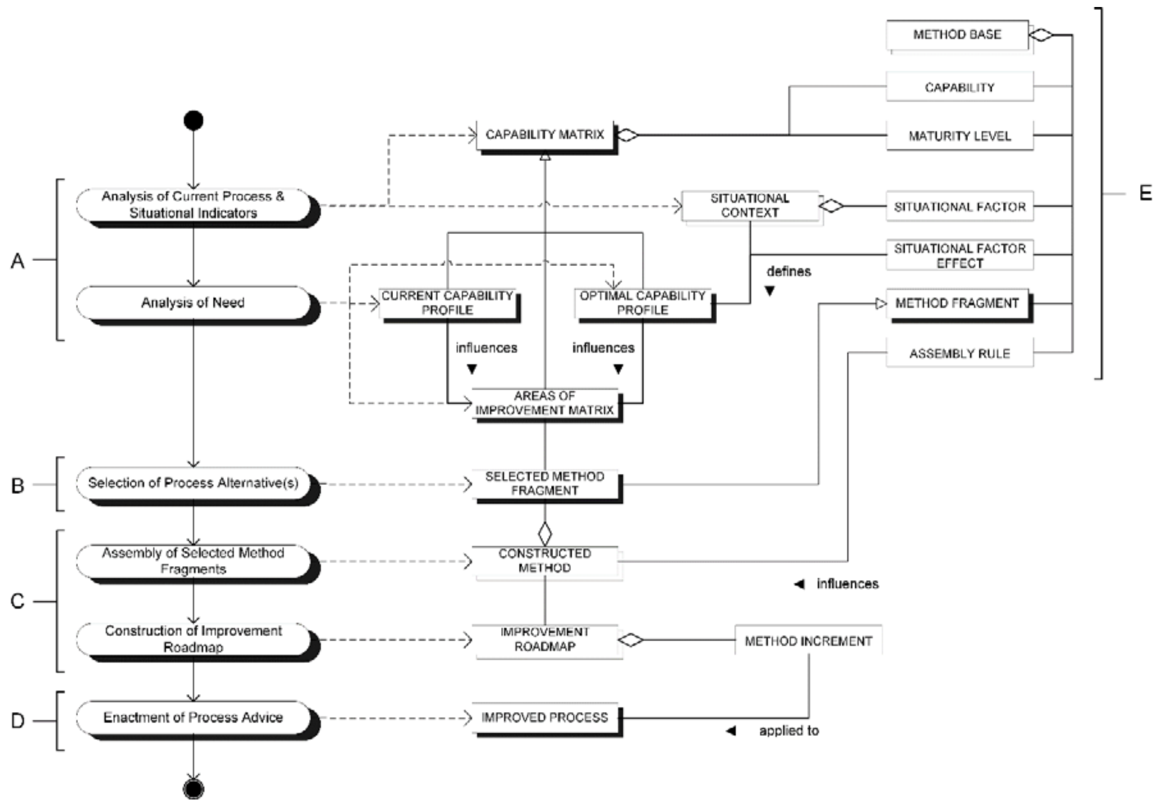
In the following sections, we describe the role that these components play during a process improvement activity in more detail.

Realization of the OME: Dynamic View

We describe the dynamic aspects of our implementation. We first present the overall process view, and then focus on the five main usage scenarios that the OME allows for.

Process View

The OME supports a set of distinct goals that practitioners are interested in achieving during process improvement. These goals are tightly related to the layers in the logical view, and are in line with the incremental process improvement paradigm in Figure 1. We describe how the process view—illustrated via the process-deliverable diagram (PDD) (van de Weerd & Brinkkemper, 2008) in Figure 1—realizes these goals.



A
B
C
D
E
Analysis of Current Process & Situational Indicators
Analysis of Need
Selection of Process Alternative(s)
Assembly of Selected Method Fragments
Construction of Improvement Roadmap
Enactment of Process Advice
CAPABILITY MATRIX
SITUATIONAL CONTEXT
CURRENT CAPABILITY PROFILE
OPTIMAL CAPABILITY PROFILE
influences
influences
AREAS OF IMPROVEMENT MATRIX
SELECTED METHOD FRAGMENT
CONSTRUCTED METHOD
IMPROVEMENT ROADMAP
IMPROVED PROCESS
METHOD INCREMENT
influences
applied to
defines
METHOD BASE
CAPABILITY
MATURITY LEVEL
SITUATIONAL FACTOR
SITUATIONAL FACTOR EFFECT
METHOD FRAGMENT
ASSEMBLY RULE

Figure 6. Meta-Process View of the OME

The figure outlines the processes, the artifacts that are created and used, and their interdependencies and relations. The process side describes a sequential path through the functionalities of the OME. However, complexity arises from the fact that each step is typically performed iteratively, and that the results of each step result in modifications of the data in the method base through the use of feedback mechanisms.

Method Assessment

The OME process starts with the analysis of the current processes and situational indicators. This activity results in filling in a capability matrix and a situational context. This is followed by an analysis of the need, which determines the current capability profile, the optimal profile to be achieved through the process improvement effort, and the areas of improvement. These processes are detailed through Scenario A below.

Method Discovery

The process follows with a selection of process alternatives for reaching the optimal profile. This requires the manual discovery of relevant method fragments that are compatible with a specific organizational context, as described in scenario B.

Method Improvement

The chosen fragments are assembled within a constructed method, and the definition of an improvement roadmap that defines a plan for integrating the new fragments. This part of the OME process is detailed in Scenario C.

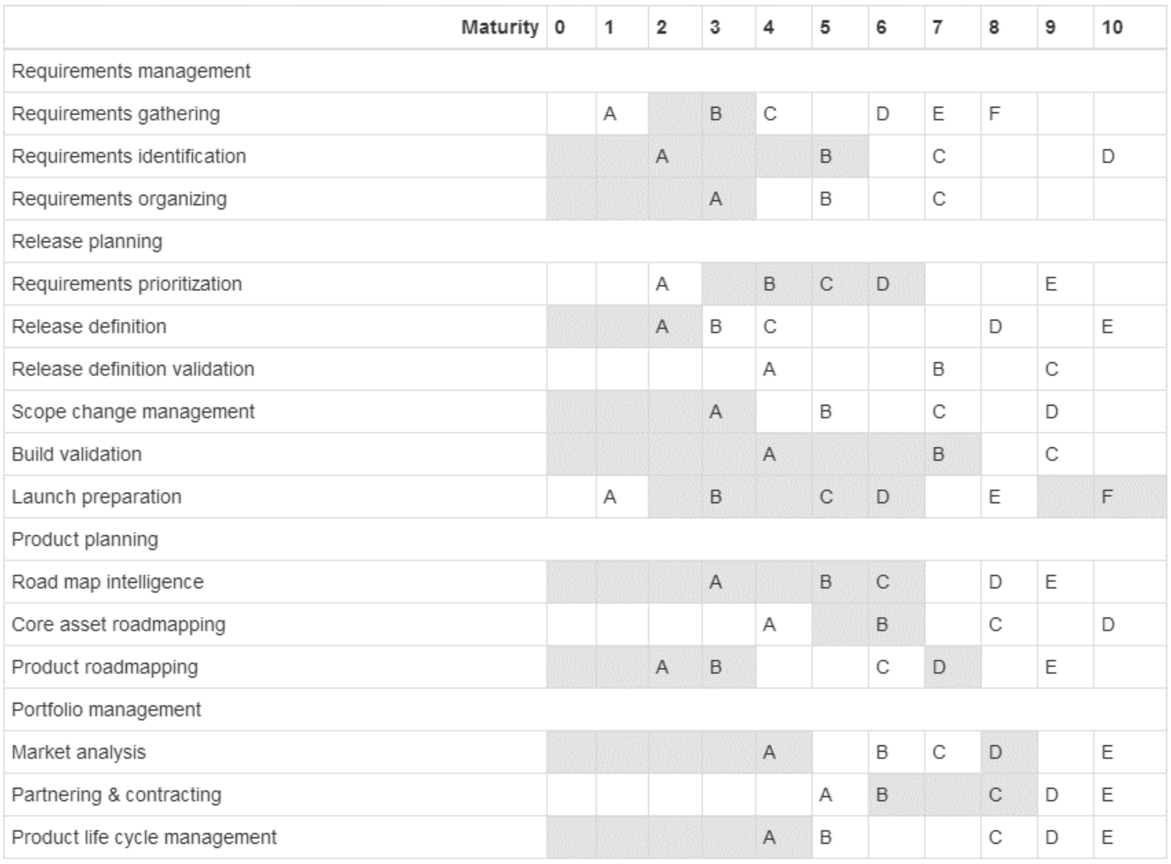
Method Enactment

The OME main process terminates with the enactment of process advice, which results in the improved process. The process should lead to a situation in which the organization meets (or is very close to) the optimal capability profile. This phase is described in Scenario D.

In addition to the processes in Figure 6, the OME also supports a general-purpose process that concerns the administration of methods. This is detailed in Scenario E below.

Scenarios

We detail the five major usage scenarios of the OME. They are closely tied to the layers in the OME conceptual model, and further refine the process view described above.



Maturity	0	1	2	3	4	5	6	7	8	9	10
Requirements management											
Requirements gathering		A		B	C		D	E	F		
Requirements identification			A			B		C			D
Requirements organizing				A		B		C			
Release planning											
Requirements prioritization			A		B	C	D			E	
Release definition			A	B	C				D		E
Release definition validation					A			B		C	
Scope change management				A		B		C		D	
Build validation					A			B		C	
Launch preparation		A		B		C	D		E		F
Product planning											
Road map intelligence				A		B	C		D	E	
Core asset roadmapping					A		B		C		D
Product roadmapping			A	B			C	D		E	
Portfolio management											
Market analysis					A		B	C	D		E
Partnering & contracting						A	B		C	D	E
Product life cycle management					A	B			C	D	E

Figure 7. Example method assessment result in the OME

Scenario A: Method Assessment

The method assessment in OME provides a structured overview of the quality of an organizations SPM process, providing the hammer and screwdriver during a method improvement effort. During the last few years, several of the concepts related to method assessment and improvement in SPM have been proposed and elaborated. Bekkers, van de Weerd, Brinkkemper, & Mahieu (2008) suggest that the general assessment of an organization or a specific business unit within that organization can be performed using a set of situational factors. The assessment of an organization's SPM methods can be performed using a set of capabilities, which have been obtained through extensive empirical analysis (Bekkers et al., 2010). Based on the results of these two assessments, a maturity matrix can be determined that outlines the current maturity of the organization.

Scenario B: Method Discovery

Each method is described in a separate page of the system. These pages contain all data relevant to the particular method, as described in the previous scenario. In addition, several tools are provided that generate overviews of the available methods. With these tools, the method base becomes a fully-searchable database of method knowledge, allowing method administrators or process owners to quickly find relevant methods.

The main entry point for this activity is a form that allows facetted search, enabling to user to filter the available methods based on business function, situational relevance, implemented capabilities, and keywords. Throughout the system, users can perform contextual searches that result in an overview of methods that are relevant to the current system state. For example, during the improvement activity, users can easily access methods that are similar to the methods that have been proposed by the system, in order to gain a more complete view and enabling them to make well-informed choices.

Scenario C: Method Improvement

The assessment results that are obtained during scenario A are the main input for the improvement activities supported by the OME. In the current implementation, these activities include the selection of method fragments that could improve the assessed process, the integration of the selected method fragments into the current process, and the creation of a roadmap that supports the enactment of the improved process description.

A selection of relevant method fragments is determined based on the situational profile of the organization in combination with the set of missing capabilities. This set is the delta between the currently implemented capabilities and the capabilities at or below a maturity level indicated by the user. This implies an explicit choice regarding the relevant business functions and the goal maturity level.

The system queries the method base using the set of requested capabilities and the situational factors as constraints. The results are shown to the user in summarized form. When multiple fragments are available with a similar signature (i.e. similar capabilities and situational constraints) then the user is given a choice between these fragments. Based on the meta-information of the method fragments, such as description and rationale, the user can select the fragments that he deems suitable.

If the user has provided a process description of the current process in the form of a PDD, preferably with the help of a method engineer, then the selected method fragments can be integrated into that process. This integration is performed based on the dependencies between the deliverables described by the method fragments. Any issues related to this integration, such as duplicate capabilities and incompatible method fragments are reported to the user, who can then make appropriate changes to the set of selected method fragments.

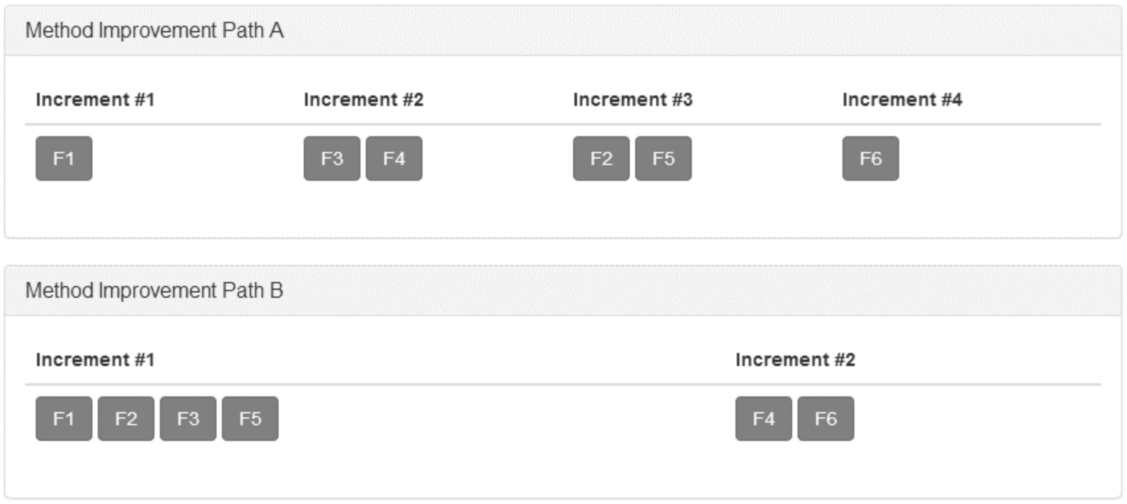


Figure 8. Method improvement paths

The integrated process is then broken down again into atomic pieces based on the dependencies between deliverables. The resulting set of atomic fragments is prioritized based on factors such as the maturity level of the capabilities they relate to. This prioritized list is used to determine one or more paths that support the implementation of the proposed changes, as illustrated by Figure 8. These paths (depicted by the large boxes) consist of multiple increments (depicted by the columns) each containing a set of one or more fragments (depicted as dark boxes), allowing an incremental improvement style. A sidebar shows details regarding the changes contained within each steps, such as added or removed activities and deliverables.

Scenario D: Method Enactment

Enacting process improvements is a complex undertaking influenced by many factors such as organizational culture, technological constraints, and management support. Such a problem cannot be solved by a tool, as it is largely a social challenge. However, we believe that the OME can provide some support related to the enactment of changes from a technological perspective. For this purpose, the OME incorporates a mechanism for translating the method changes to tool changes.

The mechanism requires a mapping between the system and one or more tools that are used within the process. Each increment is translated to a set of changes to the tool. Examples of such changes are added screen or changed properties to specific models. The results of each transformation are displayed to the user for review. A detailed description of this process is provided in the next section.

Scenario E: Method Administration

The method base contains the nails and screws of the OME. It is a fundamental concept of method engineering, and it consists of a set of method fragments that can be combined into a mature and situational SPM method, in addition to experience related to the applicability and usability of those method fragments.

In the OME, new methods enter the method base through an administration panel. Each method is described in terms of a name, a goal, a free-format description, the situation in which it is applicable, the capabilities that it helps implementing, and relevant literature.

A method can be further detailed in terms of a PDD. When this is the case, the method administrator separately draws the diagram using the MetaEdit+ modeling tool, which can be imported into the OME (see the spotlight), and linked to the appropriate method.

Spotlight: Administration of Process Deliverable Diagrams

A core component of the underlying method base is formed by PDDs, which allow for a detailed representation of the structure and flow of method fragments. They constitute a solid tool for method assembly and the planning of method improvement. We outline how PDDs are created and used in our toolset (see Figure 9).

The metamodeling tool MetaEdit+ (J.-P. Tolvanen & Rossi, 2003) is the main tool for constructing and maintaining PDDs. It exports PDDs to an XML-based file format, which is then imported into the OME.

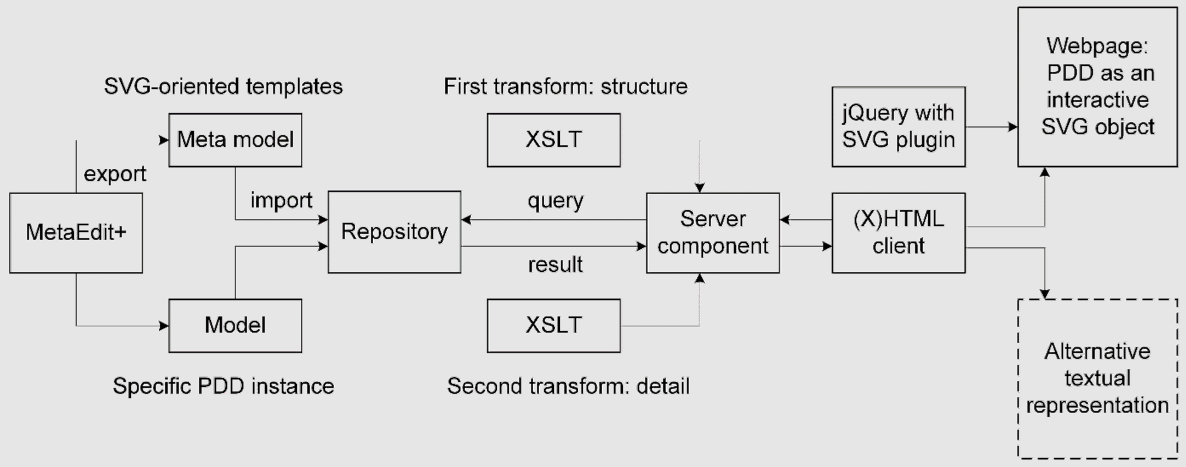


Figure 9. PDD processing: from MetaEdit+ to OME

MetaEdit+ exports two documents: one generic meta-model and one PDD specific model. The meta-model is stored in the OME repository once. Based on the meta-model, incoming models are transformed in two passes to embeddable SVG diagrams. The first pass deals with the graphical positioning of the elements in the diagram. The second pass refines the diagram and fills in text values. Once the diagram is used on a page within the OME, additional behavior is attached to the diagram through JavaScript, making the diagram interactive. In addition, different representations can be generated based on the information in the model (activity description tables, deliverable lists, etc.).

Technical Zoom-In: Method Enactment

We detail how the incremental method enactment components of the OME realization have been implemented by interfacing with the Jira[1] requirements management tool. Jira integrates a workflow engine that we use to support method enactment.

A thorough explanation of the process that we followed to select an appropriate SPM tool is provided by Vlaanderen & Tuijl (2013). The main criteria for our choice were the support of workflow management, the capability to support documents, simple-to-use configuration, modifiability, and that it is widely known and used in the industry.

Conceptual mapping

The mapping from PDD method fragments to Jira's data model is shown in Figure 10. This mapping is essential to allow using Jira's workflow engine for method enactment.

[1] Jira is an issue tracking system, and its data model is centered on the notions of issue and issue types (http://www.atlassian.com/software/jira/).

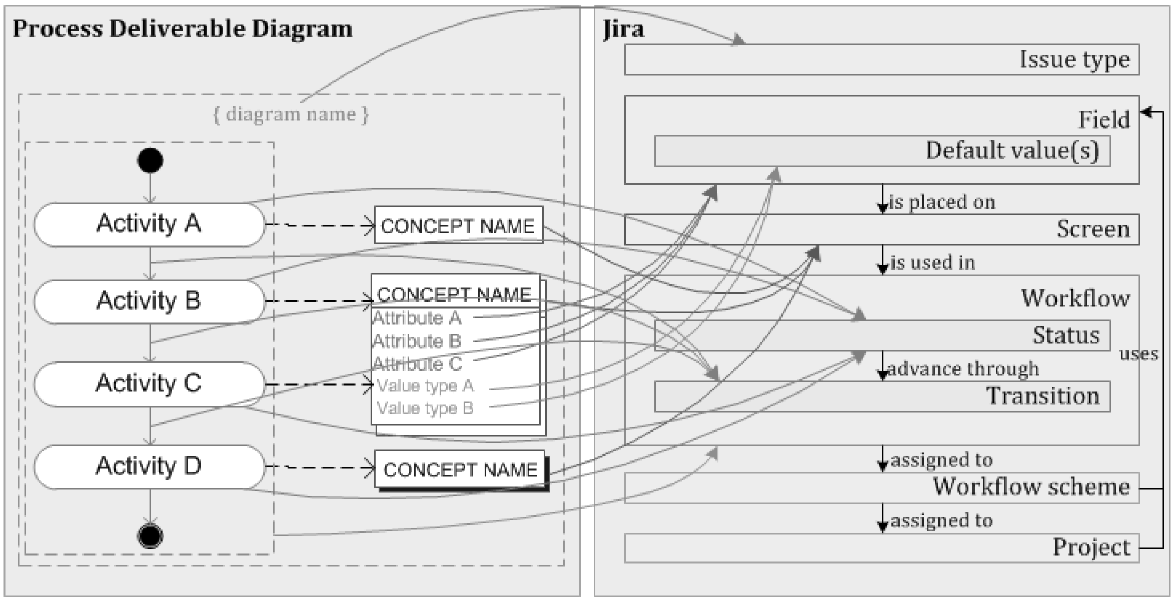


Figure 10. Mapping of PDDs to Jira's data model

The left-hand side of Figure 10 shows a simplified PDD fragment. The right-hand side of the same figure shows the essential elements of Jira's data model. The arrows link the method fragment meta-model to the data model. Table 4 provides a textual explanation of the relationships between both fragments.

The diagram name of the fragment that a user creates is used to create an issue type in Jira. For instance, an initial PDD for a fragment called 'Requirement management process' would result in an issue type in Jira. All the data that is entered in Jira in relation with this issue type belongs to that specific fragment. When the PDD of the fragment is updated, the existing issue type will not be affected. Issue types can be used in multiple projects.

The complete process side of the fragment is mapped to a workflow in Jira. A workflow is represented by a name and the entire workflow is based on an XML structure. Structurally, workflows in Jira are similar to processes in PDDs. A workflow consists of a status (of an issue) and transitions between statuses. A status is a step in the workflow through that, when completed, advances the workflow. For instance, after completing activity A in a workflow and marking this as complete, the workflow advances to next status (i.e., the next activity). Each status is linked to a screen so that a status can present information to the user.

Jira advances from a status to the next by following the path defined by transitions. PDD transitions are mapped 1-to-1 to Jira transitions. The name of the transition is based on the upcoming activity in the PDD.

Table 3. Mapping of Method Fragments to Jira

#	PDD Fragment	Relationship	JIRA Fragment	Usage Domain
1	Diagram name	is used to create an	Issue type	per project
2	Process side	will produce a	Workflow	per project
3	Activity	is used to create a	Status	in a workflow
4	Transition	is used to create a	Transition	in a workflow
5	Concept	is used to create a	Screen	per project
6	Attribute	is used to create a	Field	per project
7	Value type	will be used as (a)	Default value(s)	per field

The name of a PDD concept is used to create a screen in Jira. For instance, given a concept called REQUIREMENT, a screen is created with the name REQUIREMENT. A screen contains fields that are based on attributes of the PDD concept. If a concept does not contain attributes, the screen does not have any input fields. The screen is linked to a status that can contain many other functions such as attaching a document or adding comments to a particular status.

Technical description

After a fragment has been created in MetaEdit+, a MERL script (the MetaEdit+ scripting language; J.-P. Tolvanen & Rossi, 2003) exports the fragment to an external tool. This tool extracts the relevant meta-data from the fragment, including the fragment name and its version, and converts the fragment to an intermediate structure based on the PDD meta-model as defined by van de Weerd, Brinkkemper, & Versendaal (2007).

Once all relevant data has been extracted, it is shown in a user friendly manner to the user, who can verify data consistency. Once the data is verified, the enactment process is triggered. For each relevant database table of Jira, three objects are created based on the Model-View-Controller (MVC) design pattern; a Value Object (VO), a Data Access Object (DAO), and a Controller Object (CO). These objects form the connection between MetaEdit+'s data structure and Jira's data structure.

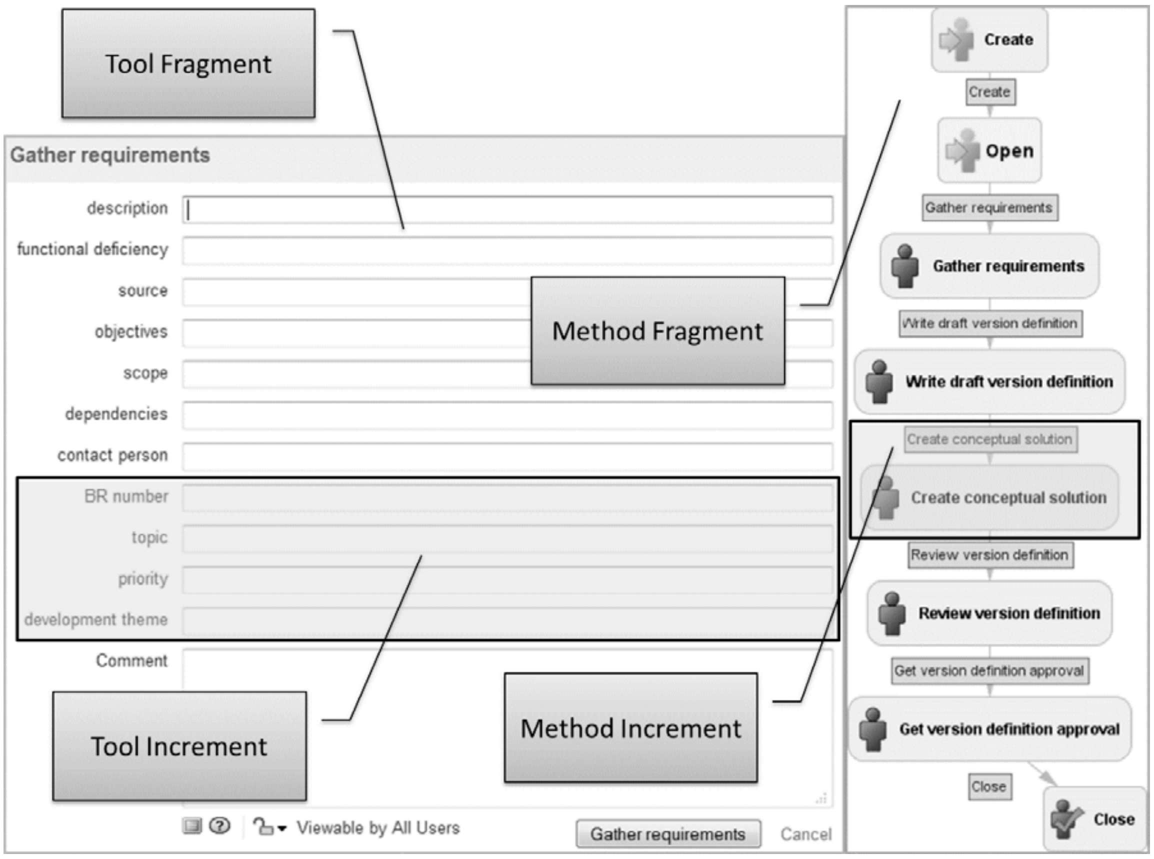


Figure 11. Adapted Method Fragment in Jira

Through a set of prepared SQL statements, the data fragment is then stored in Jira's configuration database. Prepared statements are used to increase safety and avoid SQL injection (i.e. attacking the database deliberately through query manipulation). We used PDO in conjunction with MySQL 5.0 because this allows us to abort the entire querying execution process when a single query fails (it supports transactions with commit/rollback commands).

Figure 11 shows the method increment and tool increment based on increment #4 in van de Weerd, Brinkkemper, & Versendaal (2010). On the left side of Figure 11, the fields for the respective concepts of increment #4 are shown. The fields in Jira are based on the attributes of the concepts REQUIREMENTS DOCUMENT and REQUIREMENT. On the right hand side of Figure 11, the workflow is shown that is in accordance with the workflow in fragment #4. The shaded areas denote the method and tool increment.

Evaluation

Any design activity is a process involving multiple iterations of design generation and design testing (Hevner, March, Park, & Ram, 2004). The work described in this paper represents one of these cycles. We report here on a qualitative evaluation of different parts of our approach: the conceptual design, the overall implementation, and the enactment.

Expert Evaluation of the Prototype

Definition

For this evaluation, we are mainly interested in the opinion of practitioners regarding the implementation, since the conceptual design has been evaluated in earlier work, as described above. For this reason, we structured the interviews around the following questions:

- How well does the OME implementation support the described scenarios?
- Is adequate information provided at key points?
- Is this an effective approach for searching, visualizing, and managing method fragments?
- Would this concept be feasible in a real-life setting?

Planning

We selected five product managers from product software organizations in the Netherlands for the evaluation of the current implementation of the OME. These product managers were not previously familiar with the OME concept.

Execution

The evaluation is structured around the five scenarios described above. We interviewed the participants in separate sessions, to prevent any contamination of opinions. Each session lasted approximately 45 minutes. First, the main concepts were explained. Then, the implementation was demonstrated according to the five scenarios. After each scenario, the participant’s interpretation was checked against the original intent. Once it was verified that the interpretation and intent were aligned, we obtained feedback through a semi-structured interview.

During the interviews, the participants were encouraged to provide additional comments, in response to the screens that they were shown. This provided us with some interesting ideas and insights for future design iterations.

Results

Most participants envisioned the method base as a collaborative system, where practitioners could share experience and thus develop a higher standard of product management processes and perhaps some sort of standardization. None of the participants opposed the idea of sharing these experiences, on the condition that all information would be anonymized.

The style of continuous, iterative improvement was in general highly appreciated. The overall process (assessment, interpretation, improvement, and enactment) was deemed relevant and realistic, and the fact that process owners would have appropriate tools for process improvement was much appreciated. They also liked very much the possibility of retrospection, i.e. looking back at earlier assessment and reflecting on the changes. This confirms the results from our earlier study (van Stijn, Vlaanderen, Brinkkemper, & van de Weerd, 2012). The participants indicated a wish for improved support in this area, in the form of a historical overview of earlier assessments and improvements.

Scenario A

The assessment forms posed no significant problems to the participants, but the maturity matrix was found difficult to interpret. We showed a condensed version, which everyone found easy to interpret, and an expanded version, which is the original form and shows more detail, but which was harder to interpret. The interviewees suggested introducing improved visual aids such as a legend and detailed information regarding the capability levels.

Scenario B

With respect to the method base and the way in which method fragments are presented, the participants expressed concerns regarding the origins of method fragments. A common perception was that high-quality method descriptions would be critical to any successful implementation of the OME. In addition, some participants indicated the need for more detailed data such as common usage scenarios of the method fragments, templates, roles, and meta-data such as complexity.

Scenario C

The main concern regarding the improvement scenario was that participants were not convinced that the system would be capable of reflecting and incorporating the complex situation of ‘the field’. Soft factors such as team culture, costs, and the existence of ad-hoc processes were indicated as confounding factors that were insufficiently addressed in the design.

Regarding the improvement roadmap, the experts mentioned the need for more detailed information concerning the contents of each improvement step as well as the rationale behind the scenarios. This last point was related to the need of prioritization of improvement suggestions, as indicated by a few participants. A recurring comment was the importance of the distinction between quick wins and improvements with a larger impact on resources.

Scenario D

The enactment mechanism was only briefly discussed during this evaluation round. Due to its complexity, this part was evaluated in a separate study. The results are described in a separate section below.

Scenario E

The administration of the method base was not deemed relevant to the profile of the participants. Therefore, we focused on functionality related to the discovery and usage of knowledge.

Interpretation

The results of this evaluation round provide an early, high level of the areas of the system that require thorough analysis during further elaboration. A limitation during any prototype evaluation is the expectancy of the participant, for whom it is difficult to separate the proposed functionality and workflow from the visual design aspects. The evaluation results of scenario A and B suggest the need for an analysis of information needs. However, no major architectural changes seem to be required.

Feedback for all scenario’s confirmed the earlier observation that practitioners are skeptical towards suggestions provided by automated, knowledge-based reasoning. A real life setting will always be constrained by conflicts amongst stakeholders, limited resources, and the need to balance short and long term goals. In the context of method fragment integration and planning, participants indicated that the proposal has merits, but that they need assurance that the results are reliable and robust.

Technical Evaluation of the Enactment Mechanism

Definition

The goal of the technical evaluation of the enactment mechanism is to obtain concrete insight into the capabilities and shortcomings of the current solution. We do so by simulating a series of successive method increments and their enactment in the SPM tool as described above.

Planning

The data for this evaluation consists of a base method, and four method increments that describe changes to this base method. To emphasize the notion of mimicking a real-life setting, we selected the base method and method increments from a previously conducted case study (van de Weerd et al., 2010) at a vendor of ERP software (see Table 5 for an overview). The increments show the evolution of a requirement management method that advances from a very simple method to a more elaborate approach. The increments, along with a textual elaboration of each increment, can be found in the paper by van de Weerd et al. (2010b).

Table 4. Overview of Method Increments

INCREMENT	**GOAL**
0	Introduction requirements document
1	Introduction design document
2	Introduction version definition
3	Introduction conceptual solution
4	Introduction requirements database, division market and business requirements, and introduction of product families

Execution

We executed the enactment mechanism with the method increments listed above as input. After enacting each method increment, we compared the resulting tool configuration to the method description. Although the enactment tool is deterministic in nature, this approach allowed us to discern subtle shortcomings in both the realization of the tool as well as the constraints that can be posed by an industrial tool.

Results

During the development of the prototype, we encountered some issues that still inhibit a fully automated approach to method enactment.

- **Standardization of Process Deliverable Diagrams.** While the PDD format has been clearly defined and formalized (van de Weerd & Brinkkemper, 2008) in practice we encountered different dialects of PDDs. This is a limitation in that many existing fragments cannot be transformed, due to their deviation from the standard.
- **Support for abstraction levels.** PDD's support the visualization of different levels of aggregation and abstraction within the process. The ability to implement these levels highly depends on the workflow engine. This has been an obstacle in Jira; as a result, our technique currently does not support all types of activities.
- **Usage of forks, joins and decisions.** The use of forks, joins and decisions is very common in processes. However, many workflow engines do not fully support these constructs. While it supports forks and joins, Jira does not explicitly support decision points; fortunately, one can add conditions to transitions to mimic this behavior.

- **Generalizations, associations and aggregations.** We could not find a way through which we could make sound use of generalizations, associations and aggregations.

Expert Evaluation of the Enactment Mechanism

Definition

Our interviews focused on assessing the rationale behind the approach, the feasibility of the enactment mechanism, and the required adjustments.

Planning

We conducted 5 semi-structured interviews with product managers from different medium/large-scale organizations in the Netherlands. These participants were not the same people who evaluated the OME prototype.

Execution

We posed a total of 14 questions. 7 of these were aimed at eliciting information directly related to the companies' current situation. The other 7 focused on the evaluation of the enactment mechanism. A demo of the enactment mechanism was shown to the product manager in the form of a video that showed the mechanism in action[1]. During the video, a verbal explanation was provided to make things clearer to the product manager. At the end of each interview, time was allocated to document any kind of other information that was provided by the product manager, i.e. remaining concerns and/or opinions.

Results

Each of the interviewees agreed on the fact that process adaptation is an important and continuous activity throughout the organization or department. Each organization struggled in the beginning with their business process(es), and each company reinvented the wheel numerous times before a standard process was in place. Out of the 5 companies, 4 had process models in place, mostly at a high level of abstraction. The remainder company did not have any model in place due to its small size. Over time, each company shifted from having a very detailed process model to a more high-level process.

The interviewees agree that a major issue with the current approach is rigidity. A successful enactment mechanism needs to be flexible, supporting non-linear, complex methods, and allowing exceptions. This includes the support for decisions, multiple stakeholders, and a flexible stance towards the relation between the process model and reality. Most interviewees agreed that any enactment approach that forces people to work in a specific way would be met with much resistance, which is a result that is in line with recent work in software process improvement (Baddoo, 2003; Sulayman, Urquhart, Mendes, & Seidel, 2012).

The interviewees also commented on the fact that the current approach does not deal with all possible process model changes. It focuses mainly on the addition of steps and/or deliverables, while leaving out the deletion of process elements.

Conclusions and Outlook

We described the toolset that implements of the Online Method Engine concept and reported on the feedback that we obtained through semi-structured interviews with practitioners that had experience with process improvement.

The experience that is reported in this paper—which consists of technical mechanisms, design choices, and feedback from the field concerning the OME—constitutes an important source of information for researchers and practitioners in the field of process improvement. We envisage that

[1] The video can be downloaded from http://www.staff.science.uu.nl/~vlaan107/enactment_demo.wmv

this information can be leveraged in order to reuse successful design choices and technologies, and to create awareness of possible limitations and obstacles.

Our future work aims to evolve the OME implementation from a research prototype to a product that can be effectively used in the industry. This will involve identifying which technologies can be reused, and which ones need to be replaced; building a large public method base, also through incentives for organizations to contribute; and integrating our toolset with existing information systems and knowledge management systems, in order to facilitate its adoption.

The evaluation that we performed in the context of this research has a broad scope, yielding high-level results. These results are useful in guiding future research, mainly in the area of method fragment integration and planning. As we proceed implementing the Online Method Engine, frequent and thorough evaluation of each subsystem is required in order to alleviate the issues identified in this study.

Acknowledgement

The authors would like to thank Dr. Inge van de Weerd (Vrije Universiteit Amsterdam) and Geurt van Tuijl for their contributions to the design and implementation of the OME.

Bibliography

Baddoo, N. (2003). De-motivators for software process improvement: an analysis of practitioners' views. *Journal of Systems and Software*, *66*, 23–33.

Basili, V. R. (1993). The Experience Factory and its relationship to other Improvement Paradigms. In *Proceedings of the European Software Engineering Conference* (pp. 68–83).

Becker, J., Janiesch, C., & Pfeiffer, D. (2007). Reuse Mechanisms in Situational Method Engineering. In *Situational Method Engineering: Fundamentals and Experiences* (Vol. 244, pp. 79–93).

Becker, J., & Knackstedt, R. (2007). Configurative method engineering–on the applicability of reference modeling mechanisms in method engineering. In *Proceedings of the America Conference on Information Systems* (pp. 1–12).

Bekkers, W., Spruit, M., van de Weerd, I., van Vliet, R., & Mahieu, A. (2010). A situational assessment method for software product management. In T. Alexander, M. Turpin, & J. van Deventer (Eds.), *Proceedings of the European Conference on Information Systems* (pp. 22–34). Pretoria, South-Africa.

Bekkers, W., van de Weerd, I., Brinkkemper, S., & Mahieu, A. (2008). The Influence of Situational Factors in Software Product Management: An Empirical Study. In *Proceedings of the International Workshop on Software Product Management* (pp. 41–48). Washington, DC, USA: IEEE Computer Society.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, *38*(4), 275–280.

Brinkkemper, S., & Harmsen, F. (1995). Configuration of situational process models: An information systems engineering perspective. *Software Process Technology*, *913/1995*, 193–196.

Brocke, J. vom, & Sinnl, T. (2011). Culture in business process management: a literature review. *Business Process Management Journal*, *17*(2), 357–378.

CMMI Product Team. (2002). *Capability Maturity Model Integration (CMMI). Engineering* (p. 647). Pittsburgh.

Deming, W. E. (2000). *Out of the Crisis* (p. 523). MIT Press.

El Emam, K. (1997). *SPICE: The theory and practice of software process improvement and capability determination*. Los Alamitos, CA, USA: IEEE Computer Society Press.

Feiler, P., & Humphrey, W. (1993). Software process development and enactment: Concepts and definitions. In *International Conference on the Software Process*. Pittsburgh, PA: Software Engineering Institute.

García, J., Amescua, A., Sánchez, M.-I., & Bermón, L. (2011). Design guidelines for software processes knowledge repository development. *Information and Software Technology, 53*, 834–850.

Gonzalez-Perez, C., & Henderson-Sellers, B. (2008). A work product pool approach to methodology specification and enactment. *Journal of Systems and Software, 81*(8), 1288–1305.

Harmsen, F., & Brinkkemper, S. (1995). Design and implementation of a method base management system for a situational CASE environment. In *Proceedings of the Second Asia-Pacific Software Engineering Conference (APSEC'95)* (p. 430). Washington, DC, USA: IEEE Computer Society.

Harmsen, F., Brinkkemper, S., & Oei, J. L. H. (1994). Situational method engineering for informational system project approaches. In *Proceedings of the IFIP WG8.1 Working Conference on Methods and Associated Tools for the Information Systems Life Cycle* (pp. 169–194). New York, NY, USA: Elsevier Science Inc.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105.

Heym, M., & Osterle, H. (1992). A semantic data model for methodology engineering. In *Computer-Aided Software Engineering, 1992. Proceedings., Fifth International Workshop on* (pp. 142–155).

Karlsson, F. (2008). Method Tailoring as Negotiation. In *Proceedings of CAiSE Forum* (pp. 1–4). New York, New York, USA: ACM Press. doi:10.1145/1479190.1479193

Karlsson, F. (2012). Longitudinal use of method rationale in method configuration: an exploratory study. *European Journal of Information Systems, 22*(6), 690–710. doi:10.1057/ejis.2012.30

Kelly, S., Lyytinen, K., & Rossi, M. (1996). MetaEdit+ A fully configurable Multi-User and Multi-tool CASE and CAME Environment. In *Proceedings of the Conference on Advanced Information Systems Engineering* (pp. 1–21).

Kruchten, P. (1995). Architectural Blueprints—The "4+ 1" View Model of Software Architecture. *IEEE Software, 12*(November), 42–50.

Mirbel, I. (2007). Connecting method engineering knowledge: a community based approach. In *Situational Method Engineering: Fundamentals and Experiences* (Vol. 244, pp. 176–192). Springer. Retrieved from http://www.springerlink.com/index/b76874145ullt431.pdf

Mirbel, I., & Ralyté, J. (2006). Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, *11*(1), 58–78.

Mishra, D., Aydin, S., & Mishra, A. (2013). Situational Requirement Method System: Knowledge Management in Business Support. *New Trends in Databases and Information Systems, 185,* 349–359. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-32518-2_33

Nunamaker Jr., J. F., Chen, M., & Purdin, T. D. M. (1990). Systems development in information systems research. *Journal of Management Information Systems - Special issue on management support systems*, *7*(3), 631–640.

ObjectManagementGroup. (2005). Unified modeling language 2.0. *http://www.omg.org/spec/UML/2.0/.*

Ocampo, A., & Munch, J. (2006). Process Evolution Supported by Rationale: An Empirical Investigation of Process Changes. In *Lecture Notes in Computer Science 3966* (pp. 334–341). Berlin/Heidelberg: Springer.

Petersen, K., & Wohlin, C. (2010). Software process improvement through the Lean Measurement (SPI-LEAM) method. *Journal of Systems and Software*, *83*(7), 1275–1287. doi:10.1016/j.jss.2010.02.005

Pino, F. J., García, F., & Piattini, M. (2008). Software process improvement in small and medium software enterprises: a systematic review. *Software Quality Journal*, *16*(2), 237–261.

Prakash, N., & Gupta, D. (1998). An Architecture for a CAME Tool. In H. Kangassalo (Ed.), *Information Modelling and Knowledge Bases X* (pp. 200–219).

Ralyté, J. (2004). Towards situational methods for information systems development: engineering reusable method chunks. In *Proceedings of the International Conference on Information Systems Development*.

Rolland, C. (2009). Method engineering: towards methods as services. *Software Process: Improvement and Practice*, *14*(3), 143–164. doi:10.1002/spip.416

Rossi, M., Ramesh, B., Lyytinen, K., & Tolvanen, J.-P. (2004). Managing Evolutionary Method Engineering by Method Rationale. *Journal of the Association for Information Systems*, *5*(9), 356–391.

Rus, I., & Lindvall, M. (2002). Knowledge management in software engineering. *IEEE Software*, *19*(3), 26–38.

Saeki, M. (2003). CAME: The first step to automated method engineering. In R. Crocker & G. L. Steele Jr. (Eds.), *Proceedings of the Workshop on Process Engineering for Object-Oriented and Component-Based Development* (pp. 7–18). Anaheim, California, USA: ACM.

Saeki, M., & Iguchi, K. (1993). A meta-model for representing software specification & design methods. In *Proceedings of the IFIP WG8.1 Working Conference on Information System Development Process*.

Si-Said, S., Rolland, C., & Grosz, G. (1996). MENTOR: a computer aided requirements engineering environment. In *Advanced Information Systems Engineering* (pp. 22–43).

Souer, J., Joor, D.-J., Helms, R., & Brinkkemper, S. (2011). Identifying commonalities in web content management system engineering. *International Journal of Web Information Systems*, *7*(3), 292–308.

Souer, J., & Mierloo, M. Van. (2008). A Component Based Architecture for Web Content Management: Runtime Deployable WebManager Component Bundles. *2008 Eighth International Conference on Web Engineering*, 366–369. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4577904

Sulayman, M., Urquhart, C., Mendes, E., & Seidel, S. (2012). Software process improvement success factors for small and medium Web companies: A qualitative study. *Information and Software Technology*, *54*(5), 500–479.

Tolvanen, J. P. (1998). *Incremental method engineering with modeling tools: theoretical principles and empirical evidence*. University of Jyväskylä.

Tolvanen, J.-P., & Rossi, M. (2003). MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators. In R. Crocker & G. L. Steele Jr. (Eds.), *Proceedings of the Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA03)*. Anaheim, California, USA: ACM.

Van Berkum, M., Brinkkemper, S., & Meyer, A. (2004). A Combined Runtime Environment and Web-Based Development Environment. In *Proceedings of the Internation Conference on Advanced Information Systems Engineering* (pp. 307–321).

Van de Weerd, I., & Brinkkemper, S. (2008). Meta-modeling for situational analysis and design methods. In *Handbook of Research on Modern Systems Analysis and Design Technologies and Applications* (p. 35). Information Science Publishing.

Van de Weerd, I., Brinkkemper, S., & Versendaal, J. (2007). Concepts for Incremental Method Evolution : Empirical Exploration and Validation in Requirements Management. In *International Conference on Advanced Information Systems Engineering* (pp. 469–484). Springer.

Van de Weerd, I., Brinkkemper, S., & Versendaal, J. (2010). Incremental method evolution in global software product management: A retrospective case study. *Information and Software Technology*, *52*(7), 720–732.

Van de Weerd, I., Versendaal, J., & Brinkkemper, S. (2006). A product software knowledge infrastructure for situational capability maturation: Vision and case studies in product management. In B. Regnell, E. Kamsties, & V. Gervasi (Eds.), *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 1–16). Luxemburg, Luxemburg: Essener Informatik Beiträge.

Van Stijn, P., Vlaanderen, K., Brinkkemper, S., & van de Weerd, I. (2012). Documenting Evolutionary Process Improvements with Method Increment Case Descriptions. In D. Winkler, R. V. O'Connor, & R. Messnarz (Eds.), *European System, Software & Service Process Improvement & Innovation Conference* (pp. 193–204). Vienna, Austria: Springer.

Vlaanderen, K., Brinkkemper, S., & van de Weerd, I. (2012). On the Design of a Knowledge Management System for Incremental Process Improvement for Software Product Management. *International Journal of Information System Modeling and Design (IJISMD)*, *3*(4), 21. Retrieved from http://www.igi-global.com/article/design-knowledge-management-system-incremental/70925

Vlaanderen, K., & Tuijl, G. Van. (2013). Incremental Method Enactment for Computer Aided Software Engineering Tools. In S. Nurcan, H. A. Proper, P. Soffer, J. Krogstie, R. Schmidt, T. A. Halpin, & I. Bider (Eds.), *Workshop for Exploring Modelling Methods for Systems Analysis and Design* (pp. 370–384). Barcelona: Springer.

Vlaanderen, K., van de Weerd, I., & Brinkkemper, S. (2012). On the Design of a Knowledge Management System for Incremental Process Improvement for Software Product Management. *International Journal of Information System Modelling and Design*, *3*(4), 46–66.

Vlaanderen, K., van de Weerd, I., & Brinkkemper, S. (2013). Improving software product management: a knowledge management approach. *International Journal of Business Information Systems*, *12*(1), 3–22.

Wiig, K. M., de Hoog, R., & van der Spek, R. (1997). Supporting knowledge management: A selection of methods and techniques. *Expert Systems with Applications*, *13*(1), 15–27.