Situational Fit in Incremental Method Engineering

Inge van de Weerd, Department of Information, Logistics and Innovation, VU University Amsterdam, Amsterdam, The Netherlands

Dominique Mirandolle, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands

Sjaak Brinkkemper, Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands

ABSTRACT

Almost all software vendors use methods and techniques in their software development and production processes. In order to improve the maturity of their processes, an incremental method engineering approach can be followed to adapt and improve methods and techniques. In order to ensure the suitability of selected methods, the authors propose the concept of situational fit to balance environmental characteristics, company characteristics, and information system development methods. They carried out a case study to illustrate the process of incremental method engineering. Furthermore, the authors performed a quantitative analysis on a data set of 38 companies to evaluate the use of situational fit.

Keywords: *Incremental Method Engineering, Production Processes, Situational Factors, Situational Fit, Software Development, Software Product Management, Vendors*

INTRODUCTION

Many software companies struggle with improving their information system development methods (cf. Conradi, Fernström, & Fuggetta, 1993; Niazi, Wilson, & Zowghi, 2005; Coleman & O'Connor, 2008). Pino, García, and Piattini (2008, p. 253) found in a systematic literature review that "the software engineering community has shown an ever-increasing interest in tackling software process improvement in SMEs". However, they also found that small companies find it hard to use the standard SEI and ISO models such as CMM and SPICE as they need to be adapted in order to be used successfully.

The domain in which we carry out our research is the domain of software product management (SPM). SPM is the discipline that governs a software product over its whole life cycle, from its inception to customer delivery, in order to generate the biggest possible value to the business (Ebert, 2007). Also in the case of SPM, companies find it hard to bring their processes to a higher level (van de Weerd, Versendaal, & Brinkkemper, 2006). To improve their processes, they need to design new methods or adapt the existing ones, while there

DOI: 10.4018/jismd.2012100102

is little education available in SPM domain (van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal, & Bijlsma, 2006). Several authors proposed method engineering approaches to create or adapt methods to the company or situation at hand (cf. Brinkkemper, 1996; Aydin & Harmsen, 2002; Ralyté, Deneckère, & Rolland, 2003; Ågerfalk et al., 2003). Van de Weerd, Brinkkemper, and Versendaal (2007) introduced the term *incremental method engineering* to describe the implementation of method adaptations over time, in order to improve the overall performance of a method. A method increment is defined "a collection of method fragments that have been introduced in the method during the method adaptations between t^i and t^{i-1}" (van de Weerd, Brinkkemper, & Versendaal, 2007, p. 474). An important factor here is whether the methods or method fragments fit the situation of the company or project for which it is selected. We call this term *situational fit*. Situational fit is not a new term. In literature on organizational design, situational fit is defined as "the balance of the environmental conditions, the strategy, the management style, the climate, the size, and the technology" (Burton, Lauridsen, & Obel, 2002, p. 1463). Translating this to the method engineering domain, we define situational fit as the balance of the environmental characteristics, company characteristics, and information system development methods.

The aim of this study is to deliver a proof of concept of how situational fit and incremental method engineering can be used as a selection mechanism for process improvement in software companies. The main research question in this study is formulated accordingly:

"How can we use situational fit and incremental method engineering to support process improvement in the software industry?"

By answering this question we aim to contribute to the field of method engineering 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 fit of the method and the company, can improve the processes in the software companies.

In the following section, we first give an overview of related work. Then, we describe our research approach. In the subsequent sections, we present the results of the different steps followed in this approach: method selection and analysis; an exploratory case study in which we explore how incremental method engineering can be used to support process improvement; and a quantitative analysis to evaluate the concept of situational fit. Finally, we present our conclusions and directions for future research.

Related Work

Clark (2000, p. 65) stresses the importance of a clear assessment of process maturity effects, since "software project managers have no way of determining how much improvement is due to process maturity versus other factors" and describes many factors that influence the level of maturity. Also other authors present situational factors, such as Harter et al. (2000) and Schackmann and Lichter (2006). These lists of situational factors are mostly informal and not exhaustive. Therefore, Bekkers et al. (2008) carried out a more systematic research, in which they presented a list of 27 situational factors relevant to SPM, with the level of influence they have on method fragments.

Nejmeh and Riddle (2005) underline the importance of situational factors in process change. They state that an organization's context is of crucial importance in process change cycles. The 'one size fits all' approach cannot be used, since each organization operates in its own context. Van de Weerd, Brinkkemper, and Versendaal (2010) underline this in their retrospective case study in which they analyzed method evolution in an organization over a period of 12 years. They found that changing economic situations required their case company to adapt its methods frequently.

We see that several approaches have been introduced to make it easier for companies to change their development methods (Kumar & Welke, 1992; Tolvanen, 1998; Aydin & Harm-

sen, 2002). To help companies select a proper approach to adapt an existing method, Ralyté, Deneckère, and Rolland (2003) present a generic model for situational method engineering. In their approach, the method engineer is able to combine the approaches that fit the method engineering project the best by setting intentions (goals) and connect these with strategies. Ågerfalk et al. (2003) 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, Ramesh, Lyytinen, and Tolvanen (2004) 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.

Vlaanderen, van de Weerd, and Brinkkemper (2011) use the concept of incremental method engineering as a principle in their Online Method Engine. Incremental method engineering is a specific type of situational method engineering, where development methods are over time incrementally adapted to the changing conditions. This principle is used in the Online Method Engine, which enables organizations to acquire a custom-made advice to improve their processes incrementally. An important part of the Online Method Engine is the method base, which is loaded with existing method fragments. Accordingly to the situational factors and maturity level of a company, method fragments are chosen out of the method base in order to create a more mature method.

The domain for which the Online Method Engine is initially proposed is Software Product Management. By developing the Software Product Management Competence Model, Bekkers, van de Weerd, Spruit, and Brinkkemper (2010) provide an overview and structure to the software product management domain. The model, presented in Figure 1, 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 six capabilities were defined. The maturity matrix is depicted in Table 1. If all are implemented, full maturity is reached.

Research Approach

This research follows a mixed methods approach, in which we combine both quantitative and qualitative research techniques (Johnson & Onwuegbuzie, 2004). The reason we follow this approach is that in this way we can combine qualitative research strengths such as to create an in-depth, rich account (Yin, 2009) and quantitative research strengths such as being able to investigate a larger amount of cases, which increases the generalizability of the research findings. Our research consists of three main steps: 1) method selection and analysis, 2) a qualitative analysis, and 3) a quantitative analysis. In the remainder of this section, we elaborate on each step.

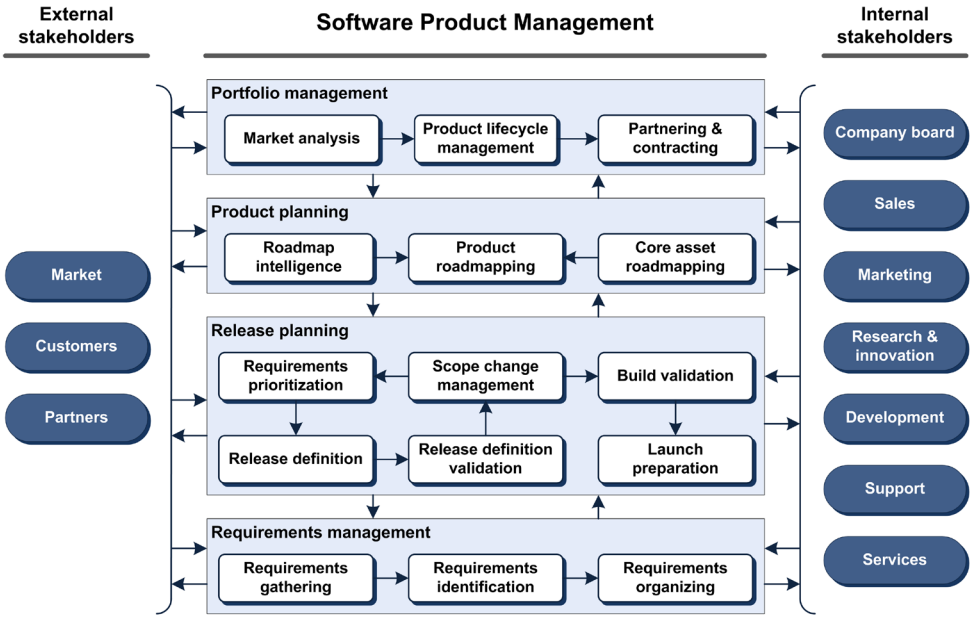
1. Method Selection and Analysis

For both the qualitative part and quantitative part, we need methods that we can analyze and map to situational factors and maturity levels. During our analyses, we focus on a small part of the release planning stage in the SPM Competence Model, namely requirements prioritization. We have chosen this particular process since we are familiar with the literature and existing methods in this focus area.

We analyze eight requirements prioritization methods. The analysis is performed by measuring their maturity according to the Competence Model, developed in earlier research (Bekkers, van de Weerd, Spruit, & Brinkkemper, 2010). Additionally, we describe the situational factors per method, based on work by Bekkers, van de Weerd, Brinkkemper, and Mahieu (2008).

2. Qualitative Analysis

Figure 1. Software product management competence model



The qualitative part of our research consists of an exploratory case study in which we show how incremental method engineering can support process improvement. We analyze the prioritization method that is used by the case company, as well as its maturity level 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.

We chose the case study approach because we are inquiring about a contemporary set of events, over which we have no control (Yin, 2009). The objective of this case study is to show how situational fit can be achieved by linking method fragments to maturity levels and situational factors. The unit of analysis is a software vendor called Teezir, a 'search solutions' company. The type of case study we set up has a single-case design, in which we believe the company is a typical case of which the lessons learned are assumed to be informative about the experience of average companies (Yin, 2009).

3. Quantitative Analysis

The quantitative part of our research covers an analysis on a database of existing case studies gathered in earlier research. In order to confirm the findings from the exploratory case study, we carry out a quantitative analysis on the data of 38 software product management assessments, which are gathered in previous research. With this analysis we show that situational factors are an adequate instrument to select methods from a database and map these to companies based on their situational profile and maturity level.

METHOD SELECTION AND ANALYSIS

The methods that are selected for this research are extracted from the SPM domain, in par-

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

ticular from the focus area of requirements prioritization. Different ways exist to prioritize requirements. Some existing methods are very complex and involve many stakeholders, while others are simple. First, the requirements prioritization focus area and its capabilities are introduced. Then, an overview of the selected prioritization methods is presented. Finally, the situational factors of each method are listed.

Requirements Prioritization Capabilities

Table 1 presents the SPM maturity matrix, consisting of the 15 focus areas, each with its own number of capabilities. The focus area specific maturity levels are represented by the letters A-F in Table 1 and are spread over maturity level 1 to 10 (the topmost row in Table 1). The capabilities are distributed across the maturity levels to indicate a best practice order, in which capabilities in the maturity matrix are implemented left to right. For example, before the A capability of 'Requirements identification' is implemented, first the A capability of 'Requirements gathering should be in place. The levels indicate how mature a company is; the higher the level, the more mature the company is.

In this research we focus on requirements prioritization. This focus area contains five capabilities (denoted with letters A-E) (Bekkers, van de Weerd, Spruit, & Brinkkemper, 2010).

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.

Table 2. Requirements prioritization methods

Method	Implemented Capabilities				
Binary Priority List (Bebensee, van de Weerd, & Brinkkemper, 2010)	A	B			
Quality Function Deployment (Mizuno & Akao, 1990)	A	B	C		
WinWin requirements negotiation model (Boehm, 1988)	A	B	C		
MOSCOW (Stapleton, 2002)	A	B	C	D	
Cost Value Approach (Karlsson & Ryan, 1997)	A	B	C	D	
Requirements Triage (Davis, 2003)	A	B	C		E
Integer linear programming approach (van den Akker, Brinkkemper, Diepen, & Versendaal, 2008)	A	B	C	D	E
Features Prioritization Matrix (Wiegers, 1999)	A	B	C	D	E

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.

Each capability can be implemented in the organization by using existing requirements prioritization methods. Some of these methods will cover all of the capabilities, whereas others only cover a few.

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 by 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.

When methods miss a capability, the maturity level only scores up to the capability before the first missing capability. For the

Requirements Triage method (Davis, 2003) 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 (PDDs). A PDD is a diagram that integrates an activity diagram on the left-hand side and a deliverable view on the right-hand side (van de Weerd & Brinkkemper, 2008). An example of such a PDD can be seen in Figure 3. For all eight requirements prioritization methods such a PDD and a corresponding concept table and activity table were created.

Situational Factors

Bekkers, van de Weerd, Brinkkemper, and Mahieu (2008) present 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. "A situational factor [3] is any factor relevant for product development and product services" (van de Weerd, Versendaal, & Brinkkemper, 2006, p. 101). Examples of situational factors are company size, the development philosophy that is followed, and the number of requirements that are submitted by customers each month. 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.

QUALITATIVE ANALYSIS

For our qualitative analysis, we carried out an exploratory case study at a software vendor 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 user himself. By dragging and dropping the preferred widgets on the dashboard, a suitable application for the situation or customer at hand can be generated.

Data Collection Procedure

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.

In the interviews at the case company, we analyzed the requirements prioritization method that was used. We described their method and visualized it in a PDD. With this information we were able to define the maturity level of the case company. Additionally, we elaborated on the case company's situational factors.

Analysis Procedure

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. Then, the method fragment which suited the case company best was implemented in the original method.

Figure 2 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.

Case Study Results

The case company uses the Dynamic Systems Development Method (DSDM) (Stapleton, 1999). The requirements prioritization method that is used in DSDM (and by the case company) is the MOSCOW method, as depicted in Figure 3.

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 3.

The requirement prioritization method used at the case company nowadays is MOSCOW. 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.

Of the eight methods we analyzed in this research, three contain activities that implement maturity level E. These are Requirements Triage (Davis, 2003), Integer Linear Programming (van den Akker, Brinkkemper, Diepen, & Versendaal, 2008), and the Features Prioritization Matrix (Wiegers, 1999). 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). In addition to the situational factor values of the case company (the keyhole), Table 3 also shows the values of the situational factors (the pins of the keys) of the three mature methods. The bottom rows of Table 3 shows 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 Table 3. 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 Table 3 we sum up how many matches and mismatches each method contains.

Table 3 shows that Wiegers' Features Prioritization Matrix is not dependent on a lot of factors and it has only one mismatching factor with the case company. On the other hand, the Integer Linear Programming approach and Requirements Triage have both four mismatching situational factors. They are both suitable for large projects, with a large amount of incoming requirements. 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, 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. Concluding, 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 of Wiegers' Features Prioritization Matrix. 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. Figure 4 illustrates how the additional activities could be added to the PDD of the case company method and how the deliverables change. Changes are marked in grey.

As can be seen in Figure 4, in Wiegers' Features Prioritization Matrix, the value of a requirement is calculated by estimating the

Figure 2. Fitting the candidate methods to the case company's method

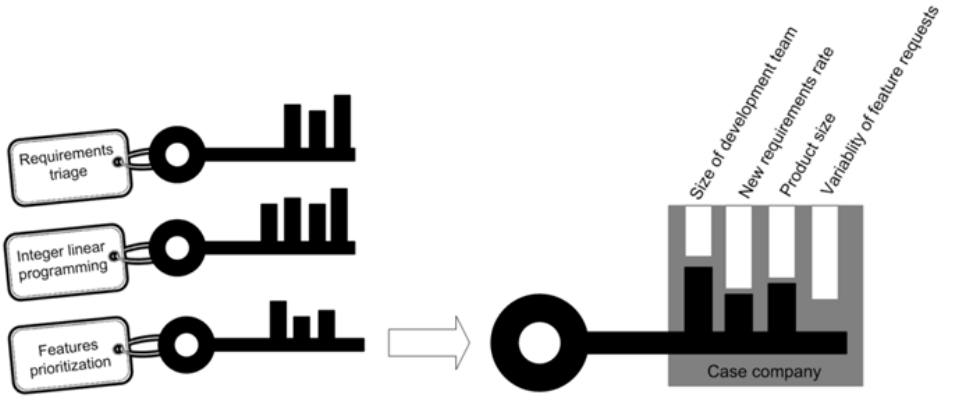


Figure 3. Process-deliverable diagram of the MOSCOW method

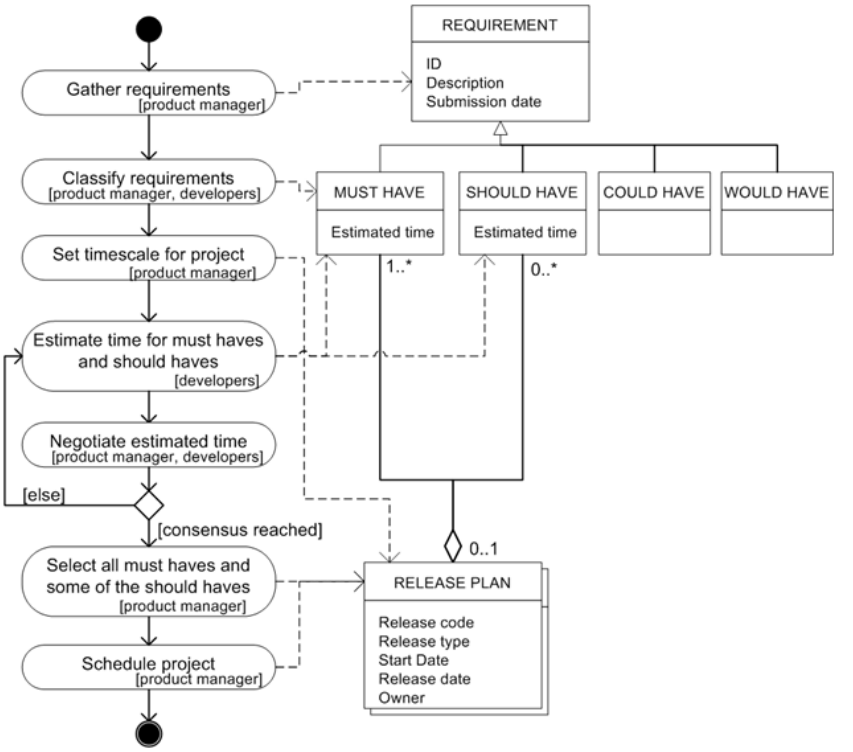
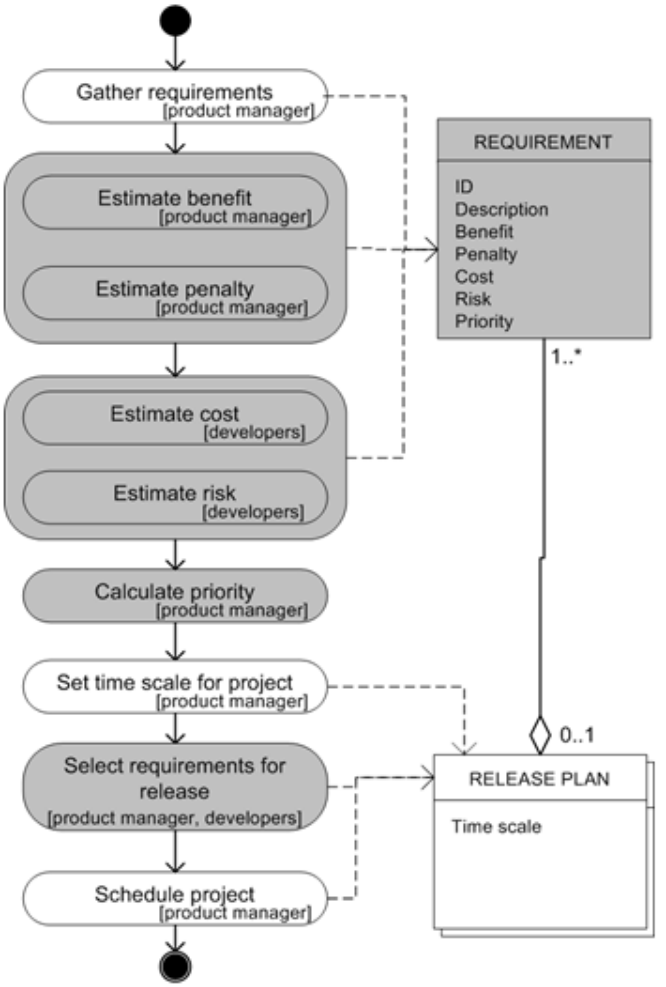


Table 3. Situational factors case company

Situational Factor	Case Company	RT	ILP	FPM
Development philosophy	Iterative			
Size of business unit team	6 FTE	Medium		
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	*High*		
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		Medium	Medium
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		*High*	
New requirement rate	3 requests per year	*High*	*High*	Low to medium
Number of products	1		*High*	
Product lifetime	3 year			
Product size	350 KLOC	*Large*	*Large*	Small to medium
Product tolerance	High (not sensitive to bugs)			
Software Platform	.NET			
Company policy	High level of influence	High		
Customer involvement	Medium involvement			
Legislation	Loose	*Strict*		
Partner involvement	High level of influence	High	High	*Low*
	Matches	**26**	**26**	**29**
	Mismatches	**4**	**4**	**1**

Figure 4. Method increment with prioritization using Wiegers' matrix



benefits, penalties, costs and risks per requirement on a scale from 0-9 (Wiegers, 1999). All requirements and corresponding values are stored in a spreadsheet. Wiegers developed the multiple stakeholder sheet to be used 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.

Validity Threats

In an exploratory case study, the quality of the methodology should be judged on the basis of three types of validity (Yin, 2009). Firstly, the construct validity concerns the validity of the used research method. We ensured this by using different sources of data (interviewees and

documents) to guarantee that problems related to subjectivity and bias of data were avoided.

Secondly, the external validity is about the domain to which the results can be generalized. The research domain of this case study is software product management in software vendors. Although we cannot claim that our findings in the case study can be generalized to all software vendors, we do believe that the case itself is a typical case and that the findings can be generalized to theory (Lee & Baskerville, 2003), in this case the theory of incremental method engineering and situational fit.

Finally, the reliability of the case study is about demonstrating that the results of the study can be replicated. This threat is mitigated by clearly describing each step in the case study in such a way it can be replicated, by maintaining a case study database with the information used in the case study.

Discussion

Using the multiple stakeholder sheet of the Features Prioritization Matrix seems to 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.

QUANTITATIVE ANALYSIS

We carried out a quantitative analysis on a set of 38 assessments. These assessments were carried out in the period September 2010 until November 2011. Each assessment contains two parts: a part that lists the situational factors of the company and a part that contains the capabilities that are implemented in the company. For each case, the product manager or person in a similar role was interviewed. During this interview of approximately two hours, first the situational factors were retrieved, which were later cross-checked by checking the company website and, if provided, company documentation. The second part of each interview focused on the capabilities. First, the product managers were given a short introduction by presenting the competence model for SPM. This ensured that the interviewee was aware of the terminology used in the interview.

In Figure 5, an excerpt is illustrated from the situational factor list. The situational factors are divided in five categories: Business unit characteristics, Customer characteristics, Market characteristics, Product characteristics, and Stakeholder characteristics.

Figure 6 shows an excerpt of the list of capabilities. Capabilities are divided over the 15 focus areas of the SPM Competence Model (Figure 1). Each capability can be designated with a 'Yes' (the capability is implemented in the company) or 'No' (the capability is not implemented in the company).

Recoding of the Situational Factors

The first step was recoding the situational factor variables. Situational factors all had different measurement scales such as FTE for Business unit size, amount of requirements for Requirements rate and so on. In the case study, the comparison of the situational factors was done manually, but in the quantitative analysis, this was not possible. In order to make the cases and techniques easily comparable, we recoded the following situational factors:

1. Size of business unit terms (measured in FTE).
2. Number of customers.
3. Release frequency (measured in days).
4. New requirements rate (per year).

Table 4 shows how the factors are recoded. Please note that the recoding of the variables is done only for the purpose of simplifying the

Figure 5. Excerpt of the situational factor list

Situational factor	Description	Unit	Value
Business unit characteristics			
Development philosophy	The category of development philosophy the business unit follows. E.g. SCRUM, which is agile.	Is agile:	No
Size of business unit team	The total number of employees working at the business unit, expressed in FTE's (full-time equivalent). An FTE of 1.0 means that the person is equivalent to a full-time worker, while an FTE of 0.5 signals that the worker is only half-time.	Accumulated FTE of all business unit employees:	40
Customer characteristics			
Customer loyalty	The loyalty of the customer, determined by judging the likelihood that customers will not switch to another software supplier.	Low / Medium / High	High
Customer satisfaction	The level of customer satisfaction, measured on a scale of 1 to 10, where 1 is the lowest and 10 is the highest level of satisfaction.	Scale (1-10)	8
Customer variability	The percentage of customers that have customer specific features or adaptations of features implemented.	Percentage of customers that have customized features	0%
Number of customers	The number of customers that currently hold a license to the product.	Number of customers	153

Figure 6. Excerpt of the capability list

#	Cap.	Statement	Answ
		Requirements gathering	
1	A	Requirements are being gathered and registered.	Yes
2	B	All incoming requirements are stored in a central database, which is accessible to all relevant stakeholders.	Yes
3	C	All incoming requirements are automatically stored in a central database (e.g. by means of an online helpdesk).	Yes
4	D	Requirements are gathered from all relevant internal stakeholders: support, services, development, sales & marketing, research & development (parties not present in your organization can be ignored).	Yes
5	E	Customer's and prospect's requirements are being gathered and registered, and the customer or prospect is informed of the status of their requirements.	Yes
6	F	Requirements are systematically gathered from partner companies.	No
		Requirements identifying	
7	A	Market Requirements are rewritten to Product Requirements using a pre-defined template if the Market Requirement is applicable to a product.	Yes

Table 4. Recoded situational factors

	Old	New	Old	New	Old	New
Size of business unit team	<10	Small	10-49	Medium	>=50	Large
Number of customers	<100	Small	100-999	Medium	>=1000	Large
Release frequency	<31	Small	31-182	Medium	>=183	Large
New requirements rate	<120	Low	120-599	Medium	>=600	High

fitting process. The thresholds used are chosen to the best insight of the authors.

After a first pilot run in which all cases were compared with the prioritization methods, we found out that the situational factors provided were too restrictive. Methods were rejected since too many mismatches occurred, often with situational factors that had (as we believed) hardly an influence on the method prioritization choice, such as sector and product size. Therefore, we decreased the amount of situational factors that was being used in the comparison by including only those factors which we believed were required in the selection process. Table 5 shows the resulting list of situational factors per prioritization method.

Assessing the Situational Fit and Maturity Fit

In Table 6, a fragment is listed of the table in which the situational fit and process improvement fit of each case with the eight methods. We use an 'x' to indicate this method could actually be used to improve the maturity of the prioritization process. We use grey shading to show which technique(s) has the best fit with case. When we look for example at Case 1, we see that although there is a situational fit, none of the techniques can be used to improve the maturity in this particular case. Reason for this is that Case 1 already had all capabilities for the Requirements prioritization technique implemented. When we look at Case 2, we see that two techniques (Integer Linear Programming and Feature Prioritization) are suitable from both a situational as well as an improve-

Table 5. Situational factors of the prioritization methods

	BSL	QFD	WinWin	Moscow	C-V	RT	ILP	FPM
Size of business unit team		Large		Small		Medium		Medium
Number of customers		High						
Release frequency		Low	Low	High	Medium		Medium	Medium
Variability of feature requests	Low	Low			Low			
Development platform maturity							High	
New requirements rate	High	Low	Low	Medium	Low	High	High	Medium
Company policy		High		Low		High		
Customer involvement	Low	High	High		High			
Partner involvement	Low				High	High	High	Low

Table 6. Excerpt of situational fit and process improvement fit table

Case #	BPL	QFD	Winwin	Moscow	Cost-Value	Triage	ILP	FPM
1								
2				x	x	x	x	x
3	x	x	x	x	x	x	x	x
4	x	x	x	x	x	x	x	x
5		x	x	x	x	x	x	x
6		x	x	x	x	x	x	x
7		x	x	x	x	x	x	x

Table 7. Summary of the case-method mapping

Number of suitable methods	0	1	2	3	4
Amount of cases	6	17	9	5	1

ment perspective. This also holds for the other cases in the example.

Selection of the Best Fitted Method(s)

In Table 7 we show the summary of the 38 cases. The average amount of methods that had a fit (number of suitable techniques) with a case was 1.46, with a minimum of 0 (in six cases) and a maximum of 4 (in one case). Four of the six cases with 0 suitable techniques already achieved full maturity on the Requirements Prioritization process. In the other two cases there was a misfit between situational profile and suggested methods based on the maturity level.

Table 8 shows how many cases fitted each prioritization method. The features prioritization matrix is linked to most cases. Most important reason is that it is one of the two methods that covers all five capabilities. This ensures that the method is suitable for all cases, also those that are quite mature. The other method that covers all cases is the Integer Linear Programming approach. This method is only linked to two cases. Most important reason for this is that this method is especially suitable for companies with a high partner involvement, which was not the case for many companies in the data set. The same holds for the cost-value approach. Three other methods that scored low are the Quality Function Deployment approach, the Binary Priority List and the Moscow approach. Main reason for this is that the methods only cover two or three SPM capabilities. Moreover, in the case of the Quality Function Deployment, seven situational factors have been assigned, whereas the others have three to five factors assigned to them. The more factors are assigned to a method, the more difficult it is to find a situational fit.

DISCUSSION

The results show that situational fit can be used as a selection mechanism for methods or method fragments. However, this is only a first exploration and several limitations should be taken into account. First, assigning the situational factor values to the prioritization methods has been done by the authors. This designation was done based on experience and on the indicators in the documentation in which the methods were described, but there might be discussion about some of the values. The same holds for

Table 8. Situational and process improvement fit table

Methods	BPL	QFD	Winwin	Moscow	Cost-value	Triage	ILP	FP
Cases	2	0	10	2	0	13	2	25

the thresholds in Table 4. However, since the objective of this research is to show how the mechanism of situational fit and incremental method engineering works, we do not see this as a threat to the reliability of this research.

Secondly, the amount of situational factors that has been assigned to a method has a direct influence on how often the method is linked to a case. The reason is that in this research, the situational factor effects are very strictly used. In the case of the cost-value approach this can be seen by looking at the values of customer involvement and partner involvement. Both these values have been assigned with 'High', which means that cases with low customer and partner involvement did not meet these criteria. However, this particular method is also very well usable by companies with low and medium involvement. A distinction should be made between situational factors that are *restrictive* (i.e., using the method in cases with no fit would lead to problems) and situational factors that are *encouraging* (i.e., using the method in cases with no fit would not lead to problems, but the method will lead to extra good results if there is a fit).

CONCLUSION AND FURTHER RESEARCH

In this research we used a mixed method approach to answer our research question: "How can we use situational fit and incremental method engineering to support process improvement in the software industry?" First, we carried out a case study that showed that situational fit can be used to let company improve their software processes. We found that three out of the eight methods we analyzed were from a maturity point of view interesting for the case company. By comparing the situational factors of these three methods with the situational factors of the case company, we found that one method (Wiegers' Features Prioritization Matrix) has a situational fit with the company and could be used to improve the current prioritization process. By integrating the Wiegers method fragments in the existing release planning methods, a process improvement step could be made.

The case study carried out is a first evaluation of the principle of situational fit in incremental method engineering. 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 more, we carried out a complementary quantitative analysis on a data set of 38 cases. The analysis showed us that based on maturity and situational fit, most cases could be linked to one or multiple suitable techniques. In further research, the fitting process needs refining to prevent a too restrictive effect of the situational factors.

Furthermore, it might be interesting to instead of using the current situational factors of the case company, 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 of 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.

REFERENCES

Ågerfalk, P. J., Wistrand, K., Karlsson, F., Börjesson, G., Elmberg, M., & Möller, K. (2003). Flexible processes and method configuration: Outline of a joint industry-academia research project. In *Proceedings of the 5th International Conference on Enterprise Information Systems.*

Aydin, M. N., & Harmsen, F. (2002). Making a method work for a project situation in the context of CMM. In *Proceedings of the 14th International Conference on Product Focused Software Process Improvement*, Rovaniemi, Finland (pp. 158-171).

Bebensee, T., van de Weerd, I., & Brinkkemper, S. (2010). Binary priority list for prioritizing software requirements. In R. Wieringa & A. Persson (Eds.), *Proceedings of the 6th International Working Conference on Requirements Engineering: Foundation for Software Quality* (LNCS 6182, pp. 67-78).

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 2nd International Workshop on Software Product Management* (pp. 41-48).

Bekkers, W., van de Weerd, I., Spruit, M., & Brinkkemper, S. (2010). A framework for process improvement in software product management. In *Proceedings of the International Conference on European Systems & Software Process Improvement and Innovation* (Vol. 99, pp. 1-12).

Boehm, B. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61–72. doi:10.1109/2.59

Brinkkemper, S. (1996). Method engineering: Engineering of information systems development methods and tools. *Information and Software Technology*, *38*(4), 275–280. doi:10.1016/0950-5849(95)01059-9

Burton, R. M., Lauridsen, J., & Obel, B. (2002). Return on assets loss from situational and contingency misfits. *Management Science*, *48*(11), 1461–1485. doi:10.1287/mnsc.48.11.1461.262

Clark, B. K. (2000). Quantifying the effects of process improvement on effort. *IEEE Software*, *17*(6), 65–70. doi:10.1109/52.895170

Coleman, G., & O'Connor, R. (2008). Investigating software process in practice: A grounded theory perspective. *Journal of Systems and Software*, *81*, 772–784. doi:10.1016/j.jss.2007.07.027

Conradi, R., Fernström, C., & Fuggetta, A. (1993). A conceptual framework for evolving software processes. *ACM SIGSOFT Software Engineering Notes*, *18*(4), 26–35. doi:10.1145/163626.163631

Davis, A. M. (2003). The art of requirements triage. *Computer*, *36*(3), 42–49. doi:10.1109/MC.2003.1185216

Ebert, C. (2007). The impacts of software product management. *Journal of Systems and Software*, *80*, 850–861. doi:10.1016/j.jss.2006.09.017

Harter, D. E., Krishnan, M. S., & Slaughter, S. A. (2000). Effects of process maturity on quality, cycle time, and effort in software product development. *Management Science*, *4*(46), 451–466. doi:10.1287/mnsc.46.4.451.12056

Karlsson, J., & Ryan, K. (1997). A cost-value approach for prioritizing requirements. *IEEE Software*, *14*(5), 67–74. doi:10.1109/52.605933

Kumar, K., & Welke, R. J. (1992). Methodology engineering: A proposal for situation-specific methodology construction. In Cotterman, W. W., & Senn, J. A. (Eds.), *Challenges and strategies for research in systems development* (pp. 257–269). New York, NY: John Wiley & Sons.

Lee, A. S., & Baskerville, R. L. (2003). Generalizing generalizability in information systems research. *Information Systems Research*, *14*(3), 221–243. doi:10.1287/isre.14.3.221.16560

Mizuno, S., & Akao, Y. (Eds.). (1990). *Quality function deployment: Integrating customer requirements into product design*. Portland: Productivity Press.

Nejmeh, B. A., & Riddle, W. E. (2005). A framework for coping with process evolution. In M. Li, B. Boehm, & L. J. Osterweil (Eds.), *Proceedings of the International Software Process Workshop* (LNCS 3840, pp. 302-316).

Niazi, M., Wilson, D., & Zowghi, D. (2005). A maturity model for the implementation of software process improvement: An empirical study. *Journal of Systems and Software*, *74*, 155–172. doi:10.1016/j.jss.2003.10.017

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. doi:10.1007/s11219-007-9038-z

Ralyté, J., Deneckère, R., & Rolland, C. (2003). Towards a generic model for situational method engineering. In J. Eder & M. Missikoff (Eds.), *Proceedings of the 15th International Conference* (LNCS 2681, pp. 95-110)

Rossi, M., Ramesh, B., Lyytinen, K., & Tolvanen, J. (2004). Managing evolutionary method engineering by method rationale. *Journal of the Association for Information Systems*, *5*(9), 12.

Schackmann, H., & Lichter, H. (2006). A cost-based approach to software product line management. In *Proceedings of the International Workshop on Software Product Management* (pp. 13-18).

Stapleton, J. (1999). Dynamic systems development method. In *Proceedings of the Technology of Object-Oriented Languages and Systems* (p. 406).

Stapleton, J. (2002). *DSDM business focused development*. Reading, MA: Addison-Wesley.

Tolvanen, J.-P. (1998). *Incremental method engineering with modeling tools: Theoretical principles and empirical evidence* (Unpublished doctoral dissertation). Jyväskylä Studies in Computer Science, University of Jyväskylä, Jyväskylä, Finland.

van de Weerd, I., & Brinkkemper, S. (2008). Meta-modeling for situational analysis and design methods. In Syed, M. R., & Syed, S. N. (Eds.), *Handbook of research on modern systems analysis and design technologies and applications* (pp. 38–58). Hershey, PA: Idea Group. doi:10.4018/978-1-59904-887-1.ch003

van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a reference framework for software product management. In *Proceedings of the 14th International Requirements Engineering Conference*, Minneapolis/St. Paul, MN (pp. 319-322).

van de Weerd, I., Brinkkemper, S., & Versendaal, J. (2007). Concepts for incremental method evolution: Empirical exploration and validation in requirements management. In J. Krogstie, A. Opdahl, & G. Sindre (Eds.), *Proceedings of the 19th International Conference on Advanced Information Systems Engineering* (LNCS 4495, pp. 469-484).

van de Weerd, I., Brinkkemper, S., & Versendaal, J. (2010). Incremental method evolution in global software product management: A retrospective case study. *Journal of Information & Software Technology*, *52*(7), 720–732. doi:10.1016/j.infsof.2010.03.002

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 *Proceedings of the 12th Working Conference on Requirements Engineering: Foundation for Software Quality*, Luxembourg (pp. 97-112).

van den Akker, M., Brinkkemper, S., Diepen, G., & Versendaal, J. (2008). Software product release planning through optimization and what-if analysis. *Information and Software Technology*, *50*(1-2), 101–111. doi:10.1016/j.infsof.2007.10.017

Vlaanderen, K., van de Weerd, I., & Brinkkemper, S. (2011, April 20-22). The online method engine: From process assessment to method execution. In *Proceedings of the IFIP WG 8.1 Working Conference on Method Engineering*, Paris, France.

Wiegers, K. E. (1999). First things first: Prioritizing requirements. *Software Development Magazine*, 24-30.

Yin, R. K. (2009). *Case study research: Design and methods* (4th ed.). Thousand Oaks, CA: Sage.

Inge van de Weerd is assistant professor at the Department of Information, Logistics and Innovation (ILI) at the VU University Amsterdam. She obtained her doctorate degree in 2009 at the department of Information and Computing Sciences at Utrecht University, the Netherlands. Her research interests focus on the domains of software product management, method engineering, and cloud computing. She is active as a committee member for several scientific conferences and as vice chair of the International Software Product Management Board (ISPMA).

Dominique Mirandolle is a master student in Business Informatics at Utrecht University. She holds a bachelor degree in Information Science, also from Utrecht University. During her master studies she attended ETH Zurich for one semester in the master of Management, Technology and Economics. Currently, she is writing her master thesis in the field of Software as a Service and Software Ecosystems. Previous topics she researched include User Acceptance Testing, software mash-ups and the Internet of Things. Next to her studies, she has been involved in several student associations as an active member and chief editor of a science magazine.

Sjaak Brinkkemper is Professor of Organisation and Information at the Department of Information and Computing Sciences at Utrecht University, the Netherlands. He leads a group of about thirty researchers specializing in product software development and entrepreneurship. The main research themes of the group are methodology of product software development, implementation and adoption, and business-economic aspects of the software industry. He has authored several books and serves on the editorial board of MIS Quarterly, Journal of Database Management, *and the* European Journal of Information Systems.