Incremental Method Engineering for Process Improvement - A Case Study

Dominique Mirandolle, Inge van de Weerd and Sjaak Brinkkemper

Utrecht University, Department of Computer Science,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
d.e.mirandolle@students.uu.nl
{i.vandeweerd,s.brinkkemper}@cs.uu.nl

Abstract. In order for companies to improve the maturity level of their development process, they need to design new methods or adapt the existing ones. This research aims to deliver a proof of concept of how incremental method engineering supports the maturation of methods in a product software company. We show how the adaptation of a method can lead to a higher maturity level. We assessed the method of a case company by means of the situational assessment method, resulting into the maturity level and situational factors. We also modeled eight different prioritization methods according to their maturity level and situational factors to find out which of these could be implemented into the case company's method in order to evolve to a higher maturity level. After matching the situational factors of the available methods with the company factors we find one method that is suitable to implement into the existing method at the case company. We explain how the implementation can take place and how this would evolve the method to full maturity.

Keywords: Incremental method engineering, software product management, competence model, situational factors, maturity matrix

1 Introduction

Product software companies have to be on track with the latest changes in the field of software development and product management. Naturally, their processes and methods need to be adjusted accordingly to the changes in the environment and growth of the company. Yet, many product software companies find it difficult to improve the maturity level of their methods [1]. In order for companies to improve this maturity level they need to design new methods or adapt the existing ones, while there is little education available in the software product management (SPM) area [2].

In this research, we use an incremental method engineering approach to improve an organization's process maturity. Method engineering (ME) is the discipline to design, construct and adapt methods, techniques and tools for the development of information systems [3]. If a method is tuned to the project at hand, this is called situational ME. When only a method fragment, and not the entire method, is changed, this is called incremental ME. A method increment can be defined as “a method

adaptation, in order to improve the overall performance of a method" [1]. Incremental ME can be seen as a sub type of situational ME, where incremental ME focuses more on evolving a method in time towards a higher maturity level by changing small parts of the method.

1.1 Aim of this Research

The aim of this study is to deliver a proof of concept of how incremental ME supports the maturing, and thus the improvement of processes in a product software company. The main research question in this study is formulated accordingly:

"How can incremental method engineering support process improvement in the software industry?"

By answering this question we aim to contribute to the field of incremental ME by showing how the adaptation of a method can lead to a higher maturity level of that method. We elaborate on how method increments, based on Situational Factors (situational factors) of both the method and the company, can evolve the method of our case company. This verifies the theoretical description of incremental ME.

1.2 Related Work

Several approaches have been introduced to make it easier for companies to change their development methods [4, 5, 6]. To help companies select a proper approach to adapt an existing method, Ralyté et al. [7] present a generic model for situational method engineering. In their approach, the method engineer is able to combine the approaches that fit the ME project the best by setting intentions (goals) and connect these with strategies. Ågerfalk et al. [8] also present a method to help method engineers with the configuration and adaptation of methods. They propose the use of pre-made reusable configurations of a base method suitable for a specific characteristic of a development situation. Rossi et al. [9] claim that method users, but especially method engineers need to be aware of the rationale of the method in order to coordinate the development and evolution of an existing method base.

Van de Weerd et al. use the concept of *incremental method engineering* as a principle in their Product Software Knowledge Infrastructure (PSKI) [1]. Incremental method engineering is a specific type of situational method engineering, where development methods are over time incrementally adapted to the changing conditions. The principle is used in the PSKI, which enables organizations to acquire a custom-made advice to improve their processes incrementally. An important part of the PSKI is the method base, which is loaded with existing method fragments. Accordingly to the Situational Factors (situational factors) of the company, method fragments are chosen out of the method base in order to create a more mature method. The domain for which the PSKI is initially proposed is Software Product Management. By developing the Software Product Management Competence Model [10] (Fig. 1), they give an overview and structure to the software product management domain. The model divides the internal functions of software product management into four business functions: portfolio management, product planning, release planning and requirements management, which contain a total of 15 focus areas, such as

'requirements prioritization' and 'product roadmapping'. Furthermore, a maturity matrix for SPM was developed, in which for each focus area three to five capabilities were defined. The maturity matrix is depicted in Table 1. If all are implemented, full maturity is reached. The methods we analyzed in this research are all requirements prioritization methods, in the business function release planning.

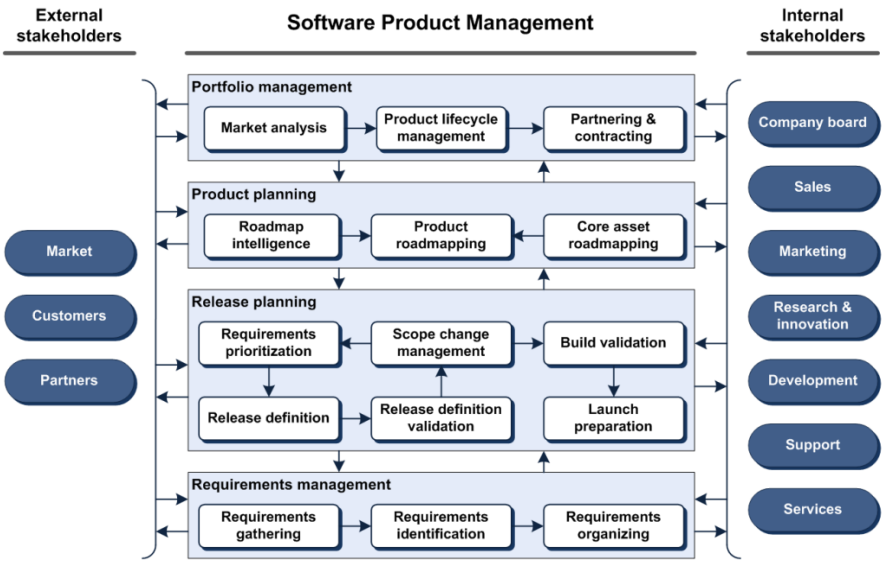

Fig. 1. SPM Competence Model

2 Research Approach

In order to deliver the proof concept and answer the main research question, we perform a case study [11] at a product software company called Teezir, hereafter called 'the case company'. We focus on a small part of the release planning stage in the SPM Competence Model: requirements prioritization. We have chosen this particular process as it has not yet reached the highest maturity level within the case company, according to the assessment. Additionally, we are familiar with several methods applicable in this stage and literature about this stage. In this stage, the requirements for an information system are sorted according to importance for certain stakeholders. The literature we use for this research mainly consists on literature about requirements prioritization methods and method engineering.

Our research approach consists of two main steps:

1. *Select and analyze methods.* In order to deliver a proof of concept we analyze eight requirements prioritization methods. The analysis is performed by measuring their maturity according to the Competence Model, developed by Bekkers & van de Weerd [10]. Additionally, we describe the situational factors per method, based on work by Bekkers et al. [12].

2. *Case study*. We analyze the prioritization method that is used by the case company, as well as its maturity levels and situational factors. Based on this information, we map the method fragments found in the previous step to the case company and select the best one. Finally, we propose how to implement this method fragment.

3 Selection and Analysis of Methods

The methods that are selected for this research are from the SPM domain, in particular focusing on requirements prioritization. Different ways exist to prioritize requirements. Some existing methods are very complex and involve many stakeholders, while others are simple. In this section, first the requirements prioritization focus area is further analyzed. Then, an overview of the selected prioritization method is presented. Finally, the situational factors of each method are listed.

3.1 Requirements Prioritization

Table 1 presents the SPM maturity matrix, consisting of the 15 focus areas, each with its own number of specific maturity levels. The focus area specific maturity levels are represented by the letters A-F in Table 1 and range from maturity level 1 to 10 (the topmost row in Table 1). In this research we focus on requirements prioritization. This focus area contains five capabilities (denoted with letters A-E) [10].

The five requirements prioritization capabilities and their goals are:

A. *Internal stakeholder involvement*
   Goal: Improved product quality & increased involvement of internal stakeholders in the product management process.
   Action: All relevant internal stakeholders indicate the requirements that should be incorporated in future releases by assigning priorities to the requirements.

B. *Prioritization method*
   Goal: Structure the requirement prioritization process and therewith provide a solid prioritization which is balanced and clear to all parties involved.
   Action: A structured technique is used.

C. *Customer involvement*
   Goal: Incorporation of customer needs and wishes in the product.
   Action: Customers and prospects indicate the requirements that should be incorporated in future releases by assigning priorities to the requirements from their point of view. Customers can also be represented by delegates.

D. *Cost revenue consideration*
   Goal: Create a financial basis for the prioritization.
   Action: Information about costs and revenues of each (group of) requirement(s) is taken into account during the requirements prioritization (costs can be expressed in other means than money).

E. *Partner involvement*
   Goal: Improved product quality & increased involvement of external stakeholders in the product management process.

Action: Partner companies indicate requirements that should be incorporated in future releases by assigning priorities to the requirements.

Table 1. SPM Maturity Matrix

	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											
Roadmap 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 lifecycle management					A	B			C	D	E

3.2 Requirements Prioritization Methods

The eight requirements prioritization methods were selected based on the availability of literature of each method. The methods are shown in Table 2, along with the capabilities that are implemented in them. When combining the information available in Table 1 and Table 2, we can see that for example the Binary Priority List method has a maturity level of 4. We can conclude this, since Table 2 shows that capability A and B are implemented in this method. According to Table 1, an implementation up to capability B has a maturity level score of 4.

Table 2. Requirements Prioritization Methods

Method	**Implemented capabilities**				
Binary Priority List [14]	A	B			
WinWin requirements negotiation model [15]	A	B	C		
Integer linear programming approach [16]	A	B	C	D	E
Requirements Triage [17]	A	B	C		E
MOSCOW [18]	A	B	C	D	
Cost Value Approach [19]	A	B	C	D	
Quality Function Deployment [20]	A	B	C		
Features Prioritization Matrix [21]	A	B	C	D	E

When methods miss a capability, the maturity level only scores up to the capability before the first missing capability. For the Requirements Triage method [14] this means that we measure the maturity level only up to capability C. This results in a maturity level of 5.

We modeled all the selected methods in Process Deliverable Diagrams (PDD). A PDD is a diagram that integrates an activity diagram on the left-hand side and a deliverable view on the right-hand side [13]. To illustrate our research method, Fig. 2 shows the activities and deliverables of the Requirements Triage method [17]. For all eight requirements prioritization methods such a PDD and a corresponding concept table and activity table were created.

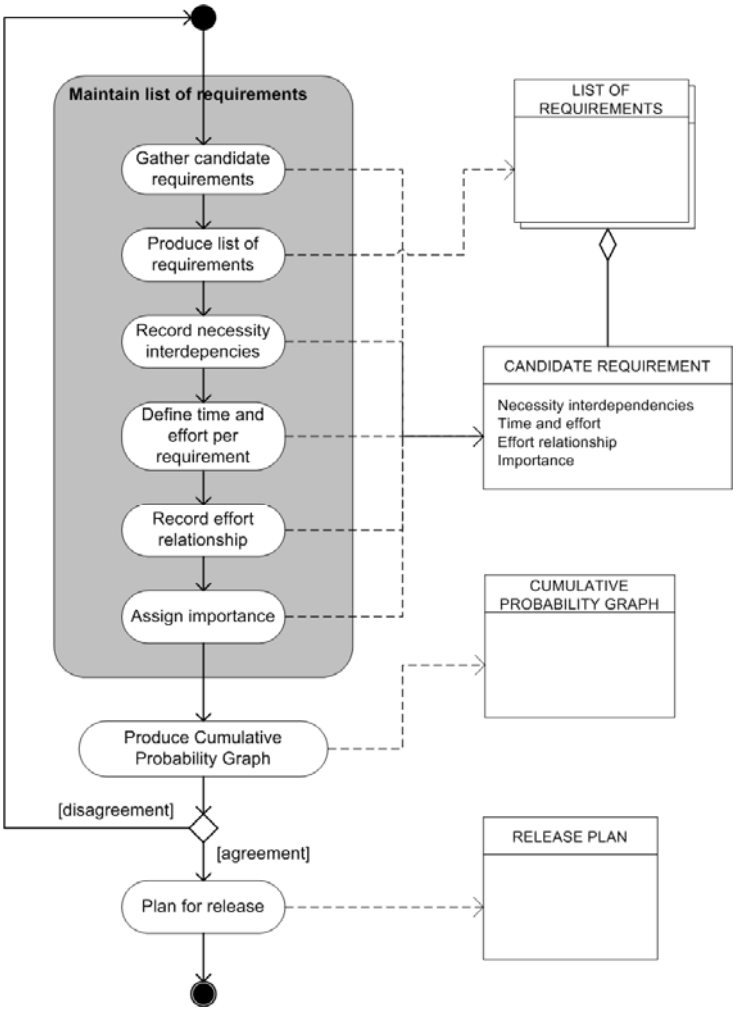

Fig. 2. PDD of the Requirements Triage method

3.3 Situational Factors

Bekkers [12] presents, in a case study among 14 software product companies, a list of 31 situational factors which need to be kept in mind when configuring or choosing a development method. For each of the selected methods we have marked whether the value of the situational factor is of any importance and if so, what value would suit the method best. As an example, we show the results of this analysis for Requirements Triage in Table 3. Out of the 31 situational factors, 11 are of importance in this method. This is because Requirements Triage focuses on cooperation between business and development departments. The situational factors of which the value does not affect that functionality of the method, which can contain any value, are left out in this table.

Table 3. Situational Factors of the Requirements Triage method

Situational factor	Value
Size of business unit team	Large
Size of development team	Small to Medium
Number of customers	High
Number of end-users	High
Release frequency	High
Variability of feature requests	Large
Product size	Large
Company policy	High
Customer involvement	High
Legislation	Strict
Partner involvement	High

4 Case Study

4.1 Case Study Design

The in-depth investigation in this case study takes place a product software company called Teezir, a ‘search solutions’ company (hereafter called ‘the case company’. Their main product is a web based dashboard that integrates various widgets containing representations of the online reputation of a specific brand name (for example term clouds, sentiment analysis, volume of mentions etc.). The dashboard is a standard product which can be customized by the client himself. By dragging and dropping the preferred widgets on the dashboard, a suitable application for the situation or customer at hand can be generated.

By carrying out interviews at the case company, we create a complete overview of the used prioritization method, situational factors and maturity level. This overview enables us to select the best fitting candidate method that, once implemented, will bring the case company to a higher maturity level.

Fig. 3 illustrates how the process of fitting methods works. The process of comparing the situational factors of the candidate method to the case company's method can be seen as trying to fit a key on a keyhole. In a way we have created a keyhole by defining the situational factors of the case company. The different values of the situational factors are the holes in the keyhole's cylinder. All the candidate methods are keys, of which the situational factors are pins that need to match into the holes of the keyhole's cylinder. The situational factors of the candidate methods need to be as equal as possible to those of the case company (however, not all situational factors are relevant in this case). All we need to do is find the key that matches the keyhole.

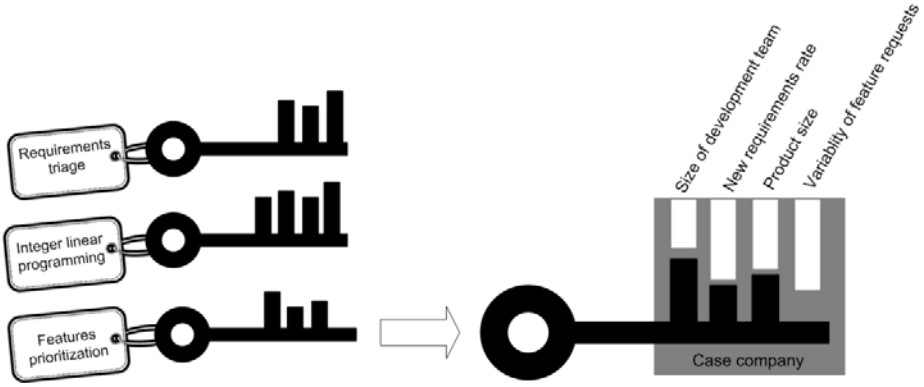

Fig 3. Fitting the candidate methods to the case company's method

4.1.1 Conduction of Case Study

In a semi-structured interview with the case company's software engineer, we analyzed the requirements prioritization method that is used at this company. We described their method and visualized it in a Process Deliverable Diagram (PDD): a diagram that integrates an activity diagram on the left-hand side and a deliverable view on the right-hand side [13]. With this information we were able to define the maturity level of the case company nowadays. Additionally, we elaborated on the case company's situational factors. Once we knew at which maturity level the case company was operating and which situational factors influence the company, we could suggest them to adopt (one of the) method fragments we analyzed in order for them to grow and develop their method towards a higher maturity level. Finally, the method fragment which suited the case company best is implemented in the original method. We elaborated on how this can take place and visualized the matured method in a PDD.

4.1.2 Analysis of Case Study Evidence

The case company uses the Dynamic Systems Development Method (DSDM) [22]. The requirements prioritization method that is used in DSDM (and by the case company) is the MOSCOW method, as depicted in Fig. 4.

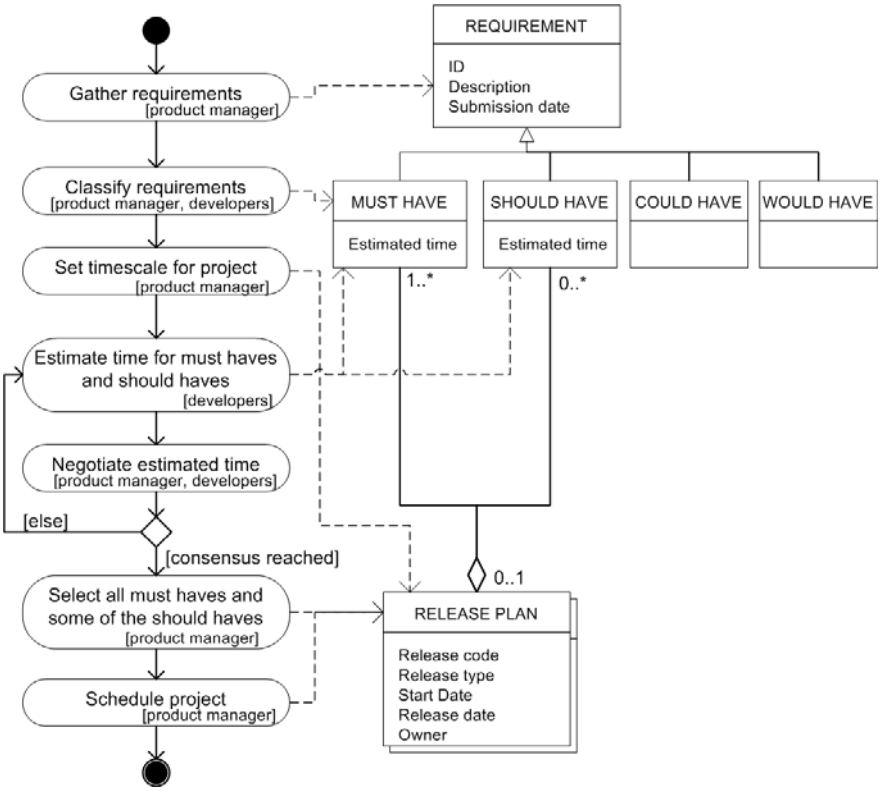

Fig. 4. PDD of the MOSCOW method

The MOSCOW requirements prioritization method contains maturity levels A to D, since it contains internal stakeholder involvement, a requirements prioritization method, customer involvement and a cost revenue consideration (measured in time).
In addition, we have analyzed the situational factors for the case company, as is presented in Table 4.

The requirement prioritization method used at the case company nowadays is MOSCOW. This method contains the maturity levels A to D. If the case company's method would evolve, activities that contain level E should be added to the method. Level E contains partner involvement, and its goal is to improve product quality and to increase involvement of external stakeholders in the product management process.

Table 4. Situational factors of the case company

Situational factor	Value
Development philosophy	Iterative
Size of business unit team	6 FTE
Size of development team	4 FTE
Customer loyalty	High
Customer satisfaction	6 (out of 10)
Customer variability	40% of customers have customized features
Number of customers	25
Number of end-users	150
Type of customers	All sorts of companies
Hosting demands	Central hosting services
Localization demand	Low
Market growth	Growing
Market size	3500+ potential customers
Release frequency	Every 250 days
Sector	Marketing
Standard dominance	Medium request for market standards
Variability of feature requests	Low
Application age	2 years
Defects per year: total	0 per year
Defects per year: serious	0 per year
Development platform maturity	Fully developed
New requirement rate	3 requests per year
Number of products	1
Product lifetime	3 year
Product size	350 KLOC
Product tolerance	High (not sensitive to bugs)
Software Platform	.NET
Company policy	High level of influence
Customer involvement	Medium involvement
Legislation	Loose
Partner involvement	High level of influence

Of the eight methods we analyzed in this research, three contain activities that implement maturity level E. These are Requirements Triage [17], Integer linear programming approach [16], and Features Prioritization Matrix [21]. Based on the situational factors of these three methods and those of the case company, we can now define which of these would suit the case company's method best (which key fits in the keyhole). Table 5 shows the values of the situational factors (the pins of the keys) of the three mature methods. We already defined the situational factors of the case company (the keyhole) in Table 4. The bottom rows of Table 5 show how many pins of the candidate methods' keys fit into the keyhole, and thus how many situational factors match to the situational factors of the case company.

The situational factors of which the value does not affect the functionality of the method and can contain any value, are left blank in this table. Additionally, we printed the situational factors that do not match the case company in italic. The cells that contain plain text do match the situational factors of the case company. At the

bottom of the table we sum up how many matches and mismatches each method contains.

Table 5. Situational Factors of the three A-E methods

Situational factor	Requirements Triage	Integer linear programming	Features Prioritization
Development philosophy			
Size of business unit team	*Large*		
Size of development team	Small to medium		Small to medium
Customer loyalty			
Customer satisfaction			
Customer variability			
Number of customers	*High*		
Number of end-users	*High*		
Type of customers			
Hosting demands			
Localization demand			
Market growth			
Market size			
Release frequency	*High*		
Sector			
Standard dominance			
Variability of feature requests	*Large*	*Large*	
Application age			
Defects per year: total			
Defects per year: serious			
Development platform maturity		*High*	
New requirement rate	*High*	*High*	Low to medium
Number of products		*High*	
Product lifetime			
Product size	*Large*	*Large*	Small to medium
Product tolerance			
Software Platform			
Company policy	High		
Customer involvement	*High*	*High*	*High*
Legislation	*Strict*		
Partner involvement	High	High	High
Matches	22	25	30
Mismatches	9	6	1

In Table 5 it can be seen that Wiegers' Features Prioritization Matrix is not dependent on a lot of factors. The method is known to be applicable on almost every kind of project. Requirements Triage is on the other hand suitable for projects with eleven specific situational factors. Requirements Triage focuses on projects in which large amounts of requirements are involved. The method is designed specifically to

deal with a ‘chaos’ of requirements, since it originates from the medical domain, where patients need to be ‘sorted’ or ‘triaged’ as quickly as possible. Additionally, it tries to involve as many stakeholders as possible (e.g. customers, developers, financial and legal representatives, etc.), which explains why company policy, customer involvement, legislation and partner involvement all have a high influence on the method. This suggests that the method deals with large projects, in which a large number of end-users are involved.

On the other hand, the Integer Linear Programming approach and Requirements Triage have six and nine mismatching situational factors respectively. They are both suitable for large projects, with a large amount of products involved. Therefore, it seems obvious to choose the Features Prioritization Matrix method to expand the case company’s current method to maturity level E.

The case company could evolve its requirements prioritization method by applying the multiple stakeholder sheet in the MOSCOW method. This would mean that all stakeholders, including partners, would be involved in the requirements prioritization method. If the multiple stakeholder sheet would be used, all requirements first get a value based on the opinion of all stakeholders. The requirements that have the highest calculated value will be implemented. Fig. 5 illustrates how the additional activities would be added to the PDD of the MOSCOW method and how the deliverables change. Changes are marked in grey.

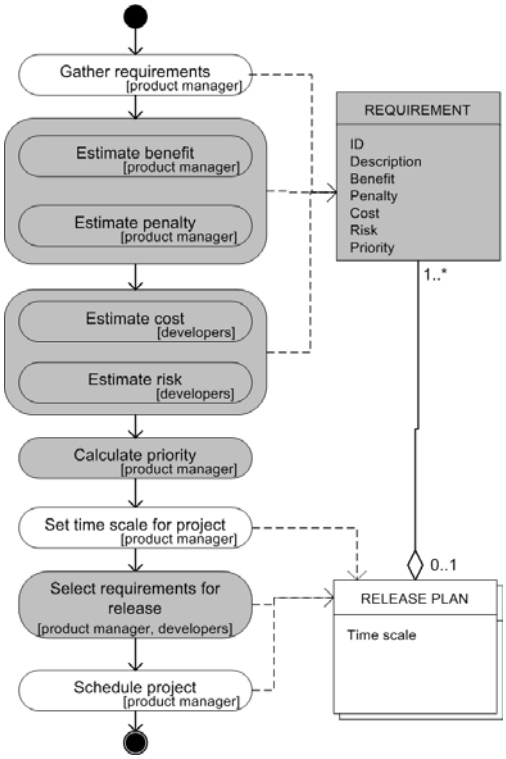

Fig. 5. Method increment with prioritization using Wiegers’ matrix

As can be seen in Fig. 5, in Wiegers' Features Prioritization Matrix, the value of a requirement is calculated by estimating the benefits, penalties, costs and risks per requirement on a scale from 0-9 [21]. All requirements and corresponding values are stored in a spreadsheet. Wiegers developed the 'multiple stakeholder sheet', which is very useful in case there are multiple stakeholders that have different visions when it comes to the variables that result in the value of a requirement. In this case, those multiple stakeholders are the product manager and the developers. After assigning the values, the priorities are calculated and used as a basis for selecting the requirements for the next release.

4.2 Discussion

Using the multiple stakeholder sheet and thus the Features Prioritization Matrix seems be a useful way of integrating the opinion of all stakeholders, including partners, into the requirements prioritization method. The main advantage of adapting the method this way is that all involved stakeholders get an opportunity to influence the requirements prioritization. Additionally, the classification of requirements is turned into a calculated result out of estimating variables instead of an estimation of the importance of the overall requirement. This results most likely into a more accurate and realistic requirement prioritization.

The case study carried out is a first evaluation of the idea of incremental method engineering, through marching situational factors. Although often described in literature, not many practical examples have been presented. Therefore, we believe that although this is just a single case study, it is an important contribution to the method engineering field. However, in order to strengthen our argument, we should carry our more case studies [11]. Also, the method base in this research contained eight requirements prioritization methods. Further research can be done with a larger method base, in order to fine tune a method more specifically to the situational factors of a case company.

Furthermore, instead of using the current situational factors of the case company, it might also be interesting to use situational factors that the company predicts or aims to reach in the near future. For example, if a company wishes to expand its number employees this could be registered in the list of situational factors while matching them to a suitable method. By doing this the company might be less likely to outgrow its method in a short time.

A last important issue for further research is the evaluation of the method fragment implementation at the case company. Currently, we link this implementation to an increase in maturity. However, more interesting is whether the increment also leads to an increase in performance. Indicators that could be used for this are duration of the decision process, customer satisfaction or time-to-market.

5 Conclusion

In this research we have analyzed the requirement prioritization method of a case company according to maturity level and situational factors. We have also analyzed

eight requirement prioritization methods on maturity level and situational factors to find out which of these could be implemented into the case company's method in order to let this method evolve to a higher maturity level. We used a comparison with keys (candidate methods) and a keyhole (case company's method) to visualize how a suitable method can be chosen out of all candidate methods. By doing this we have answered our main research question and illustrated how incremental ME can support the maturing of an information systems development method in a product software company.

We have found that the case company implemented the MOSCOW requirement prioritization method, which contains maturity levels A-D. In order to mature the method, level E would need to be added. Three out of the eight methods we analyzed contained maturity level E. By comparing the situational factors of these three methods with the situational factors of the case company, we have found that one method (Wiegers' Features Prioritization Matrix) is suitable to add to the existing method in order to let it mature.

For further research, we plan to carry out more case studies and extend the method base with more method fragments. Furthermore, we aim to verify whether the proposed method increments actually improve performance.

References

1. Weerd, I. van de, Versendaal, J., Brinkkemper, S.: A Product Software Knowledge Infrastructure for Situational Capability Maturation: Vision and Case Studies in Product Management. In: Proceedings of the 12th Working Conference on Requirements Engineering: Foundation for Software Quality, 97-112 (2006)
2. van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L.: Towards a Reference Framework for Software Product Management. In: 14th International Requirements Engineering Conference, Minneapolis/St. Paul, MN, USA, 319-322 (2006)
3. Brinkkemper, S.: Method Engineering: Engineering of Information Systems Development Methods and Tools. In: Information and Software Technology 38(4), 275-280 (1996)
4. Kumar, K. & Welke, R.J.: Methodology Engineering: A Proposal for Situation-Specific Methodology Construction. In: Challenges and Strategies for Research in Systems Development, pp. 257-269. John Wiley & Sons, Inc. New York, NY (1992)
5. Tolvanen, J.-P.: Incremental Method Engineering with Modeling Tools: Theoretical Principles and Empirical Evidence. Jyväskylä Studies in Computer Science, Economics and Statistics 47, University of Jyväskylä, PhD Dissertation thesis (1998)
6. Aydin, M.N., Harmsen, F.: Making a Method Work for a Project Situation in the Context of CMM. In: 14th International Conference on Product Focused Software Process Improvement, Rovaniemi, Finland, 158-171 (2002)
7. Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: Proceedings of the 15th International Conference CAISE'03, Klagenfurt/Velden, Austria, LNCS 2681, Springer, pp. 95-110 (2003)
8. Ågerfalk, P.J., Wistrand, K., Karlsson, F., Börjesson, G., Elmberg, M., Möller,K.: Flexible Processes and Method Configuration: Outline of a Joint Industry-Academia Research Project. In: 5th International Conference on Enterprise Information Systems (2003)
9. Rossi, M., Ramesh, B., Lyytinen, K., Tolvanen, J.: Managing Evolutionary Method Engineering by Method Rationale, In: Journal of the Association for Information Systems: Vol. 5: Iss. 9, Article 12 (2004)

10. Bekkers, W., Weerd, I. van de, Spruit, M., and Brinkkemper, S.: A Framework for Process Improvement in Software Product Management. In: A. Riel et al. (Eds.): EuroSPI 2010, CCIS 99, 1-12 (2010)
11. Yin, R.K.: Case Study Research: Design and Methods. Fourth Edition. SAGE Publications, California (2009)
12. Bekkers, W. , van de Weerd, I. , Brinkkemper, S. , Mahieu, A.: The Influence of Situational Factors in Software Product Management: An Empirical Study, In: 2nd International Workshop on Software Product Management, pp.41-48 (2008)
13. van de Weerd, I., Brinkkemper, S.: Meta-modeling for Situational Analysis and Design Methods. In: M.R. Syed and S.N. Syed (Eds.), Handbook of Research on Modern Systems Analysis and Design Technologies and Applications (pp. 38-58). Hershey: Idea Group Publishing (2008)
14. Bebensee, Th., van de Weerd, I., Brinkkemper, S.: Binary Priority List for Prioritizing Software Requirements. In: 6th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2010), LNCS 6182 (2010)
15. Boehm, B.: A Spiral Model of Software Development and Enhancement. In: Computer, May 1988, 61-72 (1988)
16. van den Akker, M., Brinkkemper, S., Diepen, G., Versendaal, J.: Software Product Release Planning Through Optimization and What-if Analysis. Information and Software Technology, 50(1-2), 101-111 (2008)
17. Davis, A. M.: The Art of Requirements Triage. In: Computer 36(3), 42-49 (2003)
18. Stapleton, J.: DSDM Business Focused Development. Addison-Wesley Professional, Reading (2002)
19. Karlsson, J., Ryan, K.: A Cost-Value Approach for Prioritizing Requirements. In: IEEE Software 14(5), 67-74 (1997)
20. Mizuno, S., Akao, Y. (Eds.): Quality Function Deployment: Integrating Customer Requirements into Product Design. Portland: Productivity Press Inc. (1990)
21. Wiegers, Karl E.: First Things First: Prioritizing Requirements. In: Software Development Magazine, Sept. 1999, 24-30 (1999)
22. Stapleton, J., Dynamic Systems Development Method, In: Proceedings of the Technology of Object-Oriented Languages and Systems, p. 406, June 07-10 (1999)