The Online Method Engine: From Process Assessment to Method Execution

Kevin Vlaanderen, Inge van de Weerd, and Sjaak Brinkkemper

Utrecht University,
Department of Computer Science,
P.O. Box 80.007,
3508 TA, Utrecht, the Netherlands,
{k.vlaanderen,i.vandeweerd,s.brinkkemper}@cs.uu.nl

Abstract. The field of method engineering has seen an increasing amount of interesting approaches and techniques over the last ten years. The coverage of these techniques ranges from the modeling of processes and systems to the situational construction of new ones. However, access to the required domain knowledge is often not available, and the effort required for effective method engineering is in most cases too much. To overcome these problems, we propose an incremental approach for process assessment, process improvement, and process execution, based on method engineering techniques and tools. The approach is implemented in the Online Method Engine; a holistic solution that supports these three aspects. In this paper, we give a conceptual overview of the approach, along with an overview of the current state of development.

Keywords: Method Engineering; Assessment; Process Improvement; Online Method Engine; Method Fragments; Method-as-a-Service; Software Product Management

1 Introduction

Many researchers [17, 15, 10, 4] describe the use of a method base in situational method engineering. Method fragments can be stored in it, for example by using the MEL method engineering language [6]. Once retrieved from the method base, they can be combined following the assembly rules described by Brinkkemper [5]. More recently, work has been performed on allowing incremental method evolution [23]. According to this work, method fragments can be used to describe and improve the evolution of software product management methods, by allowing the insertion, modification and deletion of method fragment components.

Some method bases have actually been implemented, such as OPF [11] and the CREWS method base [14]. However, for practitioners (the actual method users), retrieving these method fragments and using those in their daily work can be cumbersome. A prerequisite is that the method user should be aware of the exact method fragment that he or she is searching for. In addition, the method user must know what to do with the retrieved method fragment, how to interpret it, and how to implement it in the organization.

To support the method engineering activity, several computer-aided method engineering (CAME or meta-CASE) tools have been developed. Most of these focus on the meta-modeling aspect. One well-known example of such a tool is MetaEdit+ [19], which enables the definition and usage of domain-specific languages. This tool was also applied in a more agile context [3]. On the other hand, several tools that focus more on the method construction aspect have been developed as well. One example in this field is the work of Saeki [18]. Work on the method base management system 'Decamerone' has been performed by Harmsen and Brinkkemper [9].

Unfortunately, the current method bases and knowledge infrastructures are too hard to use for many practitioners. They do not always know exactly what they are looking for, nor how to implement a formal method description in the processes of their organization. Therefore, in this research, we go a few steps further. We propose an Online Method Engine (OME) that can not only be used to store and retrieve method fragments, but also to assess an organization's current processes, create an advice based on this assessment, and implement this advice in the organizations processes and tools.

This research has many similarities with research on the Method as a Service, described by Rolland [16] and Deneckère et al. [7], and by Guzélian and Cauvet [8]. By adopting a Service-Oriented Architecture (SOA) for method engineering, the authors aim to change method fragments into method services which are implemented as Web services [16]. Deneckère describes how the concept of SOA is adopted in a MOA, a Method-Oriented Architecture. This MOA facilitates a method services registry in which available method services are organized. The authors describe the MOA usage in two use cases. They state that method engineers can use CAME tools to define new method with services compositions. On the other hand, method users (developers, practitioners) can use their CASE-tools to invoke remote method services. Unfortunately, the Method as a Service concept is not thoroughly understood yet.

In our vision of the OME, existing method bases are extended. The OME does not only provide a repository in which method fragments are stored, but also offers the opportunity for users to assess their own processes and investigate which ones should be improved. Based on this assessment, an improvement roadmap is created that is used as a basis for a number of method increments. Furthermore, the company's tooling infrastructure can be directly aligned with the method improvement by automatically configuring templates and work-documents.

In the remainder of this paper, we first explain the principle of model-driven process assessment and improvement. Then, section 3 describes the implementation of this principle in the OME. Finally, in section 4, we present our conclusions and further research.

2 Incremental Process Assessment and Improvement

The idea of incremental process improvement that we present in this paper consists of several separate steps. The starting point for each process improvement

is an analysis of the current process, based on which a maturity profile can be calculated. The situational factors of the company are used to determine an optimal maturity profile. By calculating the delta between these two, the required process improvement is determined. This process improvement is further detailed by relating it to suitable method fragments that can be combined into a new process that improves the company's process. This brief summary of the process is illustrated in Figure 1. Each of the steps in the model is explained in more detail in the following sections.

At several points in the text, references will be made to example implementations in the domain of Software Product Management (SPM). In contrast to most method engineering approaches, our solutions are not implemented in the software engineering domain. SPM deals with management of requirements, the definition of releases, and the definition of software products in a context where many internal and external stakeholders are involved [20]. It represents a context where the creation and application of situational methods is very relevant, but where knowledge regarding effective method implementations is scarce.

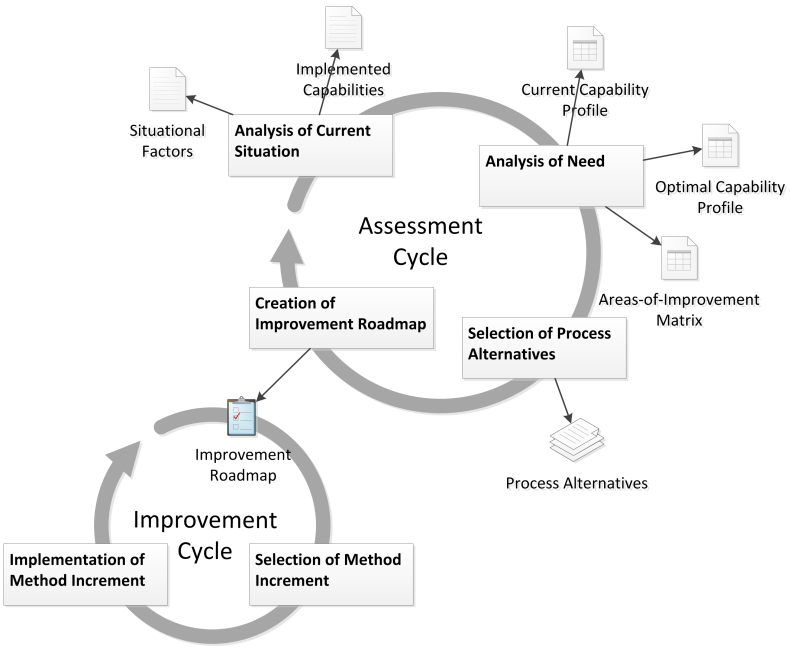


Fig. 1. Incremental Process Improvement

2.1 Analysis of Current Situation

The first step in the process improvement activity is obtaining an overview of the current situation in terms of implemented capabilities, and situational factors of

the business (unit). This approach can be generalized into a form as depicted by Figure 2. The figure is an instance of a PDD (the same applies to Figure 3). Its notation is based on a combination of a UML activity diagram and a UML class diagram. On the left-hand side, boxes indicate the activities that are to be performed. Complex activities that are not further specified here have a black shadow. On the right-hand side, the resulting deliverables are shown, along with their relationships.

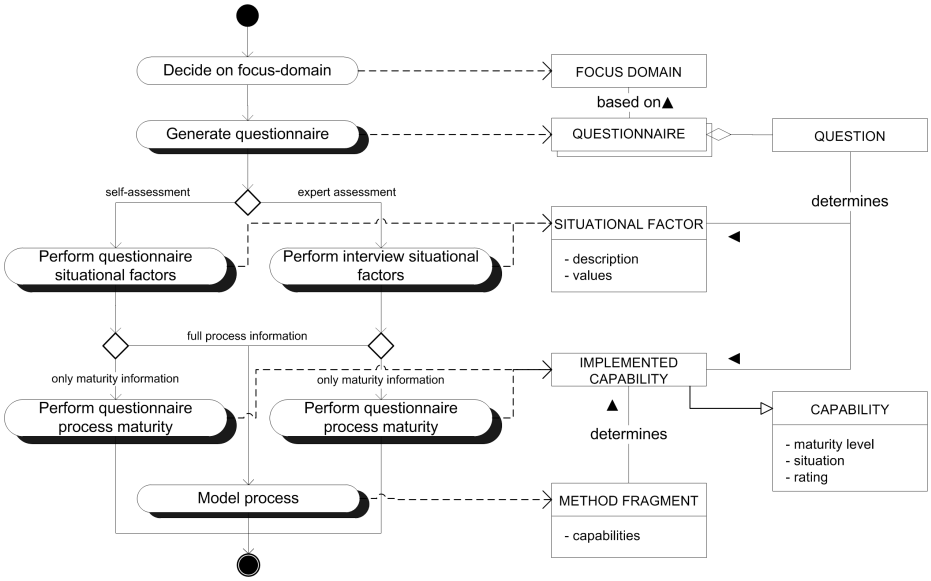


Fig. 2. Analysis of current situation

The current situation constitutes both the currently employed process as well as more generic aspects of the company at hand. During the initial phase, it is the company that needs to decide what the extent of the analysis will be, i.e. the focus domain. We can identify two types of situations regarding the motivation for employing method engineering:

- The need for improvement of a specific area. In many cases, method improvements can better be performed in an evolutionary way rather than in a revolutionary way. By doing so, you reduce risk and increase the chance of success. This also means that it is often not required to analyze the entire process. Instead, only a specific part of the process is looked at, and only for that part improvements are provided.
- The need for improvement of the entire process. For companies that do require a major improvement of their process, this should be a possibility. In those cases, the entire process should be analyzed. This group also contains (new) companies that wish to obtain advice without having a process in

place yet, or with a process that is to be abandoned altogether. Although the latter will only very rarely happen, it should be taken into account.

Based on this choice, a questionnaire is generated and performed to gather information regarding the situational context. In the area of SPM, the situational analysis has been performed by conducting a questionnaire with a list of all the relevant situational factors as described by Bekkers et al. [1]. To enhance reliability of the data, the questionnaire could be replaced by performing an interview. Similar solutions can be developed for other areas.

Data from interviews that have been held suggest that there is a variety of wishes regarding the amount of effort that companies are willing to put in, in order to obtain process improvement advice. We can distinguish two manners in which companies are willing to provide information regarding their current process:

- Full process information. In the optimal case, companies are willing to provide complete information regarding their current process, deliverables, and situational factors that describe their environment. This means that their entire process needs to be captured in a way suitable for further elaboration. Also, the situational factors need to be captured in some way, either through a questionnaire or by means of an interview. With all data available, the process improvement advice that can be obtained is the most effective. However, capturing the entire process requires significant work from an expert who is able to employ an appropriate modeling technique.
- Only situational factors and maturity information. In many cases, capturing full process information requires too much effort. Therefore, it should be possible to provide a process improvement advice based solely on the situational factors and maturity information. This option implies that the advice does not contain any information on how to implement the advice, but only what should be implemented.

If a company is willing to provide full information regarding (part of) their process, the process should be modeled by an expert, either internal or external. The resulting model should contain detailed information regarding both the process as well as the deliverables. Therefore, process-deliverable diagrams (PDDs) are a very suitable technique for this purpose. Vlaanderen et al. [22] show how PDD's can be used to model an SPM process and to capture the current maturity level of a company's product management process.

2.2 Analysis of Need

The next phase takes the situational factors and the list of implemented capabilities from the first phase as input, after which it determines how the current process could be improved. In the domain of Software Product Management, this phase has already been described by Bekkers et al. [1] in the form of the situational assessment method, but it will be summarized here for the sake of

completeness (see Figure 3). The need analysis consists of three activities; (1) construction of the current capability profile, (2) calculation of the optimal capability profile, and (3) calculation of an 'areas of improvement' matrix. The first of these three consists of translating the results from the initial maturity assessment into a form usable for further calculation.

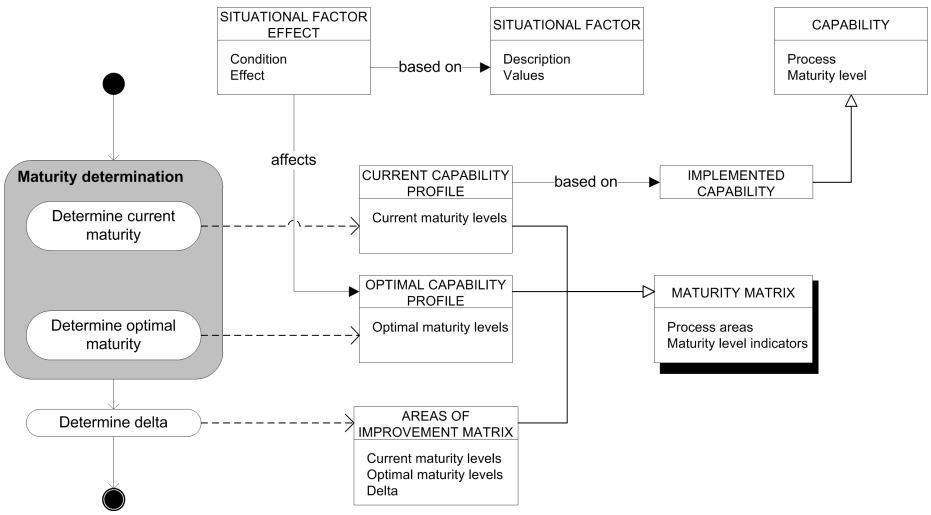
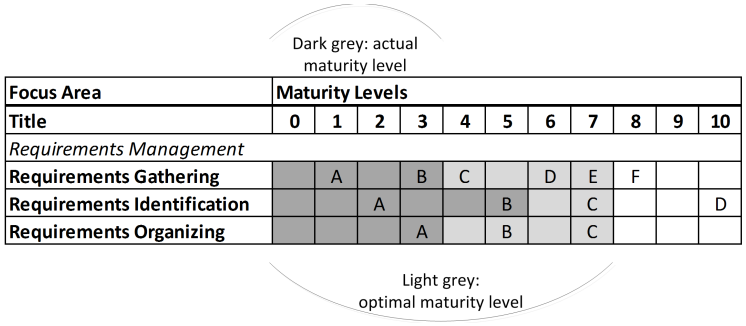


Fig. 3. Analysis of Need

The second activity is somewhat more complex. The optimal capability profile is determined by a set of situational factor effects. Several situational indicators have an associated effect. By applying all applicable situational factors effects, an optimal capability profile is obtained that is customized for the current company.

The current capability profile and the optimal capability profile are then combined into an Areas of Improvement matrix. This is again a capability matrix, with both previous matrices integrated into it. Between the two matrices, a gap can exist, which can be called the delta. This delta indicates the capabilities that need to be implemented, in order to arrive at the optimal maturity level. An example of such an Areas of Improvement matrix within the Software Product Management domain is shown in Figure 4. The actual delta is the light-grey areas, as this depicts the difference between the actual and the optimal maturity level. This set forms the basis for the next phase in the process of method improvement.

What is important in the context of this phase is the fact that users can vary in the rigidity that they demand from the method engineering process. Some wish only a partial improvement for a specific area, while others wish to improve their process to the maximum maturity level suggested for them. As stated before, evolutionary improvement is in many cases more prone to success



Focus Area	Maturity Levels										
Title	**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			

Fig. 4. Example Areas of Improvement Matrix

than revolutionary change. This implies that it should be possible to provide improvements in the form of a roadmap when the process is changed rigorously.

2.3 Selection of Process Alternatives

For the next phase, each missing capability has to be connected to a method fragment that implements the capability. The capabilities that a method fragment implements can be used as an attribute during the initial selection of method fragment candidates. As will be described later on, both process fragments as well as deliverable fragments can implement capabilities. For this reason, capability is an effective first classifier of a method fragment in the method base.

For further classification, we reuse the situational factors described by Bekkers et al. [2]. Since many fragments will be applicable in any situation, it does not make sense to describe each fragment by all factors. This would also pose a problem when the list of situational factors would change. Therefore, situational factors should only be used to indicate restrictions on the use of the fragment. The combined set of indicators for a specific fragment forms its second classifier, situation.

A third classifier of method fragments is their rating. Through the feedback of users, method fragments are rated on several aspects, such as effectiveness, complexity, etc. Method fragments with a very low rating can be ignored in most cases, while in other cases method fragments with a high rating are selected over similar method fragments with a low rating. A simple example of a method fragment with its describing attributes can be found in table 1. It is based on a prioritization technique used within the SPM domain.

Although processes, capabilities and situational factors form a very solid ground for method fragment selection, we need to take into account that we are dealing with processes in which humans are involved. This means that the resulting process needs to fit with the preferences of the people involved in it. These people need to be able to express these preferences during the selection of alternative method fragments. The results from interviews have indicated that product managers are not always willing to accept suggestions made to them by a

Wiegers' Prioritization Matrix		
Capabilities	**Situation**	**Rating**
– Internal Stakeholder Involvement – Prioritization Methodology – Customer Involvement – Cost Revenue Consideration – Partner Involvement	– # of requirements < 50 – Partner involvement >= medium	– Ease of use: 8/10 – Satisfaction: 6.5/10

Table 1. Example Method Fragment with Attributes

machine [21]. Therefore, the process should allow for differences in the amount of freedom that is provided. While it is generally a good idea to suggest one specific method fragment per capability, users should be at liberty to select another. This 'freedom-of-choice' has serious consequences for the OME. In order to make the freedom given to users useful, they need to be provided with a sufficient amount of information for them to base their decision on.

The first source of information for this is the method fragment itself. Since every method fragment can be displayed in the form of a PDD, users can use this diagram to form an initial mental image of its implications. This is possible since all related activities and deliverables are readily available in the method fragment. However, in addition to this, we also identified a need for more sources in the form of experience reports. Experience from people in similar situations is highly valued, and would thus be a valuable addition to the process.

Based on all of the sources of information combined, users should be able to make a valid and well-argued choice regarding the method fragments that should be selected, and thus regarding the changes that should be made to the existing process.

2.4 Creation of Improvement Roadmap

After the improvements have been selected, the process of embedding or implementing the process advice varies depending on the amount of information that a company has provided. The possibilities are limited when only maturity information is known, in contrast with the field of opportunities when full process information is given. In any case, the initial part of the process can be the same for both situations, as this regards the elaboration of the chosen solution into steps. Steps are needed since solutions will in many cases be too large for implementation in one iteration. An evolutionary approach has more chance of success as it will likely yield a higher acceptance due to smaller, incremental changes.

The splitting of solutions into steps is subject to several conditions. Solutions cannot be split into steps randomly. The major reason for this is that we need to take dependencies into consideration. If a company wants to increase the maturity level of its requirements gathering process from A to C (see Figure 4,

it does not make sense to implement automation before centralized registration. Instead, the first step should be to implement the activity related to level B, followed by an iteration in which level C is implemented.

In most cases, several capabilities can be implemented at the same time. However, to make iterations or steps more successful, it is probably wise to make sure that each step has some sort of goal, or a theme. This ensures a set of changes that is coherent. This way, the change-process seems less chaotic to the employee. This is important, as he or she will be the one performing the new process.

After the roadmap has been presented to the user and has been accepted, the implementation of it can start. In case that only maturity information is available, this process is fairly straightforward, as little support can be given. The changes that have been proposed need to be implemented in the company manually. In order to guide this, process descriptions and templates related to the advice are provided.

If full process information is available, then this process is considerably more complex. This part encompasses the most complex asset of method engineering, namely the assembly of method fragments. For each step, the selected method fragments need to be integrated with the existing process. As this is a difficult task, it is probably best to do this fragment by fragment.

A problem with this segmented approach, however, is the risk that some parts of the process get changed multiple times. This is unwanted, as this can lead to confusion among the people that need to perform the process. Therefore, already during the creation of the roadmap, the system should make sure that no such situations occur. This is also another argument for the statement that method fragments should be kept as small as possible. By preventing the usage of complex method fragments, the chance of overlap is made smaller, thereby increasing the chance of success of any algorithm that is charged with creating a coherent roadmap.

2.5 Selection and Implementation of Method Increments

After the assembly of the selected method fragments into the original process, the changes can actually be implemented within the company. To facilitate the change, the system can generate and/or update templates based on the original and the new process description (expressed in the PDDs).

If the company's original work documents are available, than they can be updated to reflect the new deliverables within the process. During this step, original data should be maintained while new columns, sections, formulas, etc. are added to the documents.

In case deliverables are not available, templates can be generated based on the generated process description. The generated templates should be directly usable within the new process.

In addition, the system generates full process descriptions with explanations of all steps, deliverables and roles. These descriptions aid the process owner during the implementation of the process in the company.

3 Online Method Engine

The method engineering approach described above does not adhere to the Method-as-a-Service philosophy unless it allows users to perform the created method to some extent online. This means that, instead of using various local software tools, the workflow and the deliverables are embedded in an online platform, which we will refer to as the OME. The method modification aspect is essential for improving the effectiveness of processes, but the method execution aspect ultimately allows major improvements in the efficiency of these processes by taking away a large share of the burden of maintaining a complex IT infrastructure.

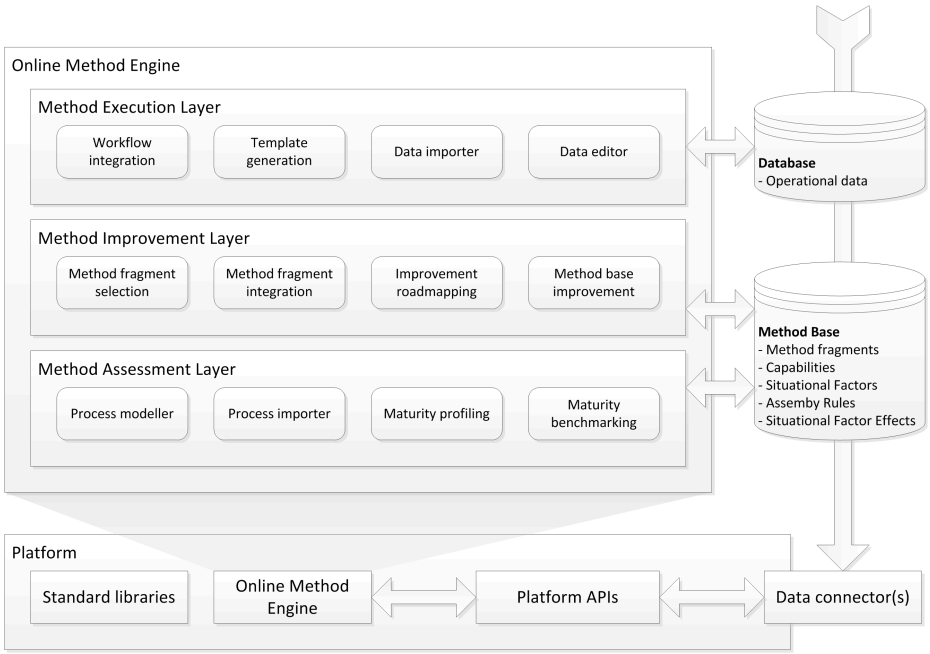


Fig. 5. Online Method Engine

We depicted our vision on the OME in Figure 5. At the bottom, the development platform is depicted. As will be described later, this refers to Google AppEngine in our approach. On top of the platform the OME is shown, with the three functional layers described throughout the previous section. Each layer contains several functional components, shown by the rounded boxes. On the right hand of the figure, two databases are shown; one for the method engineering related data such as method fragments and situational factors, and one for the method execution related data such as requirements and planned releases. These database are connected to the system using separate data-connectors, which al-

low connection to any suitable database, ranging from a MySQL database to the database of a third-party requirements engineering tool.

Such an approach implies an integration of the functionality to describe a process, assess the process, adapt the process, and then perform the process. In order to be able to do so, method fragments need to be correctly translated into an interface that offers the right set of tools for users to perform their tasks. The creation and usage of templates is a core aspect of this. Such online documents can be placed on any cloud documents solution such as Google Docs. As this environment is accessible through an API, it can be integrated into any other system, such as the OME.

In addition to the translation of deliverables into documents and the management of these documents, the activities need to be correctly translated. Aspects that need to be taken into consideration here are the correct translation of access rights based on roles, the distinction between automated tasks and user input, the type of interface that is required for a certain set of activities, and the order of the activities, i.e. sequential, simultaneous, or a mix of both.

Currently, these aspects are not all derivable from the PDDs. To solve this, either stricter rules should be applied during the creation of PDDs, or additional models should be created for defining interface, access rights and business process. The former is not a good solution, as this would make the creation of PDD's too complex. The latter is similar compared to model-driven development solutions such as OO-Method [13] and the web-based variant OOWS [12]. This would require the addition of several steps to the process for creating the required models, undermining an important aspect of the OME, namely the fact that it should be simple.

To forgo this problem, an alternative solution could be developed, based on pattern recognition. The idea behind this is that certain patterns will exist in the PDD's of processes and deliverables that can be directly related to correct solutions for the interface. For instance, activities that are performed simultaneously should be connected to a tabbed interface, with a tab for each activity. Activities that are performed linearly can always be displayed as steps, allowing to go back and forward. Such a solution would require no extra effort of the user. However, the possibilities of recognizing patterns are limited and it is very prone to modeling errors. Therefore, a user should always be able to alter the interface for a given process. Alternative interface elements should be provided for this by the system. The same holds for the generation of documents based on deliverables. As it is not always possible to derive the required file-type for a deliverable, the user should have the option to change this manually.

To capture all the requirements of the translation from method description to interface, a meta-model should be defined describing all possible translations for every construct and pattern.

3.1 Information extraction using MERL

For capturing processes in the OME, referred to as 'process modeler' in Figure 5, we currently use the tool MetaEdit+. MetaEdit+ is "an environment that allows

building modeling tools and generators fitting to application domains" [19]. It lets users define domain-specific meta-models that are used to generate a tool that is suited specifically for creating diagrams based on that meta-model. In this case, the meta-model for PDD has been implemented by defining all constructs (activities, deliverables, etc.) and rules, in addition to the visual aspects of those constructs.

Next to the modeling-capabilities of MetaEdit+, the tool also embeds a transformation language called MERL, or the MetaEdit+ R* Language. This language allows for converting diagrams into any format required. Although the language in itself is not very powerful, some tricks will make any conversion to a textual format such as XML or latex possible.

The language is normally used for code generation, in the context of model-driven development. With such approaches, the solution domain is modeled using a domain-specific language/diagram, after which the diagram is analyzed and converted into source code. In this case, the information stored in the diagrams is used to describe the context / situation of an SPM process, and to assess its maturity (model-driven assessment).

For this research, generators have been written that allow the generation of a filled-in maturity matrix based on all PDDs of a company's process [21]. Combined with the actual diagrams, these pieces of information form a good overview of the maturity of a process, along with its description in terms of activities and deliverables.

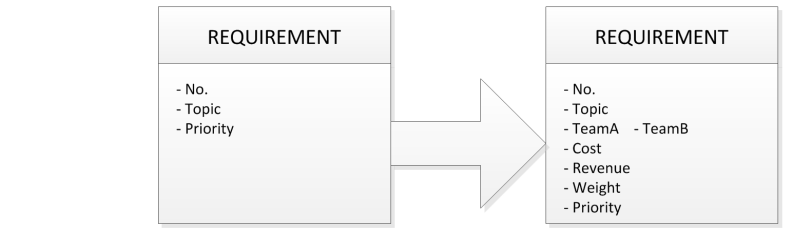
3.2 Template Generation

As described earlier, changes made to a process through the OME should be facilitated and supported as much as possible, to ensure the success of the evolution. One technique for doing so is providing automated templates based on the deliverables of a process. In the case of minor changes, changes made to a template can be incorporated in the original company documents, preserving any data already existing. We developed a proof of concept, in which we show how a Google spreadsheet can be updated dynamically by changing the meta-model of a method fragment.

In Figure 6, the meta-model of a deliverable of a requirements prioritization activity is presented. This deliverable is a requirement. In the old case, this requirement had three attributes: No., Topic, and Priority. This priority was added based on the personal preference of the one who stored the requirements.

In the new situation, another approach is used to prioritize the requirements. The variables 'Cost', 'Weight' and 'Revenue' are added and used to calculate the priority. Furthermore, the attributes 'TeamA' and 'TeamB' are added to divide the costs (in man days) to the teams.

In the two spreadsheets that are illustrated in Figure 6, the change in the meta-model of the requirements can be viewed. Extra columns have been added to the spreadsheet. In this case, some of the cells in the new template are already filled in. Normally, this is a job for the method user.



	A	B	C	D
1	No.	Topic	Priority	
2	BR1-100686	Records of emergency equipment inspection & maintenance	low	
3	BR1-100700	Service records of presence sensing systems	high	
4	BR1-100760	Accidental damage to stores, equipment, & property	medium	
5	BR1-101100	Working environment	low	
6	BR1-101101	Outline of warden training program	later	
7	BR1-100542	Safe mechanical & manual handling procedures for movement of materials	critical	
8	BR1-101064	Records of hazard investigations - reportable - non-reportable	medium	

	A	B	C	D	E	F	G	H
1	No.	Topic	TeamA	TeamB	Cost	Revenue	Weight	Priority
2	BR1-100686	Records of emergency equipment inspection & maintenance	10	15	25	130	1	0.29
3	BR1-100700	Service records of presence sensing systems		70	70	130	4	3.30
4	BR1-100760	Accidental damage to stores, equipment, & property	18		18	760	3	3.72
5	BR1-101100	Working environment		100	100	290	3	7.89
6	BR1-101101	Outline of warden training program	100		100	240	1	2.18
7	BR1-100542	Safe mechanical & manual handling procedures for movement of materials	30		30	730	5	9.93
8	BR1-101064	Records of hazard investigations - reportable - non-reportable	15		15	800	1	1.09

Fig. 6. Example of an incremental template

To make templates useful in practice, the current information as provided by the method fragment does not suffice. Relations within and between concepts are not clearly defined. This means that, for example, the formula that specifies the value of 'Cost' based on the other attributes (in the example above) cannot be modeled. At this point, we do not yet have a satisfactory solution for this.

4 Conclusions & Further Research

In this paper, we presented our vision on the OME, an online environment that can be used to assess an organization's current processes, create an advice based on this assessment, and align the company's tooling infrastructure with the method improvement by automatically configuring templates and documents.

Although the concept presented in this paper is fairly detailed, the OME that we envision is not yet operational. The complexity of such a system was already known, and this paper only strengthens the idea that we are dealing with an advanced concept requiring a lot of research effort. From this point onwards, each of the areas of the OME needs to be addressed in detail, putting together the puzzle piece by piece. Expertise in several areas will be needed, as each part of the OME has its specific challenges, from linguistic analysis for method assembly to data-optimization for the method base.

Up until this stage, the research effort has mainly been focused on analyzing the current situation and the need, with a focus on the SPM domain. Furthermore, a lot of research effort has been spent on the underlying meta-modeling techniques that are used throughout the system. This leaves the remaining areas of process alternative selection, improvement roadmap creation, and increment selection and implementation open for future research.

An important factor that can never be left out during the elaboration is the fact that the purpose of the OME is the improvement of processes. As a consequence, we are always dealing with people that bring habits, experiences, and opinions. This should not be overlooked. Doing so would result in a system that is too rigid, forcing people into ways of working that they will not accept, thereby foregoing the purpose of the system. However, if it is done right, than the OME has great potential value. We believe that this solution can increase the maturity of the software industry significantly by providing professionals with the right tools to optimize their processes.

Unfortunately, the detailed OME that is presented in this paper has not been fully validated yet. As no concrete system exists yet, doing so would have involved asking potential users to imagine themselves using such a system. This is a tremendous effort, especially due to the complexity of it, and would likely not have resulted in a valid response. However, as development continues, the user should not be forgotten. Instead, at several points in time, his opinion should be asked and corrections should be made according to it. When done correctly, this will result in a functional online method engineering environment.

References

1. Bekkers, W., Spruit, M., van de Weerd, I., van Vliet, R., Mahieu, A.: A Situational Assessment Method for Software Product Management. In: Proceedings of ECIS2010 (accepted) (2010)
2. Bekkers, W., van de Weerd, I., Brinkkemper, S., Mahieu, A.: The Influence of Situational Factors in Software Product Management: An Empirical Study. In: IWSPM '08: Proceedings of the 2008 Second International Workshop on Software Product Management. pp. 41–48. IEEE Computer Society, Washington, DC, USA (2008)
3. Berki, E.: Formal Metamodelling and Agile Method Engineering in MetaCASE and CAME Tool Environments. In: Proceedings of the 1st South-East European Workshop on Formal Methods, SEEFM'03. pp. 170–188. No. November (2003)
4. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. Information and Software Technology 38(4), 275–280 (1996)
5. Brinkkemper, S., Saeki, M., Harmsen, F.: Meta-modelling based assembly techniques for situational method engineering. Information Systems 24(3), 209–228 (1999)
6. Brinkkemper, S., Saeki, M., Harmsen, F.: A Method Engineering Language for the Description of Systems Development Methods. In: CAiSE '01: Proceedings of the 13th International Conference on Advanced Information Systems Engineering. pp. 473–476. Springer-Verlag, London, UK (2001)

7. Deneckère, R., Iacovelli, A., Kornyshova, E., Souveyet, C.: From Method Fragments to Method Services. Proceedings of EMMSAD'08 (2008)
8. Guzélian, G., Cauvet, C.: SO2M: Towards a Service-Oriented Approach for Method Engineering. In: Proceedings of the international conference IKE'07 (2007)
9. Harmsen, F., Brinkkemper, S.: Design and implementation of a method base management system for a situational CASE environment. In: Second Asia-Pacific Software Engineering Conference (APSEC'95). p. 430. Published by the IEEE Computer Society (1995)
10. Harmsen, F.: Situational Method Engineering. Ph.D. thesis, Universiteit Twente (1997)
11. Henderson-Sellers, B.: Process metamodelling and process construction: examples using the OPEN Process Framework (OPF). Annals of Software Engineering 14(1), 341–362 (2002)
12. Pastor, O., Fons, J., Pelechano, V.: OOWS: A method to develop web applications from web-oriented conceptual models. In: Proceedings of IWWOST'03. Luis Olsina, Oscar Pastor, Gustavo Rossi, Daniel Schwabe, Oviedo (2003)
13. Pastor, O., Insfrán, E., Merseguer, J., Romero, J., Pelechano, V.: OO-METHOD: An OO Software Production Environment Combining Conventional and Formal Methods. In: Proceedings of CAISE'97. vol. 1250, pp. 145–159. Springer-Verlag, Barcelona (1997)
14. Ralyté, J.: Reusing scenario based approaches in requirement engineering methods: CREWS method base. In: International Workshop on Database and Expert Systems Applications (DEXA). pp. 305–309. No. Dexa 1999, Ieee (1999)
15. Ralyté, J., Jeusfeld, M., Backlund, P., Kuhn, H., Arni-Bloch, N.: A knowledge-based approach to manage information systems interoperability. Information Systems 33(7-8), 754–784 (Nov 2008)
16. Rolland, C.: Method engineering: towards methods as services. Softw. Process 14(3), 143–164 (2009)
17. Saeki, M.: Object-oriented meta modelling. Object-Oriented and Entity-Relationship Modeling 1021, 250–259 (1995)
18. Saeki, M.: Came: The first step to automated method engineering. In: Workshop on Process Engineering for Object-Oriented (2003)
19. Tolvanen, J.P., Rossi, M.: MetaEdit+: Defining and Using Domain-Specific Modeling Languages and Code Generators. In: Proceedings of the Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA03) (2003)
20. van De Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. In: International Workshop on Software Product Management (IWSPM). pp. 3–12. IEEE (Sep 2006)
21. Vlaanderen, K.: Improving Software Product Management Processes : a detailed view of the Product Software Knowledge Infrastructure. Ph.D. thesis, Utrecht University (2010)
22. Vlaanderen, K., van De Weerd, I., Brinkkemper, S.: Model-Driven Assessment in Software Product Management. In: International Workshop on Software Product Management (IWSPM) (2010)
23. van de Weerd, I., Souer, J., Versendaal, J., Brinkkemper, S.: Concepts for Incremental Method Evolution : Empirical Exploration and Validation in Requirements Management. In: Advanced Information Systems Engineering. pp. 469–484. Springer (2007)