Software release planning

investigating the use of an advanced assessment instrument and evaluating a novel maturity framework

Slooten, R.

Award date:
2012

Eindhoven, February 2012

Software release planning: Investigating the use of an advanced assessment instrument and evaluating a novel maturity framework

by
R. Slooten

BSc Industrial Engineering and Management Science — TU/e 2010
Student identity number 0614459

in partial fulfilment of the requirements for the degree of

Master of Science
in Innovation Management

Supervisors:
dr. ir. J.J.M. Trienekens, TU/e, IS
dr.ir. K. van Oorschot, BI Norwegian Business School
ir. A. Viegers, company supervisor

TUE. School of Industrial Engineering.
Series Master Theses Innovation Management

Subject headings: software release planning, process maturity, process maturity assessment, business process modelling

Preface

This paper is the result of my master thesis project, performed in partial fulfilment of the requirements for the degree of Master of Science in Innovation Management at the Eindhoven University of Technology.

The preparations of the project started already at the end of 2010. At that time, it became clear to my boss, Marcelo de Almeida, that it was impossible to combine my part-time job with a master thesis project. He offered to look for an opportunity within the company, for me to perform my thesis project. An exciting project on software release planning was formulated. Unfortunately, he had to leave the company halfway through the project. Nevertheless, I would like to thank him for his effort to provide me with a project to complete my master degree.

Marcelo wasn't the only one who left the project halfway through. In September 2011, dr. ir. Kim van Oorschot, who was my thesis mentor, got the amazing chance to start as associate professor at the BI Norwegian Business School. This, sadly, meant she had to resign as mentor for my thesis project. However, she offered to keep involved in the project as second assessor. I would also like to thank her involvement in the project.

Furthermore, I would like to thank Arthur Viegers and dr. ir. Jos Trienekens. Arthur took on the role of company supervisor after Marcelo had left, and Jos replaced Kim as mentor from the TU/e. Thanks to them, the last phase of my thesis project could continue as planned.

I have learned a lot the past 6 months, or rather the past 5,5 years. I am grateful for the support I received from family and friends during my years at the university. They have helped me to get to this point, and stood by me every step of the way. Thank you.

Robbert Slooten

February, 2012

Management summary

Software products are of ever increasing importance in current industries. As a result, software products are increasingly being offered to a general marketplace instead of a specific customer. However, large numbers of customer improvement requests and other system aspects often result in an abundance of requirements. Preferably, all these requirements have to be implemented in the next release of the product. Yet, often too little resources are available to implement all requirements at the same time. Release planning is the process of making the decision about what new functionalities or changes will be implemented in which release of a software product (for example Bagnall, Rayward-Smith, & Whittley, 2001; Carlshamre, 2002; Maurice, Ruhe, Saliu & Ngo-The, 2006 and Saliu & Ruhe, 2005).

Although, it is clear that release planning is important to a company's success, there is little guidance available for practitioners on how to improve their release planning processes. Only one, very novel maturity framework was found that was ready to use in this project.

The Maturity Matrix

The Maturity developed by Bekkers, Van de Weerd, Spruit and Brinkkemper (2010b), defines 10 overall maturity levels four main business functions that together make up software product management (SPM). Two of these main business functions are important to the topic of release planning in this project: requirements management and release planning.

	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

FIGURE 1 THE MATURITY MATRIX (BEKKERS ET AL., 2010B)

The Maturity Matrix is, however, relatively new and has enjoyed less empirical validation than the CMMI frameworks. Therefore, this project critically evaluated the requirements management and release planning business functions of Maturity Matrix from a theoretical point of view before they were used in practice.

The theoretical evaluation turned up a number of concerns. Firstly, there were concerns on the inclusion of three focus areas in the release planning business function: scope change management, build validation and launch preparation. These focus areas don't fit with the academic literature on release planning. The following three hypotheses were formulated:

H1: Scope change management is not part of the release planning process; therefore the scope change management focus area has to be removed from the release planning business function in the Maturity Matrix.

H2: Build validation is not part of the release planning process; therefore the build validation focus area has to be removed from the release planning business function in the Maturity Matrix.

H3: Launch preparation is not part of the release planning process; therefore the launch preparation focus area has to be removed from the release planning business function in the Maturity Matrix.

Another concern regarded the assessment instrument proposed by Bekkers, Spruit, van de Weerd, van Vliet and Mahieu (2010a). They propose to use a self-assessment yes/no-questionnaire to determine the current maturity profile of a company. The organisation being assessed needs to answer the question 'Have you implemented this capability within your organisation?' for all of the 61 capabilities in the Maturity Matrix.

The use of a questionnaire is in line with the CMMI frameworks (Zubrow, Hayes, Siegel, & Goldenson, 1994) and the recommendations by De Bruijn, Freeze, Kulkarni, & Rosemann (2005) for developing your own maturity framework. However, the use of questionnaires was questioned as the best way to perform a process maturity assessment. The issue is that it may not always be possible to strictly answer only yes or no to the question 'Have you implemented this capability within your organisation?'. In addition, a literature study revealed several more serious disadvantages of using a self-assessment yes/no-questionnaire in software process assessment.

In their critical look at software capability evaluation, Bollinger and McGowan (1991) remark that in their custom process maturity assessments they don't use questionnaires at all. Instead they focus on building a detailed graphical model of the existing software process. Despite the different approach the same type of improvements are achieved. In addition, the modelling information has great value for defining exactly how and where improvements should be made. Unfortunately, further references on the use of process models in process maturity assessments were not found. Nevertheless, using process models, to collect data to base process analysis and maturity rating on, seemed very promising. This project therefore investigated the use of process models as assessment instrument. The BPMN modelling language was chosen to build the process models.

H4: Process modelling is better suited as assessment instrument than is a self-assessment yes/no-questionnaire.

Methods

To test the use of the Maturity Matrix in practice and also investigate the use of process models as assessment instrument a research project was set up within two companies. The project consisted of the following phases:

Phase 1 – Data collection and process modelling

The BPMN process modelling language was chosen to build the AS-IS process models and an iterative process modelling approach was used. In total three rounds of interviews with multiple employees were held at both companies participating in the research project. The models evolved throughout the three rounds of interviews until the company supervisors agreed on them.

Phase 2 – Process analysis and maturity rating

Using the business process models made in phase 1, the release planning process maturity was determined.

Phase 3 – Evaluation of using process models as assessment instrument
By reflecting on the use of BPMN process diagrams to analyse and improve the release planning process maturity at two companies, it was determined whether they are a better assessment instrument than the self-assessment yes/no-questionnaire proposed by Bekkers et al. (2010a).

Phase 4 – Process redesign
Using the process analysis and maturity rating done in phase 3, a process redesign proposal were formulated to improve the release planning processes at company A and B.

Phase 5 – Validation
To ensure that the final product of this project, the maturity assessment results and the change proposal, will fulfil its intended use, validation occurred throughout the project by means of regular meetings with the project supervisors. The proposed redesign was validated in a final meeting,

Results

The use of the Maturity Matrix at the two companies participating in this project turned out to be very helpful. The company supervisors were positive about results of the release planning process maturity assessment. It provided them with information about the current state of the processes and it also gave them structure to base improvement directions on.

The practical use of the Maturity Matrix confirmed H1, H2 and H3. It is recommended to consider the removal of the scope change management, build validation and release preparation focus areas from the release planning business function. These activities are very important in the process of creating and launching software products. However, they do not fit the theoretical concept of release planning.

The use of the Maturity Matrix in combination with a process modelling approach, instead of the questionnaire proposed by Bekkers et al. (2010a), proved to be successful to complete the maturity ratings. The primary benefit of using process models is the increased reliability of the maturity rating. The separation of data collection and process analysis in a maturity assessment prohibits measurement bias towards maturity levels defined in the maturity framework that is used. Also, the use of process models was shown to have advantages in putting together the improvement proposal. By adjusting the process model created for the maturity assessment according to the proposed improvements, it can more easily be shown what the impact of the improvements is. This enhances management support and also supports employee training. A clear disadvantage of using process models is the extra effort that is needed to complete the maturity assessment. However, given the promising benefits of using process models, further research is warranted.

The practical use of the Maturity Matrix also revealed a strong benefit of its design. The use of a focus area oriented model instead of a fixed-level approach has strong advantages for the guidance of software process improvements. By spreading the maturation of the single focus areas over a larger number of overall maturity levels, a fine-grained improvement plan can be put together. However, only very few maturity models have used this design of a focus area oriented model. Future research into the development of maturity models should investigate the advantages and disadvantages of different designs of maturity models.

Contents

1. Introduction

Software products are of ever increasing importance in current industries. Besides the application in PCs and laptops, software is a significant component of many physical products, for example automobiles, home appliances as well as electronic products. As a consequence an increasing part of the software produced, is aimed at being offered to a general marketplace rather than one specific customer. This type of software development is called market-driven software product development (MDSPD) (Regnell & Brinkkemper, 2005). As such, software firms now have to achieve a healthy portfolio of products and services that produce constant and sufficient profits. To help them in this pursuit, software companies often reach out to their customers directly in order to tap into what matters most to the people who will purchase their products and services. (Eisenberg, 2011).

However, customer improvement requests and other system aspects often result in an abundance of requirements. Preferably, all these requirements have to be implemented in the next release of the product. Yet, often too little resources are available to implement all requirements at the same time. Release planning is the process of making the decision about what new functionalities or changes will be implemented in which release of a software product. It aims at selecting an optimal subset of features that satisfies as many stakeholders as possible within the budget, resource and risk constraints. Proper release planning is as complex as it is important for the success of a software product (for example Bagnall, Rayward-Smith, & Whittley, 2001; Carlshamre, 2002; Maurice, Ruhe, Saliu & Ngo-The, 2006 and Saliu & Ruhe, 2005).

Although, it is clear that release planning is important to a company's success, there is little guidance available for practitioners on how to improve their release planning processes. A well-known method to support process improvement is that of process maturity models. The most famous process maturity models are the Capability Maturity Model Integration (CMMI) models. However, there is a group of scholars who claim the CMMI models aren't suitable for all organisations, particularly small and medium enterprises (SMEs). Next to that, process improvement efforts based on the CMMI models may be concentrated on irrelevant focus areas because the framework is very broad (Brodman & Johnson, 1994; Dangle, Larsen, Shaw, & Zelkowitz, 2005; Shaikh, Ahmed, Memon, & Memon, 2009). To this end, other maturity frameworks have been developed, also for release planning. The Maturity Matrix for Software Product Management (SPM) is, however, relatively new and has enjoyed less empirical validation than the CMMI frameworks. Therefore, this project critically evaluated the Maturity Matrix from both theory and practice. Additionally, the purpose of this project was to investigate the use of Business Process Modelling Notation (BPMN) process models as assessment instrument where normally questionnaires are used. It was presumed that the richer information provided by process models may lead to better results than assessment by means of questionnaires.

The report is structured as follows. Chapter 2 will introduce related work on release planning to give an understanding of the difficulties it can bring. It will also explain the concept of process maturity. The maturity framework that was used in this project in discussed in chapter 3. Chapter 4 explains why it was chosen to use process models as assessment instrument. Chapter 5 discusses the project context and methodology. The process models of the current situation at the hosting companies of this project and their process maturity ratings are presented in chapter 6. In chapter 7 the process redesigns are presented. Finally, chapter 8 and chapter 9 contain the discussion and conclusions.

2. Background

As discussed in the introduction, this project investigated an advanced method for the assessment of release planning process maturity. To give an impression of what release planning exactly is, this chapter will introduce a selected number of release planning methods and factors that play an important role in release planning. After that, the concept of process maturity will be elaborated on.

2.1. Release planning

A major problem that companies, which are developing or maintaining large and complex systems have to face, is to determine which functionalities and changes should be implemented in different releases of the software (Bagnall, Rayward-Smith, & Whittley, 2001). Release planning, the act of prioritising and selecting requirements, concerns this problem and originates from an abundance of requirements and too little resources to implement them all in the next software product release.

2.1.1. Release planning methods

A number of different release planning methods have been developed, ranging from very simple to very sophisticated. A very basic distinction in sophistication regards the scale of measurement being used. The least powerful scale to prioritise on is the ordinal scale. On an ordinal scale items are ordered according to importance so that it is possible to see which are more important than others, but it doesn't show how much more important. A far more powerful scale that can be used is the ratio scale. On this scale it is possible to quantify how much more important one item is than another (Stevens, 1946). Additionally, some methods are targeted at the complete release planning problem (i.e. selecting and planning requirements for more than one release ahead) where others are mere requirements prioritisation techniques. The examples of methods below illustrate the clear difference in level of sophistication.

MoSCoW

MoSCoW is a specific example of a grouping based prioritisation method where stakeholders place requirements in a number of groups with different priority (Berander & Andrews, 2005). It is used in the Dynamic Systems Development Method (DSDM) to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement. The categories that are used are:

Must; requirements that must be in the final product
Should; high priority requirements that preferably are in the final product
Could; nice-to-have requirements, include them if resources allows it
Would; least critical requirements, consider to include these in future releases

Using MoSCoW gives a coarse prioritisation of requirements, but it provides a good idea of what is important to stakeholders and what not. The ranking is based on an ordinal scale, since it only indicates which requirements are more important but not exactly how much more.

100 dollar game

The 100 dollar game is a prioritisation technique where the stakeholders receive 100 imaginary units to distribute between the requirements. The amount of imaginary units can also be increased in case the number of requirements calls for it. The result of the prioritisation, a ranked list of requirements, is more refined than when using MoSCoW. Next to this, the result is presented on a ratio scale, i.e. a requirement with 10 points is two times more important than one with 5 points. This gives a better understanding of differences in priorities amongst requirements (Berander & Andrews, 2005).

Value-cost analysis

To prioritise candidate requirements, value-cost analysis uses the analytic hierarchy process (AHP). AHP is member of the family of additive weighting methods and was originally proposed by Saaty (1980). AHP has proven to be very helpful in release planning. It prioritises software requirements by comparing all unique pairs of requirements to determine which of the two is of higher priority, and to what extent. In a software development project containing n requirements, $n(n-1)/2$ comparisons are needed to evaluate all possible pairs (Karlsson, Wohlin, & Regnell, 1998). By performing the AHP for both value and cost, it is possible to calculate a value-cost ratio for each requirement which can be plotted in figure 1. The figure shows three segments: requirements with a high ratio of value to cost (a value-cost ratio higher than 2), requirements with a medium ratio of value to cost (a value-cost ratio of between 0,5 and 2), and requirements with a low ratio of value to cost (a value-cost ratio lower than 0,5). Using these categories, software managers are able to effectively and accurately prioritise and select requirements (Karlsson & Ryan, 1997).

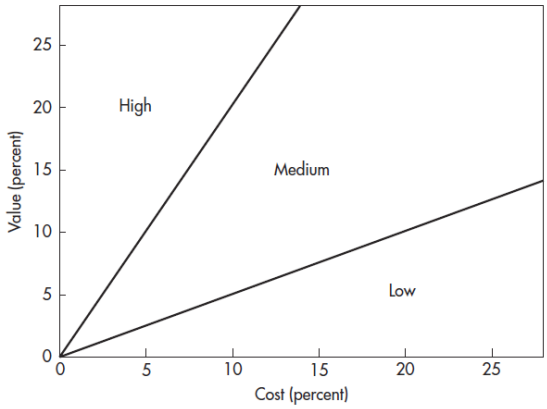


FIGURE 1 THE VALUE-COST DIAGRAM (KARLSSON & RYAN, 1997)

Evolve+

The methods discussed above only prioritise requirements; they don't assist in actually producing a release plan by selecting requirements. Evolve+ is an example of more sophisticated methods that do. The EVOLVE family of release planning methods uses genetic algorithms in combination with an iterative approach to solve the release planning problem. Examples of criteria that can be taken into account are risk of implementation, resource consumption, stakeholder satisfaction and competitiveness. Altogether, the release planning problem is stated as (Ngo-The & Ruhe, 2008):

Determine x^* *such that Utility*$(x^*) = max\,\{Utility(x): x \in X_{hard}$ *&* x *fulfils constraints of* X_{soft} *sufficiently well}.*

x^* is the set of requirements that maximises the utility function (which can include multiple variables, for example value) and satisfies all hard constraints (for example resource and budget constraints) and sufficiently satisfies the soft constraints (for example risk or competitiveness). EVOLVE+ automatically calculates solutions and the user can iteratively select the best one. Tool support to apply EVOLVE+ in practice has been developed; the web-based decision support system is called ReleasePlanner (http://www.releaseplanner.com).

The examples above show that the sophistication of the prioritisation method used can vary a lot. Sophistication depends on the method itself, but also on the aspects that are used to prioritise the requirements on. The next section takes a closer look at this.

2.1.2. Aspects influencing release planning

Many different aspects can be taken into account when prioritising requirements. These aspects are often a property or attribute related to the development project. Common aspects are importance, penalty, cost, time and risk. Using only one aspect to base prioritisation on, will make it easy to establish which requirement is most desirable, involving multiple aspects will complicate the decision-making process (Berander & Andrews, 2005).

The only large empirical study on aspects influencing requirements selection found in the preparation for this project is done by Wohlin & Aurum (2005, 2006). They investigated which criteria are used in practice to select requirements that maximise product value. The survey used in the study eventually included 13 criteria covering three important dimensions or stakeholder groups; external market/customer, company management, and development/maintenance personnel. In addition to the 13 criteria mentioned, respondents were free to add criteria of their own if they felt something was missing. In the survey, respondents were asked to rate the criteria on importance as they are used today, but also how they thought the criteria should be used in the future. The 13 criteria and average percentage values for importance according to the survey are presented in table 1 (Wohlin & Aurum, 2005).

The findings indicate that aspects related to specific customers/markets are most important, as are traditional management aspects such as cost-benefit and delivery date. The evolution/maintenance aspects have low influence on the decision making process, suggesting less attention is paid to the system perspective. In table 1, the averages of all respondents are shown. Nevertheless, the differences between the companies are not large. Companies have very similar views as to what is important when deciding whether or not to include a specific requirement in the next release. This makes the results particularly interesting, because it points to the possibility of a pattern, or common trend in views, across the software development industry (Wohlin & Aurum, 2005).

TABLE 1 AVERAGE PERCENTAGE VALUES FOR IMPORTANCE OF DIFFERENT CRITERIA TODAY AND FUTURE (WOHLIN & AURUM, 2005)

Criterion	Importance Today	Importance future
Stakeholder priority of requirement	16	17,8
Delivery data/Calendar time	14,8	12,4
Requirement's issuer	13,8	9,5
Development cost-benefit	11,1	9,8
Resources/competencies	7,7	7,4
Competitors	7,4	8,3
Complexity	6,6	5
Requirements dependencies	5,8	5
System impact	4,7	6,1
Requirement volatility	3,5	3,6
Evolution	3,5	5,5
Maintenance	3,4	5,3
Support for Education/Training	1,7	4,3

The previous discussion shows that a large number of factors influence the release planning process. It matters which method is used and which criteria are incorporated. However, irrelevant of how the release planning process takes shape, it is important that customers and developers collaborate on

requirements prioritisation. Developers can't always judge accurately which requirements are most important to the customers, and customers are not fully aware of the costs and technical effort associated with the implementation of specific requirements. When determining priorities, a balance between customer requirements and system requirements must be maintained. The architectural foundation for future developments of the system must not be sacrificed for high customer satisfaction in the short run. System developers may claim priorities are redundant, since setting priorities conflicts with their 'we can do it all' attitude. Nevertheless, product managers must ensure priorities are set early in the project as it helps to make important trade-off decisions during the development process, rather than in crash mode at the end. Ceasing development on a feature that already made it halfway to completion squanders highly valuable and limited resources (Wiegers K., 1999).

2.1.3. Release planning process improvement guidance

From the previous sections it has become clear that release planning is important to a company's success. Also, many methods exist to address the issue of release planning. There is, however, very little guidance available for practitioners on how to improve their release planning processes. An elaborate literate study revealed only two frameworks that provide guidance on the design and improvement of the release planning process.

Both frameworks that were found are very novel maturity models. The first one, the Capability Model for the Software Release Planning Process by Lindgren, Land, Norström, & Wall (2008), has not yet reached a state of completion that it can actually be used in practice. Development of the second framework, the Maturity Matrix by Bekkers, Van de Weerd, Spruit and Brinkkemper (2010b), has progressed to a state that it can be used in practice. This framework is, however, also still in an early stage of validation.

To support the development of guidance for the improvement of release planning processes, this project critically evaluated the Maturity Matrix from theory and in practice. Since process maturity plays a central role in the project, the next section will introduce the concept of process maturity. After that the Maturity Matrix will be discussed in more depth.

2.2. Process maturity

The concept of maturity is often defined as a state of development or measure of ripeness. The main idea of process maturity models is that they describe the typical behaviour of an organisation at different levels of maturity for a number of functional areas. Maturity models have been proposed for multiple corporate activities such as quality management, software development, supplier relationships and more. One of the earliest notions of a maturity grid is that of Crosby's Quality Management Maturity Grid (QMMG). It describes typical behaviour for six aspects of quality management at five different maturity levels. Perhaps the best known derivative from this work is the line of Capability Maturity Models (CMM) of the Software Engineering Institute (SEI) (Fraser, Moultrie, & Gregory, 2002). Although numerous other maturity models for software process improvement exist as well, for example ISO's SPICE and 9001, studies show that the CMM framework is used most to initiate software improvement efforts (Pino, García, & Piattini, 2008). The next section will therefore use the Capability Maturity Model Integration (CMMI) models, successors of the CMM models, to elaborate on the concept of process maturity.

2.2.1. CMMI for Development V1.3

CMMI for Development is a reference model that covers activities for developing both products and services. Organisations from many industries can use it, examples are aerospace, banking, computer hardware, software, automobile manufacturing, and telecommunications. CMMI for Development covers project management, process management, systems engineering, hardware engineering, software engineering, and other supporting processes used in development and maintenance practices (CMMI Product Team, 2010).

Maturity levels

The maturity level of an organisation is a way to measure its performance. Like most other well-known maturity models, CMMI for Development V1.3 uses a 5-staged model to represent process maturity, see figure 2. Experience has shown that organisational process improvement works best when efforts are focussed on a manageable number of process areas at a time. Therefore, each maturity level consists of related specific and generic practices for a predefined set of process areas. Progressing from one maturity level to another matures an important subset of an organisation's processes. In this way organisational process improvement is incrementally effectuated. For example, the CMMI framework defines 22 process areas, each with a number of goals and practices. To progress from maturity level 1 to level 2, a company has to achieve each specific goal of the following process areas (CMMI Product Team, 2010):

- Configuration Management
- Measurement and Analysis
- Project Monitoring and Control
- Project Planning
- Process and Product Quality Assurance
- Requirements Management
- Supplier Agreement Management

In turn, progressing to level 3 would require a company to achieve all specific goals of 11 different process areas.

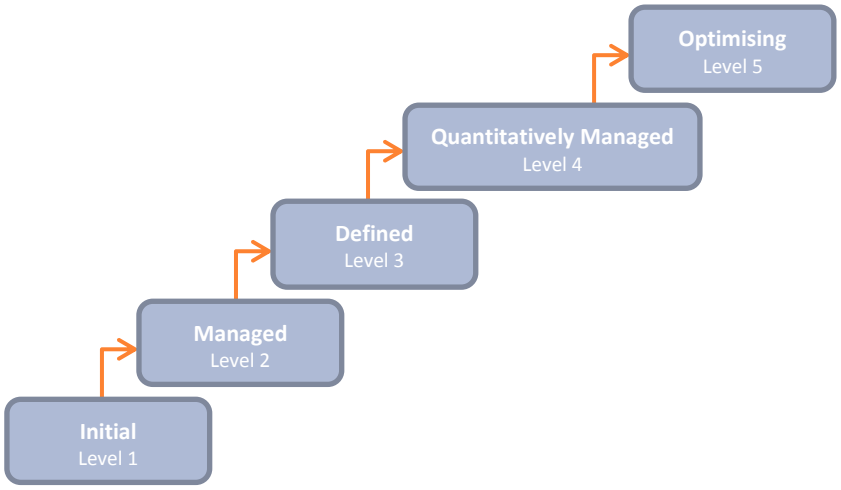


FIGURE 2 MATURITY MODEL WITH FIVE STAGES (PAULK, CURTIS, CHRISSIS, & WEBER, 1993)

Weaknesses

Although very popular and well-known, the CMM models have been argued not to be suitable for small organisations. Basically, the argument goes that CMM has been developed based on the practices of large organisations and therefore doesn't suit the needs of small businesses. SMEs don't have the resources that are needed to fully commit to the process improvements and it is too costly to comply with all the guidelines. Next to that, organisations might be making software process improvement (SPI) efforts for the wrong reasons. Sometimes, organisations are required to be at a specified CMM level to bid on contracts. Many organisations initiate process improvement programs to raise their CMM levels just because of that. Increasing the maturity level of a company just for the sake of it can lead to working on process areas having no importance for organisational performance, but it is a necessity for reaching a maturity level (Brodman & Johnson, 1994; Dangle, Larsen, Shaw, & Zelkowitz, 2005; Shaikh, Ahmed, Memon, & Memon, 2009).

In addition, the CMMI frameworks are very broad and not suitable to use in a maturity assessment specifically targeted at, for example, release planning. It is because of this reason, and the weaknesses mentioned above, that this project investigated the Maturity Matrix developed by Bekkers et al. (2010b).

The next chapter introduces the Maturity Matrix. Furthermore, it also contains a critical evaluation of the development and contents of the Maturity Matrix that was conducted before the framework was used in practice in this project.

3. The Maturity Matrix

The Maturity Matrix is part of a larger framework called the Framework for Process Improvement in Software Product Management developed by Van de Weerd, Brinkkemper, Nieuwenhuis, Versendaal and Bijlsma (2006). This framework defines four main business functions that together make up software product management (SPM). For each of the business functions, also a number of important focus areas have been defined. A figure of the framework is presented in Appendix A The SPM Competence Model. Based on this framework by Van de Weerd et al. (2006), Bekkers et al. (2010b) developed the Maturity Matrix, see figure 3. As can be seen in this figure, one of the business functions defined this model is release planning. This will be the focus of this project.

	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

FIGURE 3 MATURITY MATRIX FOR SOFTWARE PRODUCT MANAGEMENT (BEKKERS ET AL., 2010B)

In the leftmost column of the Maturity Matrix, the four main business functions of SPM and their focus areas are presented. Focus areas are important processes within a business function, for example release planning. For each focus area, a number of maturity levels have been defined, called capabilities, and are represented by the letters in the rows. The capabilities mentioned in each row are specific to the focus area in the header of the row. Looking at column 1, capability A of requirements gathering is thus completely different from capability A of launch preparation. Descriptions of each capability in the Maturity Matrix can be found in Appendix B: Maturity Matrix Capabilities.

Besides the focus area specific capabilities, the Maturity Matrix defines 10 overall maturity levels (see the top row in figure 3). To progress through maturity levels 1 to 10 in the top row, the capabilities indicated in each respective column must be achieved. To achieve maturity level 1, capabilities A of both requirements gathering and launch preparation thus have to be achieved. Like in the CMMI model, to mature one level only a selection of focus areas have to be improved. However, progressing from 1 to 10, each focus area is revisited multiple times, incrementally maturing the focus areas as well. This is unlike the CMMI models where each process area is only addressed in one specific maturity level.

Since the Maturity Matrix is still in an early stage of validation, a critical evaluation of its development and contents was conducted. This is discussed in the following sections.

3.1. Evaluation of the development of the Maturity Matrix

The evaluation of the development of the Maturity Matrix was guided by the work by Becker, Knackstedt, & Pöppelbuß (2009). Based on guidelines for design science by Hevner, March, Park & Ram (2004), they developed 7 requirements for the design of maturity models. These are summarised in table 2. Each of the requirements will be discussed below.

TABLE 2 REQUIREMENTS FOR THE DESIGN OF A MATURITY MODEL (BECKER, KNACKSTEDT, & PÖPPELBUß, 2009)

Requirements for the design of a maturity model		Descriptions
R1	Comparison with existing maturity models	The need for the development of a new maturity model must be substantiated by a comparison with existing models. The new model may also just be an improvement of an already existing one.
R2	Iterative procedure	Maturity models must be developed iteratively, i.e., step by step.
R3	Evaluation	All principles and premises for the development of a maturity model, as well as usefulness, quality and effectiveness of the artifact, must be evaluated iteratively.
R4	Multi-methodological procedure	The development of maturity models employs a variety of research methods, the use of which needs to be well-founded and finely attuned.
R5	Identification of problem relevance	The relevance of the problem solution proposed by the projected maturity model for researchers and/or practitioners must be demonstrated.
R6	Problem definition	The prospective application domain of the maturity model, as well as the conditions for its application and the intended benefits, must be determined prior to design.
R7	Targeted presentation of results	The presentation of the maturity model must be targeted with regard to the conditions of its application and the needs of its users.

3.1.1. R5 Identification of problem relevance & R6 problem definition

As indicated before in this paper, there is very little guidance for practitioners to help them improve their release planning process. The aim of the Maturity Matrix is to address this issue. The Maturity

Matrix, and the larger framework which it is part of, is intended to be used by product managers in software companies. The aim of the framework is to aid product managers in improving their SPM practices. The Maturity Matrix is used to determine an organisation's SPM maturity level and identify the areas that need improvement to reach a higher maturity level. All kinds of organisations, including small and medium sized organisations, should be able to use the Maturity Matrix as a guide for incremental process improvement in SPM (Bekkers et al., 2010b). The problem relevance and definition are thus clear.

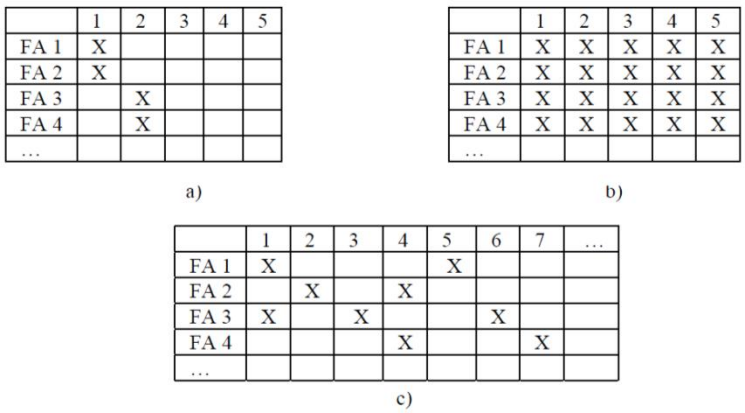
3.1.2. R1 Comparison with existing maturity models

To compare the Maturity Matrix to existing models the work by Fraser, Moultrie, & Gregory (2002) and Maier, Moultrie, & Clarkson (2011) was consulted. Together, they provide an overview of a substantial number of existing maturity models. A maturity model with the same audience, aim and scope could, however, not be found. There are models that target adjacent areas like, for example, business process management and new product development, but not specifically software product management. Also, the CMMI for Development framework contains parts that can be related to SPM, but its aim is much broader than that alone. The Maturity Matrix thus fills a gap that hasn't been previously filled by another model. However, a striking difference that was found with existing maturity models is the type of model that was used.

Fraser et al. (2002) reviewed a large number of maturity models and concluded that these can all be divided into two basics types or a combination of the two (Fraser et al., 2002):

- Continuous models (maturity grids)
- Staged model (CMM-like models)
- Hybrids and Likert-like questionnaires

Van Steenbergen, Van den Berg & Brinkkemper (2009) added a third type of maturity model to this classification: the focus area oriented model. This type of model was first used for Test Process Improvement (TPI) by Koomen and Pol (1999) and later by Van Steenbergen, Brinkkemper and Van den Berg (2007) to build an enterprise architecture maturity model. They explain the difference in models as follows: "the differences between the types of models is illustrated in figure 4: (a) in the staged 5-level models a number of focus areas is associated with each maturity level; (b) in the continuous 5-level models each focus area has the same 5 generic maturity levels; (c) in the focus area oriented models each focus area has a number of specific maturity levels" (Van Steenbergen et al., 2009, pp. 240 – 241).



	1	2	3	4	5
FA 1	X				
FA 2	X				
FA 3		X			
FA 4		X			
…					

a)

	1	2	3	4	5
FA 1	X	X	X	X	X
FA 2	X	X	X	X	X
FA 3	X	X	X	X	X
FA 4	X	X	X	X	X
…					

b)

	1	2	3	4	5	6	7	…
FA 1	X				X			
FA 2		X		X				
FA 3	X		X			X		
FA 4				X			X	
…								

c)

FIGURE 4 THREE KINDS OF MATURITY MODELS (VAN STEENBERGEN ET AL., 2009)

Bekkers et al. (2010b) chose to use the focus area oriented model for the same reasons that Van Steenbergen et al. (2009) favour this design: "it allows a more fine-grained approach, making it more suitable to our purpose of developing and improving the architectural practice, rather than merely assessing its current maturity. The focus area oriented model makes it possible to distinguish more than five overall stages of maturity. This results in smaller steps between the stages, providing more detailed guidance to setting priorities in developing the architectural practice. Departing from the five fixed maturity levels makes the focus area oriented model more flexible in defining both focus areas and interdependencies between focus areas. In our opinion this better fits the current state of maturity of the architectural practice, where complex combinations of many different factors determine its success" (Van Steenbergen et al., 2009, pp. 241-242).

In addition, Bekkers et al. (2010b) base their choice for the focus area oriented model on the argument that that the popular CMMI models and ISO's SPICE model, that have been frequently accused of being too heavy and large to use, share the design of a fixed-level model. They therefore depart from this design. Although multiple researchers indeed agree on the fact that CMM(I) and SPICE have their shortcomings, there is hardly any literature available that suggest the fault lays in the design of the models; i.e. the use of a continuous or staged 5-level model. The choice of Bekkers et al. (2010b) to move away from the fixed-level maturity model design of model is thus not reflected broadly in other literature. One example though, is a study of Sommerville and Ransom (2005). They concluded that to provide effective feedback on maturity improvement, a more continuous maturity model is needed than the continuous 3-level model they used. This is in line with the reasons provided by Bekkers et al. (2010b) to choose for a focus area oriented model.

Opting to develop a focus area oriented model is thus not in line with the large body of literature available on maturity models. However, some research showed that this approach is necessary to provide a finer-granularity improvement advised needed in practice.

What is very interesting, is that the Capability Model for the Software Release Planning Process by Lindgren et al. (2008), the only other maturity framework found that also addresses release planning, has a focus area oriented model design as well. Although they don't explicitly discuss their design choice, Lindgren et al. (2008) indicate to have been inspired by the TPI framework by Koomen & Pol (1999). So although the concept of a focus area oriented maturity model is not new, it has now inspired two recent and related efforts in designing a maturity model, where the rest of the scientific community hasn't picked up the concept of a focus area oriented model before. Theoretically speaking, there is no reason why fixed-level maturity models couldn't work for software processes, which would explain the choice of both efforts to use a focus area oriented model. Perhaps the use of the Maturity Matrix in practice can show why Bekkers et al. (2010b) and Lindgren et al. (2006) chose to use the focus area oriented design. And if it turns out it has strong advantages, the question still remains why the scientific community hasn't picked up this design on a larger scale.

3.1.3. R2 Iterative procedure & R4 Multi-methodological procedure

According to Becker et al. (2009), a maturity model must be developed iteratively. In addition a multi-methodological procedure must be adopted. In the design of the Maturity Matrix, Bekkers et al. (2010b) first performed a literature review into the specific processes within the field of SPM. Secondly, a brainstorm session was held with experts from the scientific community to create a first version of the Maturity Matrix. This version was subjected to expert validation, where business professionals provided their input. Finally, a survey amongst software companies and the SPM

community as a whole was performed to fine-tune the positioning of the capabilities in the Maturity Matrix. The design process followed by Bekkers et al. (2010b) thus meets both criteria R2 and R4.

3.1.4. R3 Evaluation

As remarked before, the Maturity Matrix is still in early stages of validation and evaluation is thus still on-going. The section above already described that the Maturity Matrix has been evaluated during the iterative development procedure followed by Bekkers et al. (2010b). Business experts have provided their input through interviews and a survey. In addition, the Maturity Matrix has been applied at 5 software organisations in the Netherlands (Bekkers & Spruit, 2010). The participating organisation indicated that the Maturity Matrix provided a much needed structure for their SPM organisation. They were also positive about its usability and usefulness. The Maturity Matrix has, however, not yet been used in a situation independent from its creators.

3.1.5. R7 Targeted publication of results

To date the Maturity Matrix has only been published in conference proceeding papers. It is not clear how the creators of the model intend to spread it amongst the targeted end user: product managers in software companies.

The evaluation of the development of the Maturity Matrix has not raised serious concerns. The framework addresses a topic that is in need of attention, namely guidance in the design and improvement of the release planning process. In addition adequate procedures and methodology were used. Points of interest are the focus area oriented design and the fact that the model hasn't been used in situations independent from the creators.

Now that the development of the Maturity Matrix has been evaluated, the following section will turn to its contents. The contents of the Maturity Matrix were evaluated to see if these are in line with the available academic literature.

3.2. Evaluation of the contents of the Maturity Matrix

Since this project focusses on release planning, the evaluation of the contents of the Maturity Matrix will be restricted to the business functions *Release planning* and *Requirements management*, thus omitting the business functions *Portfolio management* and *Product roadmapping*. Although the

TABLE 3 REQUIREMENTS MANAGEMENT AND RELEASE PLANNING FOCUS AREAS OF THE MATURITY MATRIX (BEKKERS ET AL., 2010B)

	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 organising				A		B		C			
Release planning											
Requirements prioritisation			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

focus of the project is on release planning, some focus areas in the *Release planning* business function have a prerequisite capability in the *Requirements management* business function. *Requirements management* is therefore included in the evaluation and the remainder of this paper.

3.2.1. Focus areas

The focus areas contained in the business functions *Requirements management* and *Release planning* and their capabilities are depicted in table 3.

Requirements management

Although there is no common definition of the requirements engineering process, there is general agreement on the identification of four tasks to be performed (Kavakli & Loucopoulos, 2004 & Kotonya & Sommerville, 2002):

1) **Requirements elicitation**; system requirements are discovered through consultation with stakeholders, from system documents and other sources of knowledge.
2) **Requirements analysis & negotiation**; requirements are analysed in detail to find conflicts and different stakeholders negotiate to decide on which requirements are to be accepted.
3) **Requirements documentation**; the agreed requirements are documented at an appropriate level of detail.
4) **Requirements validation**; requirements are carefully checked for consistency and completeness before they are used for the system development.

Though differently grouped and formulated, the focus areas in the requirements management business function largely cover these four tasks. An exception is requirements negotiation, which is subdivided to the release planning business function in the Maturity Matrix. Requirements gathering is a contraction of the tasks requirements elicitation and documentation. Requirements identification is a contraction of the tasks requirements analysis and validation. Based on theoretical grounds, there is no reason to group the tasks in this manner. The reason to do this must thus have derived from practical concerns; the use of the Maturity Matrix in practice during this project can maybe clarify this. The requirements organising focus area doesn't really fit with the four tasks mentioned above. However, elements of the descriptions by Bekkers et al. (2010b) can be recognised in the requirements development process area of the CMMI for Development framework. Requirements organising is likely derived from practical matter. Its inclusion in the framework will be evaluated later in this paper.

Release planning

Looking at the focus areas in the release planning business function, a division over three, product development cycle related phases emerges. The requirements prioritisation, release definition and release definition validation focus areas occur in the preparation before the development of a product starts. Scope change management occurs during the development of a product. Build validation and launch preparation occur after the development of a product has finished.

Earlier sections of this paper have already shown that release planning has received much attention in scientific literature. This is predominantly focussed on the prioritisation and selection of requirements. These actions are captured by the requirements prioritisation and release definition focus areas in the Maturity Matrix. Release definition validation is a logical derivative from the release definition activity in a commercial environment, where multiple stakeholders have to approve the plans for a new version of a product. This topic is however not discussed in scientific

literature. Scope change management has received some attention amongst scholars, but not at great length. Build validation and release preparation also don't appear in the scientific literature under the topic of release planning. Amongst scholars release planning concerns activities working towards the development of a product and also marginally the change management during the development, but not the phases after development. The inclusion of these business functions thus seems to be derived from practice. This is reflected by the work of Lindgren et al. (2008) mentioned earlier.

Lindgren et al. (2008) started the development of a maturity framework specifically for the software release planning process; the Capability Model for the Software Release Planning Process. Based on industrial case studies they derived five key-areas for release planning (Lindgren et al., 2008):

1) **Need elicitation**; refers to how and from which stakeholders needs are elicited
2) **Need documentation**; for a need to become eligible for prioritisation into a release there typically needs to be some documentation, e.g. a short description of what business benefit the need aims to fulfil
3) **Decision material**; the purpose of the decision material, produced via studies, is both for increasing confidence of data and risk reduction. Typically this is related to, e.g., refining cost-estimates, determine consequences for the existing system, refining return-of-investment calculus
4) **Product strategy**; having a well-defined and clearly communicated product strategy often helps employees within an organisation to know what to strive for
5) **Release plan decision**; the needs elicited, including its documentation and decision material, is used as a basis for prioritisation and followed by a decision of what needs to be included in the next release

The key-areas defined by Lindgren et al. (2008) affirm the earlier derived notion that release planning mainly concerns activities leading up to the development of a product. On the other hand, their framework does contain key-areas that appear in the Reference Framework for SPM in other business functions than release planning; specifically need elicitation, need documentation and product strategy. This confirms that release planning is part of a larger process that Van de Weerd et al. (2006) define as Software Product Management (SPM).

A couple of issues have been derived from the evaluation of the contents of the Maturity Matrix. The inclusion of the scope change management, build validation and launch preparation focus areas in the release planning business function departs from the accepted definitions in the scientific literature. The following hypotheses are therefore posited:

H1: Scope change management is not part of the release planning process; therefore the scope change management focus area has to be removed from the release planning business function in the Maturity Matrix.

H2: Build validation is not part of the release planning process; therefore the build validation focus area has to be removed from the release planning business function in the Maturity Matrix.

H3: Launch preparation is not part of the release planning process; therefore the launch preparation focus area has to be removed from the release planning business function in the Maturity Matrix.

Now that the Maturity Matrix has been critically evaluated, the next chapter will turn to the instrument through which data is gathered to perform the maturity assessment. Bekkers, Spruit, van

de Weerd, van Vliet and Mahieu (2010a) propose to use a self-assessment yes/no-questionnaire. This is, however, questioned as the best way of data collection for a maturity assessment.

4. The assessment instrument

The Maturity Matrix is part of a larger research effort into the Situational Assessment Method (SAM). Part of this method is to determine the so called current capability profile (CCP) to show which capabilities are implemented within an organisation, or in other words, determine the current process maturity. The SAM method uses *the capability questionnaire* to build the CCP. It consists of one statement for each capability in the Maturity Matrix and the organisation being assessed needs to answer the question 'Have you implemented this capability within your organisation?' with either yes or no for each statement. For a company to answer yes, Bekkers et al. (2010a) have defined two criteria:

- **All capabilities must be reoccurring**; the process must be executed on a reoccurring and planned basis, and not ad hoc. If a capability is not executed on a regular, predetermined basis, then the capability is not satisfied.
- **All capabilities must be documented**; a detailed description of the processes must be described in a document for all parties involved in the capability. All parties involved in the capability must at least have access to the (part of) the process describing the actions that are required of them.

Bekkers et al. (2010a) don't elaborate on their choice to use a questionnaire besides indicating it is a quick and easy way to assess process maturity that requires little effort from the company being assessed. Currently there are 61 close-ended questions in the questionnaire (Bekkers et al., 2010a).

The use of a questionnaire is in line with the CMM frameworks (Zubrow, Hayes, Siegel, & Goldenson, 1994) and the recommendations by De Bruijn, Freeze, Kulkarni, & Rosemann (2005). Based on their own experience in making a maturity model, they propose to use a questionnaire as assessment instrument. When selecting an instrument for conducting an assessment, model generalizability and resources available for the assessment need to be considered. The use of a questionnaire that can be delivered and collected electronically is recommended. Advantages of using a questionnaire are that it can be distributed easily to a wide range of respondents across geographic boundaries, it is non-invasive, and it is cost and time efficient (Brewerton, 2001 and Gillham, 2000).

However, the use of this questionnaire was questioned as the best way to perform the process maturity assessment. The creators of the Maturity Matrix and the capability questionnaire also express some concerns themselves (Bekkers et al., 2010a and Bekkers & Spruit, 2010). The issue is that it may not always be possible to strictly answer only yes or no to the question 'Have you implemented this capability within your organisation?'. Participants in the case study by Bekkers & Spruit (2010) reported that there were some capabilities which they often performed, but not all of the time. They now had to answer these questions with no, where they would like to answer them with 'in most cases'.

In addition, a literature study revealed several more serious disadvantages of using a self-assessment yes/no-questionnaire in software process assessment:

- The CMM Maturity Questionnaire has been found to be repetitive and verbose (Cattaneo, Fuggetta, & Lavazza, 1995)

- Questions in the CMM Maturity Questionnaire have been found to not be related to the real problem (Bollinger & McGowan, 1991; Cattaneo, Fuggetta, & Lavazza, 1995)
- Collecting information on the software process is iterative in nature (Bandinelli, Fuggetta, Lavazza, Loi, & Picco, 1995; Sommerville & Ransom, 2005)
- Qualitative information is needed to reflect the software process (Sommerville & Ransom, 2005)
- Finer granularity than yes/no questions is needed to reflect the software process (Sommerville & Ransom, 2005; Habra, Alexandre, Desharnais, Laporty, & Renault, 2008; Pino, Pardo, García, & Piattini, 2010; Wiegers & Sturzenberger, 2000)
- More dimensions than 'is it implemented?' are needed to reflect the software process (Niazi, Cox, & Verner, 2008)
- Assistance of an external assessor/consultant is advised (Cater-Steel, Toleman, & Rout, 2006; Habra, Alexandre, Desharnais, Laporty, & Renault, 2008)
- Questionnaires don't produce the objective evidence necessary to produce valid findings in a formal assessment since they don't allow corroboration (Gray, Sampaio, & Benediktsson, 2005)

In addition, Bekkers et al. (2010b) state the aim of the Maturity Matrix is to aid software product managers in improving their SPM practices. The assessment method should thus not only be aimed at assessing an organisation's process maturity, like a questionnaire is, but should aim at supporting the complete improvement process. Given this aim and the disadvantages of a self-assessment yes/no-questionnaire listed above, this project investigated the use of a different assessment instrument.

4.1. Process models as assessment instrument

In their critical look at software capability evaluation, Bollinger and McGowan (1991) remark that in their custom process maturity assessments they don't use questionnaires at all. Instead they focus on building a detailed graphical model of the existing software process. Despite the different approach the same type of improvements are achieved. In addition, the modelling information has great value for defining exactly how and where improvements should be made. Unfortunately, further references on the use of process models in process maturity assessments were not found. Nevertheless, using process models to collect data to base the process analysis and maturity rating on seems very promising. Research into success factors of software process improvement (SPI) shows that some of the most important factors are (Dyba, 2005; Khokhar, Zeshan, & Aamir, 2010; Niazi, Wilson, & Zowghi, 2006; Pettersson, Ivarsson, Gorschek, & Öhman, 2008; Rainer & Hall, 2002; Stelzer & Mellis, 1998):

- **Management commitment and support**; SPI efforts need to be actively supported and management must allow resources to be dedicated to the SPI effort
- **Staff / employee involvement**; people that are part of the organisation often have insight into and knowledge about what areas are in need of improvement
- **Providing enhanced understanding**; employees need to understand how their work contributes to the corporate mission and vision
- **Training and mentoring**; appropriate training and information sessions can have a direct effect on the rate of SPI progress

According to Curtis, Kellner and Over (1992), software process modelling facilitates human understanding and communication and also supports process improvement, see table 4. Human

TABLE 4 SOFTWARE PROCESS MODELLING OBJECTIVES AND GOALS (CURTIS ET AL., 1992)

Facilitate human understanding and communication
- Represent process in form understandable by humans - Enable communication about and agreement on software processes - Formalise the process so that people can work together more effectively - Provide sufficient information to allow an individual or team to perform the intended process - Form a basis for training the intended process
Support process improvement
- Identify all the necessary components of a high-yield software development or maintenance process - Reuse well-defined and effective software processes on future projects - Compare alternative software processes - Estimate the impact of potential changes to a software process without first putting them into actual practice - Assist in the selection and incorporation of technology (e.g., tools) into a process - Facilitate organisational learning regarding effective software processes - Support managed evolution of a process

understanding and communication is of high importance in effectuating the four success factors listed above. By representing processes in a form understandable to humans, process models contribute to providing an enhanced understanding of corporate processes. Process models also aid in increasing management support by showing the impact of changes to a process without first putting them into actual practice. Furthermore, to create accurate process models employees will have to be involved since they have the best knowledge about how processes are executed.

In addition, process models can be used to analyse processes by describing the process either as a data capture or a presentation exercise. This aids learning about the process. Sometimes models are needed to make decisions on the design, changes, improvements or re-design of processes (Aguilar-Savén, 2004).

Process modelling is an iterative method that provides rich information on how processes are implemented. This overcomes the disadvantages listed for the use of a yes/no-questionnaire. Furthermore, as shown in table 4 process modelling supports the SPI success factors found in literature.

A clear disadvantage of process modelling compared to the use of a questionnaire, is the extra effort that goes into building proper process models. Process modelling is also more invasive than using a questionnaire, because employees have to be interviewed about their way of working. This may not always be appreciated. Nevertheless, given the advantages mentioned above, building process models may be worth the associated disadvantages. This results in the following hypothesis:

H4: Process modelling is better suited as assessment instrument than is a self-assessment yes/no-questionnaire.

This project therefore investigated the use of process models as data collection instrument to fill in the Maturity Matrix. The choice on which modelling language was used is discussed next.

4.2. Selecting a modelling language

A wide range of modelling types and languages exists. The first step in choosing one is determining the purpose of the model. In this case the purpose of the model was to describe current processes within a firm to determine the current process maturity and formulate improvement proposals

based on that. Aguilar-Savén (2004) classifies this as a descriptive model for learning and process development/design. A number of techniques can be used for these types of models. Based on the characteristics and strengths & weaknesses from user and modeller perspective, flowcharts and data flow diagrams (DFDs) were chosen. The benefits of these models are their communication ability and ease of understanding. Yet they provide a flexible and quick way of modelling, and they are easy to verify and draw (Aguilar-Savén, 2004).

Next the modelling language had to be chosen. It was chosen to use the Business Process Modelling Notation (BPMN) language because this is able to capture aspects of both flowcharts and DFDs. Furthermore, based on the Bunge-Wand-Weber (BWW) representation model, BPMN is the most ontological complete model. Ontological completeness refers to a language's ability to capture all aspects of a real world situation (Weber, 1997). The BWW model identifies four key constructs:

- things, including properties and types of things
- states assumed by things
- events and transformations occurring on things
- systems structured around things

The BPMN language has the highest overall score on the BWW ontology. It has an almost perfect score on the things, events and system structure constructs. It however scores zero on representing states assumed by things (Recker, Indulska, Rosemann, & Green, 2005; zur Muehlen & Indulska, 2010). The inability to represent states is not expected to be problematic since flowcharts and DFDs don't concern object state.

In sum, the chapters above have shown that release planning is very important to the success of a company, but in practice it is very hard to define the release planning process. There are numerous release planning methods to choose from and a wide range of different aspects that can be included in the release planning process. Unfortunately, there is little guidance available to help companies assess and improve their current release planning process. A novel maturity framework was found that is partially aimed at the release planning process. Given its early stage of validation a critical evaluation was conducted that raised a couple of concerns. In addition, the assessment instrument proposed to be used, a self-assessment yes/no-questionnaire, was found to have serious disadvantages. Instead, according to literature, process models are more suitable to be used in a process maturity assessment.

To investigate the use of the Maturity Matrix in a release planning process maturity assessment effort, and also the use of process models as assessment instrument, a research project was setup in two companies. The following sections introduce both companies and will also discuss the methodology that was used.

5. Methods

5.1. Project context

The research project discussed in the remainder of this paper was hosted by two companies active in providing telematics solutions to the transportation industry. They provide products to acquire mobile operations data and communicate this data between dispatch/shipment control, back-end information systems, and mobile shipping unit (e.g. a truck or trailer, etc.). Data is made available across the whole supply chain, from manufacturer/shipper through logistics and freight to the end user/consignee. The services manage and present data pertaining to real time operational data to

understand and manage every key parameter needed to assure reliable implementation of mobile activities and to minimise costs associated with its implementation.

Company A was founded in 2002. Since then, it has been growing very quickly, and has built up a solid and rapidly expanding customer base in Europe. As GPRS became available in other continents, activities in the USA, China and Brazil were initiated during the years 2004 and 2005. In 2010 company A employed about 100 people worldwide; 30 in Europe, 30 in the USA, 15 in China and 25 in Brazil. The worldwide revenue amounted up to €8 million.

Company B was founded in 1991 and has been quoted on stock exchange markets since 2007. Corporate activities of company B are spread across all of Europe, and at the end of 2010 it employed 230 people. The revenue over 2010 was just over €41 million.

In June 2011, the European activities of company A were acquired by company B. In total, company B now has over 280 employees spread throughout the whole of Europe. And with a new total of more than 80.000 devices installed, it reinforced its market leader position of providing telematics solutions to the European transportation industry.

Preparations for this project started in February 2011 and it was planned to only be conducted within the European part of company A. Plans for the project were initiated by the Product and Projects Manager who experienced dissatisfaction with the release planning processes. He requested an objective assessment to see if his dissatisfaction was justified and whether the processes could potentially be improved. In the acquisition process mentioned above the Product and Project Manager however left the company. The replacing company supervisors at both company A and B decided to carry on with the project, but extend it to also include the release planning processes within company B. This resulted from the fact that processes within company A and company B are quite different from one another.

The owners and management of company A have tried to keep processes as informal and undefined as practically possible to create an agile company. This means process control and support is largely ad hoc. Release planning is mainly done by a small group of high-positioned executives and no formal rules or guidelines are used in the process.

At company B however, processes are more defined and certain protocols are in place. Due to the stock exchange quotation a number of qualifications have to be met. Also, due to the product architecture, some formalisation was needed in the past. Requirements prioritisation is mainly the responsibility of the Vice President (VP) Product Management, but other directors, managers, salesmen and region representatives are incorporated in the process as well. The process is done according to a predefined set of guidelines that prescribe how priorities are determined.

Given the fact that these two very different approaches to the release planning process now co-exist within company B it is very interesting to assess them both. The management can use the assessment results to compare both processes and how these impact the organisation. Also potential improvement recommendations are welcome.

For the sake of simplicity, in the remainder of this document the processes of the European part of company A, thus now owned by company B, will continue to be referred to as company A. When it is

needed to refer to the rest of company A still operating outside of Europe this will be explicitly mentioned.

5.2. Methodology

As discussed before, the purpose of this project is the assessment of the release planning process maturity at both the hosting companies, to test the use of the Maturity Matrix in practice and also investigate the use of process models as assessment instrument. Additionally, improvement proposals had to be formulated based on the findings of the maturity assessment.

The first part, assessment of process maturity, can essentially be split in three steps. The first step is to collect data to base the assessment on. Secondly, the processes need to be analysed according to a maturity framework. Thirdly, the analysis according to the maturity framework must result in a maturity rating.

This led to the following project phases.

5.2.1. Project phases

Phase 1 – Data collection and process modelling

As discussed before, it was decided to use AS-IS models of the release planning processes within company A and B to determine the process maturity. Using the BMPN notation, a graphical representation (business process diagram or BPD) was thus first created of the involved actors, activities, and information flows.

The basis of the process modelling process was identical at both companies. In a first session with the persons responsible for the release planning process, the CTO at company A and the VP Product Management at company B, a list of persons of interest was put together. To collect input from various points of view, the persons selected for this list were not limited to those who are involved in the release planning process directly. Persons involved in requirements gathering and the product development process in general were thus included on the list.

Interviews with the persons of interest on the lists were now arranged. Semi-structured interviews were used because they allow a change of direction when the interviewer thinks it is necessary or important (Van Aken, Berends, & Van Der Bij, 2007). After the interviews with these persons were completed the gathered information was used to build a first version of the process models. Also, the student's experience within company A prior to this project was used to write the process descriptions and build the process models.

Subsequently, the process models were used to guide a second iteration of data collection. Again, information was gathered that was now used to adjust the process models created in the first iteration.

This process continued until the company supervisors agreed with the process models built in the latest iteration. This iterative process modelling approach is depicted in figure 5.

Within company A the first session with the company supervisor resulted in a list of 9 persons as given in Appendix C: List of interviewed persons. Within company B the initial list of persons of interest included 7 persons; all those mentioned in Appendix C: List of interviewed persons except for the product manager. The initial interviews were all conducted using the interview protocol as displayed in Appendix D: Semi-structured interview protocol.

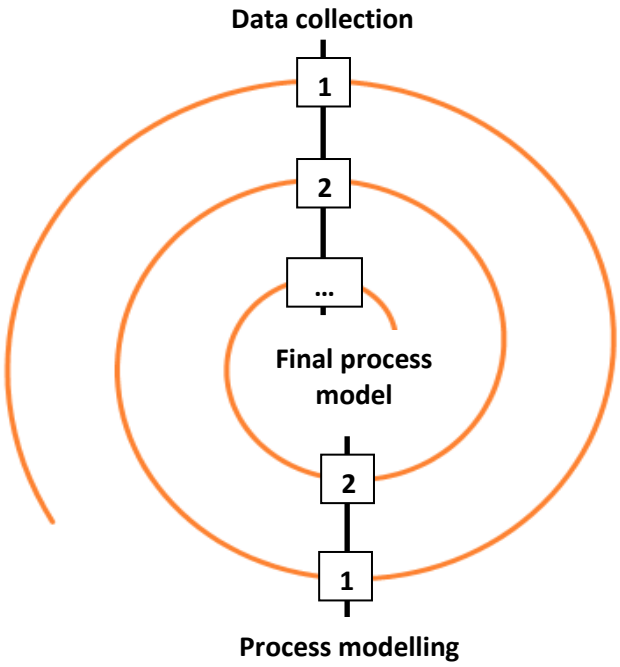


FIGURE 5 ITERATIVE PROCESS MODELLING APPROACH

In addition to the interview protocol, knowledge from prior interviews was used to elaborate on specific topics when they came up. The interviews ranged in duration from 30 minutes to one hour on average, with the exception of the interview with the Development manager at company A. During the first interview session of one hour it turned out a second session was necessary to discuss all the relevant topics, a second session of one hour was thus added.

The first version of the process models of company A were presented to the company's Sales manager and CTO. Because of the student's prior experience within the company and the ease of contact with employees within company A, the initial selection of persons to interview turned out to be exhaustive. The process models were however slightly adjusted according to their feedback. Subsequently, the second version of the process models was discussed with the CTO and Development manager. Based on their input the process models were revised a second time. The third version of the process models was again handed over to the CTO and Development manager, who now approved them.

Within company B, the first version of the process models was discussed with the VP Product Management. Based on documents acquired during the first round of interviews and in consultation with the VP Product Management, it was decided to also include a Product Manager in the interviews. Furthermore, a second interview with the VP Product Management was conducted to elaborate on topics that came up during other interviews and in the building of the process models. The extra information led to a second version of the process models that was again handed over to the VP Product Management who now approved them.

Table 5 summarises the text above. The process modelling effort at both companies shows that input from multiple stakeholders and an iterative approach is needed to get accurate process models. By having to explain corporate processes to an external party, employees are forced to stand back and think them over explicitly. In addition, using the input from multiple stakeholders can single out contradicting views on a process.

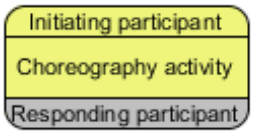
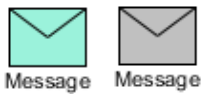
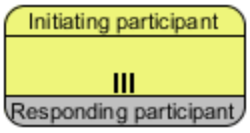
TABLE 5 NUMBER OF PEOPLE INVOLVED PER ITERATION OF THE PROCESS MODELLING PROCESS

Iteration	Number of people involved	
	Company A	Company B
1	9	7
2	3	2
3	2	1
4	2	-

Modelling notation

The BPMN notation was used to create the process models. A complete list of all the modelling elements used and their descriptions can be found in Appendix E: BPMN modelling elements. A number of extended modelling elements that are used in the models might be less familiar to the reader, table 6 therefore contains a brief descriptions of these elements.

TABLE 6 BPMN EXTENDED MODELLING ELEMENTS (OMG, 2011)

Notation	Element	Description
	User Task	A User Task is a typical "workflow" Task where a human performer performs the Task with the assistance of a software application.
	Choreography activity	A Choreography Task is an atomic Activity in a Choreography. It represents a set of one (1) or more Message exchanges. Each Choreography Task involves two (2) *Participants*. The name of the Choreography Task and each of the *Participants* are all displayed in the different bands that make up the shape's graphical notation. There are two (2) or more *Participant* Bands and one Task Name Band.
	Message	A Message is used to depict the contents of a communication between two *Participants* of a choreography task. A blue envelop indicates communication from the initiating participant. A grey envelop indicates communication from the responding participant.
	Multiple instances	A set of three vertical lines will be displayed at the bottom-centre of a choreography activity to indicate sequential Multi-Instances (i.e. more than one (1) responding participant is involved in the choreography task).

Phase 2 – Process analysis and maturity rating

Using the business process models made in phase 1, the release planning process maturity was determined. During this part of the project it was also contemplated whether the focus areas

mentioned in H1, H2 and H3 are also present at the hosting firms and are part of the release planning process.

Phase 3 – Evaluation of using process models as assessment instrument
By reflecting on the use of BPMN process diagrams in the release planning process maturity assessment at the two hosting companies, it was determined whether they are a better assessment instrument than the self-assessment yes/no-questionnaire proposed by Bekkers et al. (2010a).

Phase 4 – Process redesign
Using the process analysis and maturity rating done in phase 2, a process redesign proposal were formulated to improve the release planning processes at company A and B.

Phase 5 – Validation
To ensure that the final product of this project, the maturity assessment results and the change proposal, will fulfil its intended use, validation occurred throughout the project by means of regular meetings with the project supervisors. The proposed redesign was validated in a final meeting, where all relevant stakeholders were present. By asking domain experts their opinion about the proposed redesign, it can easily be identified whether or not the solution is feasible.

Now that the sections above have defined the project context, the project phases and the methodology used in this project, the next chapter will discuss the findings of the research project.

6. Process maturity assessments

As discussed in section 5.2.1, AS-IS process models of the current release planning processes at both companies have been made. This deviates from the normal procedures taken during a process maturity assessment using the Maturity Matrix. Section 4.1 showed that in theory process models are a better assessment instrument than the self-assessment yes/no-questionnaire proposed by Bekkers et al. (2010a). During the process of creating the models special attention was given to two items from table 4:

- Represent process in form understandable by humans
- Enable communication about and agreement on software processes

These two items were deemed to be the biggest benefit of using process models as data capturing instrument in a process maturity assessment.

6.1. Requirements management

The requirements management business function comprises the continuous management of requirements outside of releases and consists of three focus areas (Bekkers et al., 2010b):

- **Requirements gathering**; concerns the acquisition of requirements from both internal and external stakeholders
- **Requirements identification**; identifies the actual Product Requirements by rewriting the Market Requirements to understandable Product Requirements, and connecting requirements that describe similar functionality
- **Requirements organising;** structures the requirements throughout their entire lifecycle based on shared aspects, and describes the dependencies between Product Requirements

The three focus areas are too interwoven in practice to discuss them separately in a process model. The following process models and descriptions thus concern the complete requirements management business function at company A and company B.

6.1.1. Requirements management at company A

A request for a new requirement originating from an external source, primarily customers, typically enters the organisation through an implementation engineer or a member of the sales staff, to be called the 'request handler' from here on. After receiving a new requirement, the request handler will make an initial analysis of the request followed by a decision to accept it or not. If, for example, the request handler thinks the request doesn't fit at all with the products of company A, it will be turned down. When the request handler is not sure about the decision, the Sales manager is asked for a second opinion.

An employee of company A, for example a sales person or software engineer, can of course also come up with new requirements. If in this case the employee works from the Eindhoven office, he or she will contact the Product Department and then the process will continue as normal from there. Employees of the development centre in Dublin will discuss new requirements they think of with the Development manager or CTO.

Normally, a request handler will sign in a new requirement at the Product department. This usually happens by means of an email that describes the request in a few sentences. The Product department will subsequently analyse the requested feature. This is done to find and correct missing or conflicting information. In case additional information is needed the request handler is contacted. The Product department thinks about how the new functionality could look like or can be implemented in the current product. Considerations about usability, intended technological environment/partner applications and supportability are made as well. This information is put into a specification document that will be handed over to the Development manager in Dublin. The specification document is often drafted in MS Word and natural language is used.

A high priority requirement will immediately be forwarded from the request handler to the CTO or the Development manager in Dublin, usually through a telephone call or a Skype conversation. The CTO or Development manager will quickly identify product requirements and decide whether the request is indeed urgent. They will decide whether resources can be made available for accelerated development.

The part of the requirements management process described above is shown figure 6 on the next page.

The requirement specification documents coming from the Product department are further analysed by the Development manager. The focus of his analysis is more on performance, maintainability and support requirements. If in his opinion the development is possible and useful, he will make an estimation of the development effort. This is done in coordination with the software developers and CTO. If the required development turns out to be more than a couple of days, in other words project size, the requirement has to be added to the project list. If not, it will be added to Redmine, a project management tool, as a new feature. In Redmine all the different product types are classified, new features will be allocated to the product group they belong to. The Product department will receive confirmation of this and they will also get the Redmine identifier. See figure 7 for the process model.

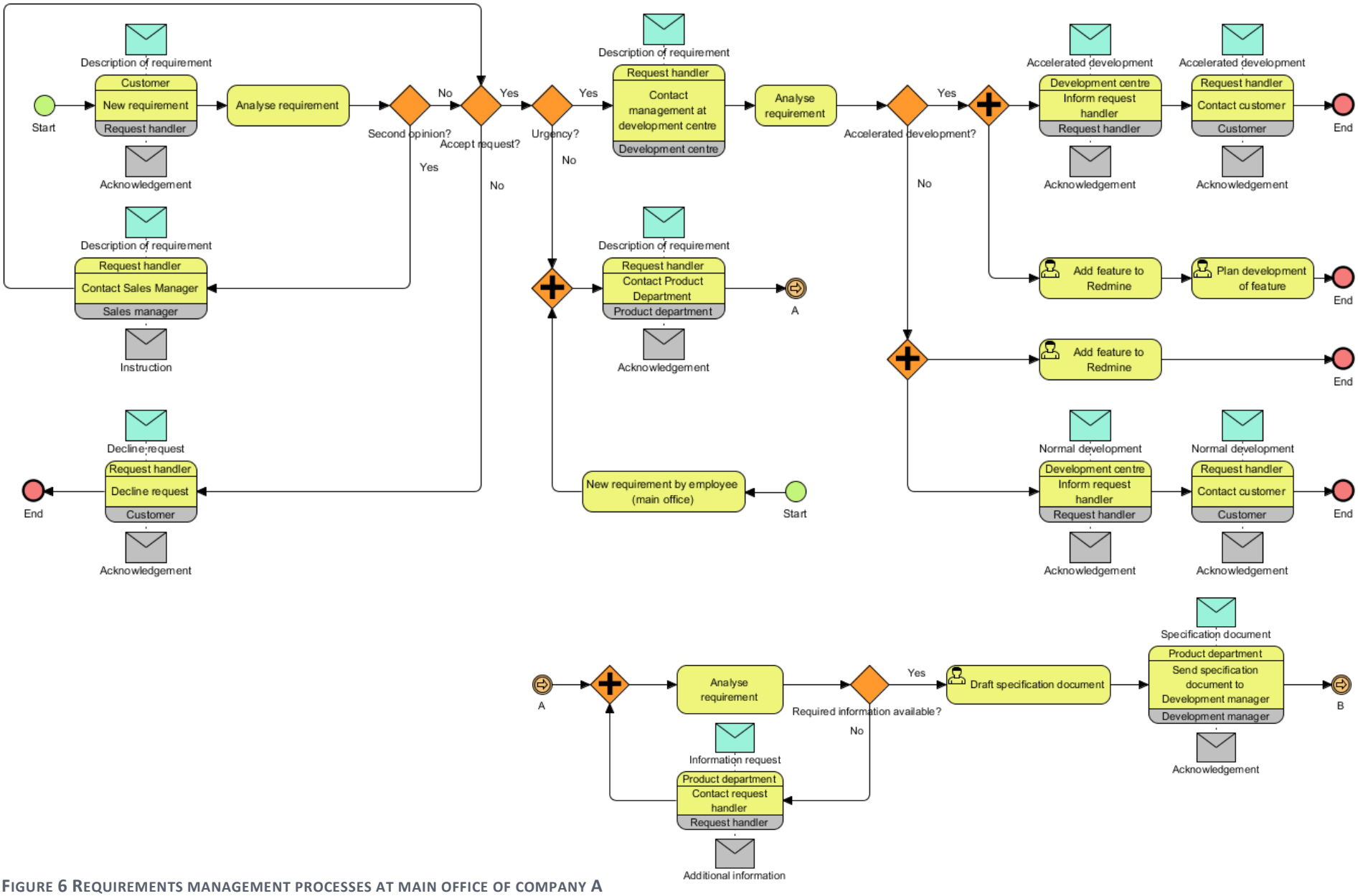


FIGURE 6 REQUIREMENTS MANAGEMENT PROCESSES AT MAIN OFFICE OF COMPANY A

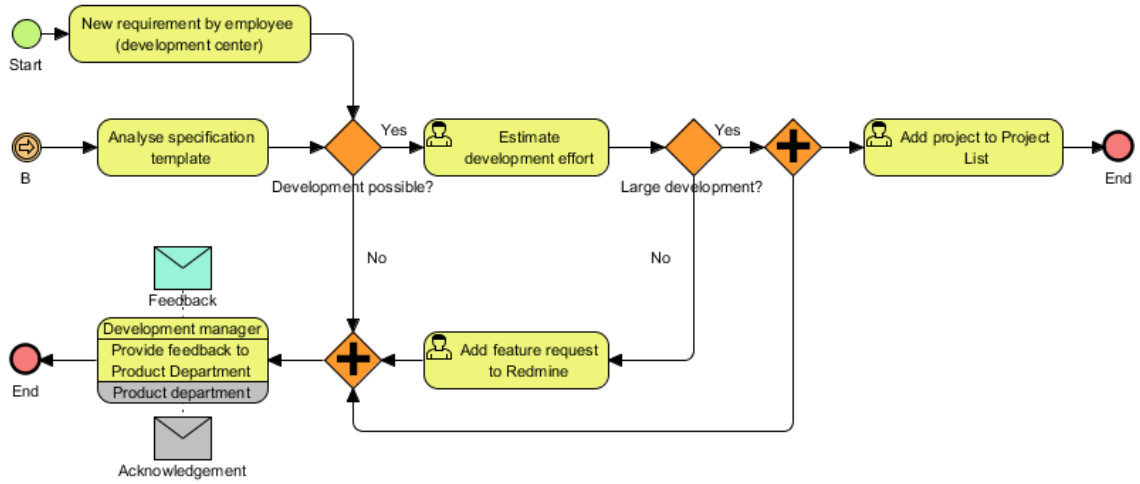


FIGURE 7 REQUIREMENTS MANAGEMENT PROCESS AT DEVELOPMENT CENTRE OF COMPANY A

6.1.2. Requirements management process maturity at company A

Looking at the capability descriptions by Bekkers et al. (2010b), see Appendix B: Maturity Matrix Capabilities, and the process models given above the following conclusions can be drawn:

- Company A gathers and registers requirements in a central database, thereby achieving capabilities A and B for requirements gathering. They don't use an automated requirements gathering systems, so capability C is not achieved. Internal stakeholders also produce new requirements, capability D is thus also achieved. Capabilities E and F are also not achieved.
- New requirements are validated by the Product department and the Development manager, thereby achieving capability B for requirements identification. However a defined template for the specification of requirements lacks, thereby skipping capability A. Similar requirements aren't pro-actively connected, so capabilities C and D are also not achieved.
- Requirements are grouped per product segment and also detailed information about the requirements is kept. Company A thus achieves maturity capabilities A and B for requirements organisation. Requirements are not linked based on dependency, capability C is not achieved.

The maturity profile resulting from the statements above is depicted in table 7.

TABLE 7 REQUIREMENTS MANAGEMENT PROCESS MATURITY AT COMPANY A

	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 organising				**A**		**B**		C			

Using process models in retrospect

During the process maturity assessment of requirements management at company A, two occasions highlighted the benefits of using process models. In the session with the Sales manager during the second iteration it became clear that sometimes the request handlers also consult him for a second opinion. Also the sequence of steps that leads to accelerated development of a requirement was revised in iteration 2. This didn't impact the maturity profile directly. However, it indicates sometimes iterative steps are needed to reveal all details of a business process.

6.1.3. Requirements management at company B

The requirements management processes at company B show comparisons with those at company A. Normally, it starts with a customer raising a new requirement to the responsible Project leader. The Project leader will pass on the requirement to his or her regional Operations manager. Occasionally, a customer may also contact the Operations manager directly. The Operations manager will then make an initial decision to accept the requirement or not. He will also judge whether the request can be largely met with the existing products and whether the customer will be satisfied with that (partial) solution. When the requirement is accepted, the Operations manager will open a request ticket in the CRM tool Efficy. This way, the request is stored centrally and accessible to everyone. It can, for example, also be seen easily which other requests a customer has submitted.

Customers of company B can also send new requirements through a link in the web services application. This, however, is hardly used and is this option is also not promoted by Project leaders and the Product management. They prefer requirements management as described above. The automated option through the web services application is thus not included in the process models.

Approximately, once a month all the Operations managers gather at the main office of company B. At this meeting all the request tickets that have been entered in the CRM system since the last meeting are discussed. Next to the Operations managers, also the VP Product management, the Product managers and a few employees from R&D are present. By actively discussing each request between the regional representatives and the product management, a complete picture about the requirement emerges. The conclusion of the discussion is to accept the requirement or not. A rough estimate is that 90% of the requests that are discussed during this meeting are accepted. The goal of this meeting is thus not to strongly reduce the amount of requirements that enter the product management system. It rather is the intention to get as much information from different points of view of all the regions as possible. After the request has been accepted, a so called user story will be added to the product management software tool VersionOne. User stories in VersionOne are grouped per product type and later, when planned, also by product version number.

As within company A, sometimes a requirement is more urgent than normal. In that case, an Operations manager may decide to contact the VP Product Management directly. This is often out of a commercial opportunity or when the Operations manager feels the request can really add a lot of value to the product. When the VP Product Management agrees, the request meeting will be bypassed and a user story is created in VersionOne. If the VP Product Management doesn't agree, a request ticket will be added in Efficy since it could be a valuable requirement to add to the product despite it's not worth accelerated development.

The complete process described above in depicted in figure 8.

New requirements can also originate internally. Firstly it can come from the Strategic Product Board. This Board chairs the CEO, the VPs of R&D, Product Management, Marketing, and Strategic Alliances, and the Customer Service manager. The board gathers every two weeks to discuss on-going matters in the market and subsequently the strategy and future directions of company B. They decide which theme the next major release will get. They can also impose strategic requirements resulting from perceived market developments. These will be entered directly into a user story in VersionOne.

Other internal requirements originate mostly from the Customer Service and Quality Assurance (QA) departments. Small quality requirements are handled directly between Customer Service or QA and

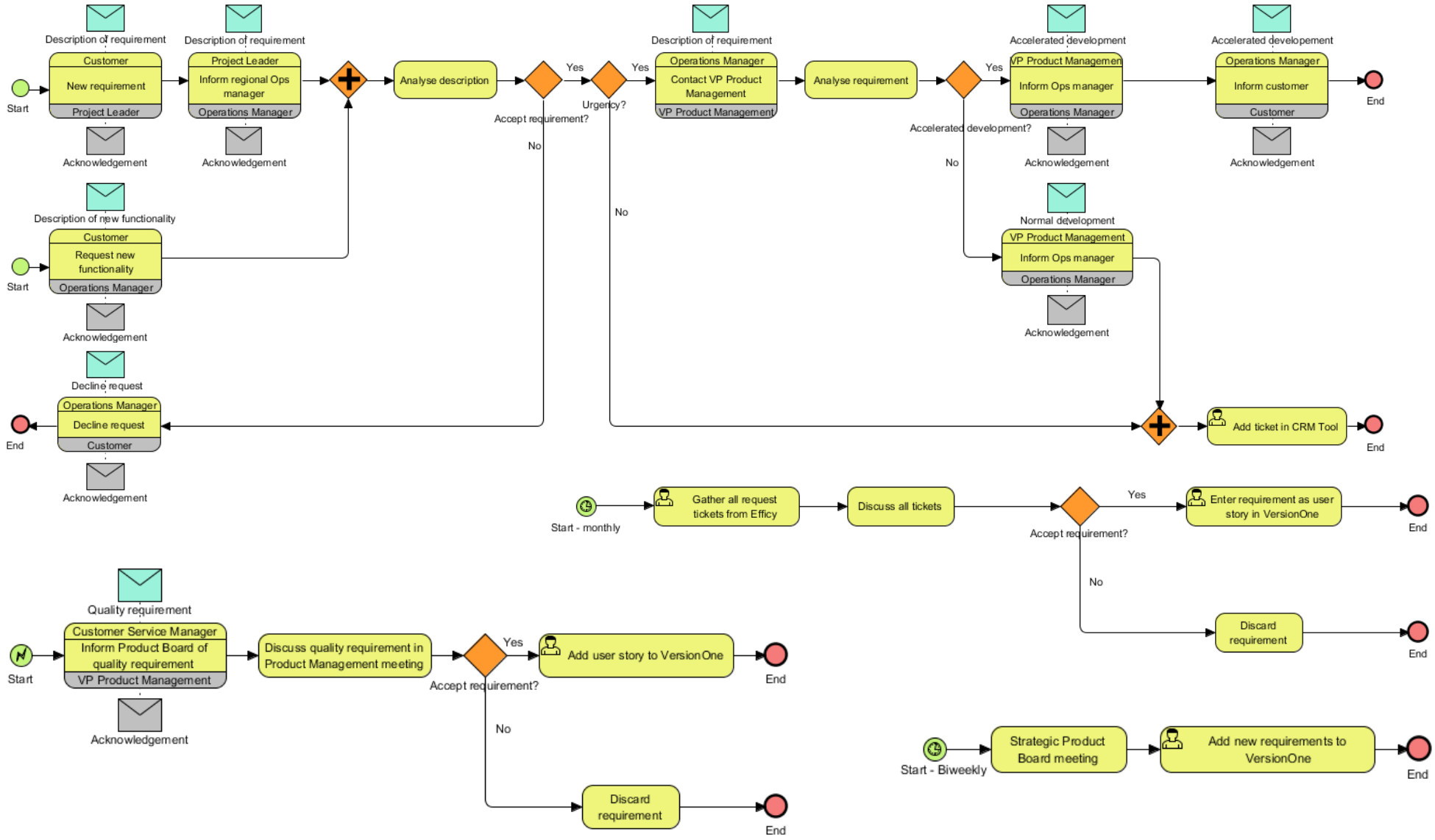


FIGURE 8 REQUIREMENTS MANAGEMENT PROCESSES AT COMPANY B

R&D. Large quality requirements, however, have to be discussed with Product Management. The Customer Service manager will discuss these requirements in either a Strategic Product Board meeting or a Product Management meeting. In case the requirement is accepted a user story will be added to the backlog in VersionOne. See figure 8 for the process diagrams.

6.1.4. Requirements management process maturity at company B

From the process models the following can be derived:

- Company B gathers and registers requirements in a central database, their CRM tool, thereby achieving capabilities A and B for requirements gathering. Although implemented in the web services application, company B doesn't really use an automated requirements gathering system at the moment, so capability C is not achieved. Internal stakeholders also produce new requirements, capability D is achieved. Capabilities E and F are also not achieved.
- New requirements are validated in the request meeting, thereby achieving capability B for requirements identification. However a defined template for the specification of requirements lacks, thereby skipping capability A. Similar requirements aren't pro-actively connected, so capabilities C and D are also not achieved.
- Requirements are grouped per product segment and also detailed information about the requirement is kept. Company B thus achieves capabilities A and B for requirements organisation. Requirements are not linked based on dependency, capability C is not achieved.

The resulting maturity profile is depicted in table 8. This shows that although very differently implemented, company B and company A have reached the same maturity profile.

TABLE 8 REQUIREMENTS MANAGEMENT PROCESS MATURITY AT COMPANY B

	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 organising				**A**		**B**		C			

Using process models in retrospect

The use of process models showed a clear benefit in the maturity rating of the requirements management process at company B. During the different iterations, it became clear that the automated requirements gathering system is hardly used in practice and also not favoured by the employees of company B. It was therefore decided to exclude it from the process models and as a result this capability is not achieved in the section above. If, in a comparable situation, a product manager would have had to answer the question 'Are incoming requirements gathered automatically in your organisation?' he might have answered yes since strictly speaking the possibility is present. This example shows that process modelling prevents the maturity rating to be influenced by the opportunistic behaviour of one person. A maturity rating based on process models may thus be more reliable than one based on data gathered through a questionnaire.

6.2. Evaluation of the requirements management business function

The sections above show that it is possible to capture all the activities of requirements management at both companies with the capabilities specified in the Maturity Matrix. In section 3.2.1 some concerns were raised about the grouping of activities in the Matrix that diverges from literature and

the inclusion of the requirements organising focus area. The process models show that the four separate requirements engineering tasks as described in section 3.2.1 are very intertwined in practice. The difficulty to discuss them separately in practice is likely to have resulted in combining the tasks of requirements elicitation and documentation into the focus area requirements gathering and combining requirements analysis and validation into the focus area requirements identification. Also, the requirements organising focus areas which is not fully represented in academic literature is relevant to requirements management in practice. The design decision by Van de Weerd et al. (2006) to depart from task categorisation as provided in requirements engineering literature thus aids the usability of the model in practice. Since the descriptions in the model still adequately capture the descriptions as available in literature no comments can be given on the structure of the requirements management business function and its focus areas.

As said before, the process models show that the actual implementation of a capability can differ largely per company. This can make it hard sometimes to decide whether a capability is implemented or not and as a result which maturity level is achieved. Another thing that stands out is the fact that both companies have skipped the same capabilities. In the requirements gathering focus area, the capability of automated requirements gathering system isn't implemented by both company A and company B. Both companies also haven't implemented the use of a template to rewrite market requirements to product requirements in the requirements identification focus area. This may be related to the specific industrial sector both companies are active in or it can be that these capabilities come less naturally than capabilities connected to higher maturity levels. This would imply the ordering by Bekkers et al. (2010b) should be revised. However, the results of only two case studies don't give enough cause to justify reordering of the capabilities in the Maturity Matrix. It is therefore recommended that future research looks into the best practice order provided in the Maturity Matrix.

Now that the evaluation of the requirements management business function has been completed, the following section will turn to the practical evaluation of the release planning business function.

6.3. Release planning

Release planning consists of the SPM capabilities needed to successfully create and launch a release. It comprises the following focus areas (Bekkers et al., 2010b):

- **Requirements prioritisation**; identified and organised requirements are prioritised
- **Release definition**; requirements that will be implemented in the next release are selected based on the prioritisation they received in the preceding process
- **Release definition validation**; focuses on the validation of the release definition by internal parties and is performed before the release is built by the development department
- **Scope change management**; handles the different kinds of scope changes that can occur during the development of a release.
- **Launch preparation**; addresses issues ranging from communication, to documentation, training, and the preparations for the implementation of the release itself to prepare the internal and external stakeholders for the launch of the new release

The focus areas in the release planning business function turned out to be less interwoven than those of requirements management as witnessed before. So although they could be discussed separately, the following sections discuss all the focus areas of release planning together. This is done so that a complete picture of the release planning business function emerges.

6.3.1. Release planning at company A

As mentioned before, company A had activities in Europe, USA, China and Brazil. However, most of the product development was, and still is, done in a separate central development centre. As part of the acquisition, the development centre is now property of company B. To support the remainder of company A outside of Europe, a Joint Development Agreement has been made between company A and B. The agreement contains a list of projects that have to be completed a year after the acquisition. After the Joint Development Agreement expires, a new structure to prioritise projects for company A needs to be put in place. It will be kept separate from the release planning of company B and its form is still to be defined. For the purposes of this project, the situation before the acquisition will be considered.

Before the acquisition, company A used an open-ended planning approach concerning the release scheme. Essentially, this means company A didn't keep to a fixed release cycle, i.e. every 6 months. The project list, prioritised by the Product Committee, formed the backbone of the release planning. The projects thus determined, for the largest part, the allocation of development resources and also the timing of the release of a new version. Although open-ended, the goal was to release a new version at least once a year to keep the product up to date/competitive and customers satisfied. Large releases like this required a combined release of new software for the on-board computers, website version and updates of the databases and server adapters. The scope of these releases usually was the top 4 projects. If work on these projects had progressed up to a certain level all on-going development was reviewed and an approximate release date was set. Based on the progress of development it was decided which other functionalities or features would also be included.

Smaller sub releases, maintenance releases, happened on a much more frequent basis. A sub release mainly contained minor bug fixes or small changes in functionality. Work on bugs and feature development is done parallel to work on the projects. The work is not scheduled precisely, but usually development is grouped per module. If, for example, work on a project requires changing the code of the messaging module, bugs and features related to that module will be done as well since that part of the code is already being revised. Of course, bug fixes and development of features also happened independently of the projects if it was important enough.

Prioritisation of projects

Previous to the acquisition, the Product Committee Board of company A would gather once every 6 months to discuss on-going projects and plan projects for roughly the next 12 months. The board chaired the company CEO, CTO and regional representatives.

In these Product Committee Board meetings, the regional representatives would provide their list of the most important requirements for their region's customers. Through these lists, the regional representatives tried to affect the future development of the products in their own favour. This makes sense, since the geologically dispersed offices are responsible for selling the solution in their local markets, where the needs can differ a lot from the needs of customers in other parts of the world. The lists of local needs were compiled by regional representatives in collaboration with their business unit managers. The needs usually concerned market pull requirements, i.e. requirements of interest for most of the customers in the region, however for an important enough customer, customer specific projects were considered as well.

Next to each representative presenting his own list of regional projects and their priorities, the CTO would put in projects originating from an internal need. This can for instance involve requirements to keep the system architecture healthy. After all the projects had been unveiled the Board would prioritise all the projects on a global level, based on their urgency and their commercial opportunity.

Subsequently, the prioritised list was handed over to the Development manager at the development centre, who would estimate the development effort. Based on this, the list could change if the estimated development effort was way higher than the Board had expected, altering the commercial (cost-value) opportunity.

The process described above is depicted in figure 9 below.

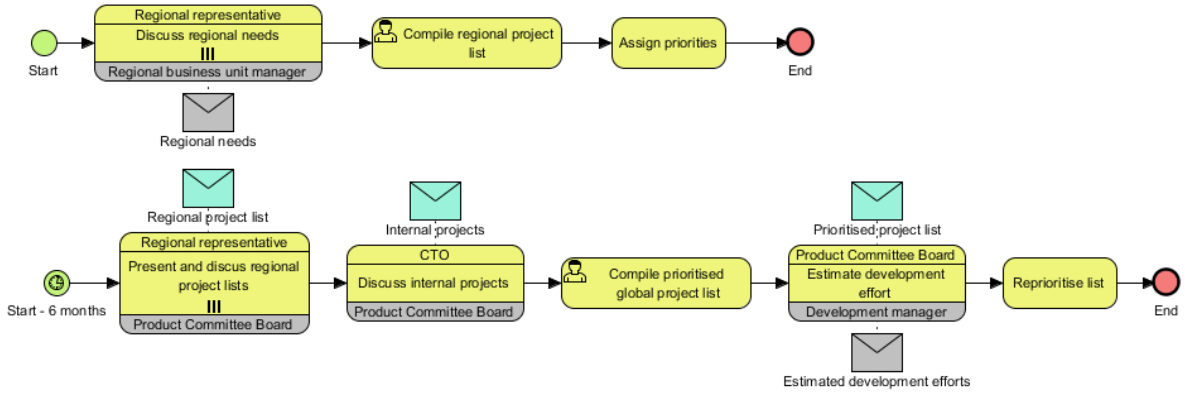


FIGURE 9 PRIORITISATION OF PROJECTS AT COMPANY A BEFORE THE ACQUISITION

Prioritisation of features

Smaller requirements, features, are still developed outside the agreements in the Joint Development Agreement that was mentioned before. The list of features in Redmine is reviewed weekly by the Development manager and CTO. During this weekly meeting new features are reviewed and the CTO will assign them a priority. A grouping based prioritisation method is used with the categories being high, normal and low. The requirement can also be discarded and removed from the feature list in Redmine.

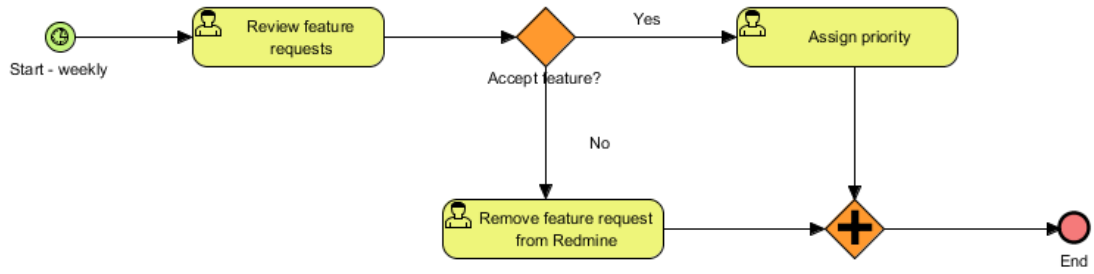


FIGURE 10 PRIORITISATION OF FEATURES AT COMPANY A

When it has been decided that a new version will be released, a release candidate version of the on-board computer software is built. This will be tested internally by the development centre and the Product department at the main office to see whether everything functions as was intended. When internal tests have been completed successfully, a pilot is carried out including a small number of trucks at one customer. The pilot will continue until the CTO, Product analyst and Support manager are confident about the quality of the new product. It will be then officially be released. Upon the release of the new software a 'release document' will be spread amongst the employees. This

document is prepared by the Product department and explains the largest and most important changes in the product.

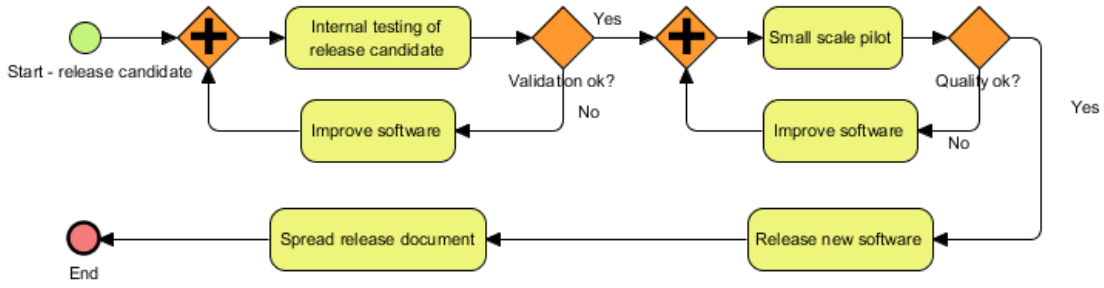


FIGURE 11 BUILD VALIDATION & RELEASE PREPARATION AT COMPANY A

6.3.2. Release planning process maturity at company A

The process models and descriptions above show that:

- Within company A, a grouping based prioritisation methodology is used. So although this is a very coarse method, capability B of requirements prioritisation is achieved. It was also indicated that the development cost-benefit criteria is very important for prioritisation, thus capability D is also achieved. Capabilities A and C are skipped since not all internal stakeholders provide input on the priorities and also external stakeholders are not part of the prioritisation process. Capability E is also not achieved.
- Company A used an open-ended release planning technique. It therefore can't be truly said a release definition is formed. None of the capabilities of release definition are achieved.
- Since there is no release definition company A also doesn't achieve any capability for release definition validation as well.
- At company A, no formal scope change management is in place. When development on projects of features turns out to be underestimated, it can be chosen to simply not release yet or exclude it from the new version.
- New products are first tested internally and externally, thus achieving capability A and B for build validation. Capability C is, certification, is not achieved.
- Internal stakeholders are informed about the new release by means of the release document, capability A in launch preparation is thus achieved. The release decision is an official 'go'. It is however not based on formal quality rules, capability B is thus not achieved.

The release planning process maturity at company A, as follows from the above, is depicted in table 9.

TABLE 9 RELEASE PLANNING PROCESS MATURITY AT COMPANY A

	0	1	2	3	4	5	6	7	8	9	10
Release planning											
Requirements prioritisation			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

Using process models in retrospect

The maturity assessment of the release planning process at company A didn't strongly benefit from using process models as data collection instrument. Although the models were slightly adjusted in

the second iteration, this would not have changed the resulting maturity rating. This indicates that using process models is not always needed to get the best results in a maturity assessment. Future research should investigate whether distinct situations can be identified where a maturity assessment benefits from using process models instead of a questionnaire.

6.3.3. Release planning at company B

Company B aims at a release cycle of approximately 6 months, varying depending on the amount of development that needs to be done. About three months before the release of the current development version, preparation for the next development version starts. It begins with the VP Product Management exporting all the user stories in the backlog of VersionOne into MS Excel. The user stories are then rated on 9 criteria to find what are called the 'golden features'. A user story can score 0, 20, 50 or 100 points on each criterion. When all user stories are scored, the scores are summed up and the user stories are ranked accordingly, those with the highest scores having the highest priority.

The golden feature criteria are:

- **Olympic minimum**; do all our competitors have this?
- **Quality**; does it increase the quality of the product?
- **Complexity**; does it decrease the complexity of the product?
- **Revenue**; does it close a deal or can we charge extra monthly costs?
- **Cost decrease**; does it decrease direct costs of company B, i.e. cheaper hardware?
- **Cost decrease customer**; does it decrease the costs for the customers?
- **Customer efficiency**; does it increase the efficiency of the customers' operations?
- **Regulatory compliance**; does it help the customer to comply with regulations?
- **Customer satisfaction**; does it increase the satisfaction of our customers' customers?

Next, the list of user stories is sent to a group of employees whose experience and knowledge is valued as input in the prioritising process. The group contains a mix of Operations and Sales managers from different regions, Product managers and others who have intimate knowledge of the market en customer demands. They are asked to compile a top 25 of user stories, number 1 gets 25 point, number 2 gets 24 points, and so on. These points are added to the scores given earlier by the VP Product Management. The voters are told in advance what the next release theme will be and they are asked to take this into account.

The top 50 user stories are now selected and put on the short list for the next version. In addition the VP Product Management will add user stories that didn't make in the top 50, but have significant added value from a strategic or theme related perspective.

Now, user stories from the shortlist are sent to responsible Product managers. They are asked to estimate an approximate development time for the user stories. The results are compared to the expectations of the Product Management. In case of large deviations a user story may become less interesting and be removed from the shortlist. The VP Product Management will now present the remaining user stories to the Strategic Product Board as a release proposal. The board will discuss the proposal and final adjustments are made in case necessary. Lastly, the release proposal is finalised and the states of the user stories in VersionOne are changed from 'unplanned' to the version they are planned for.

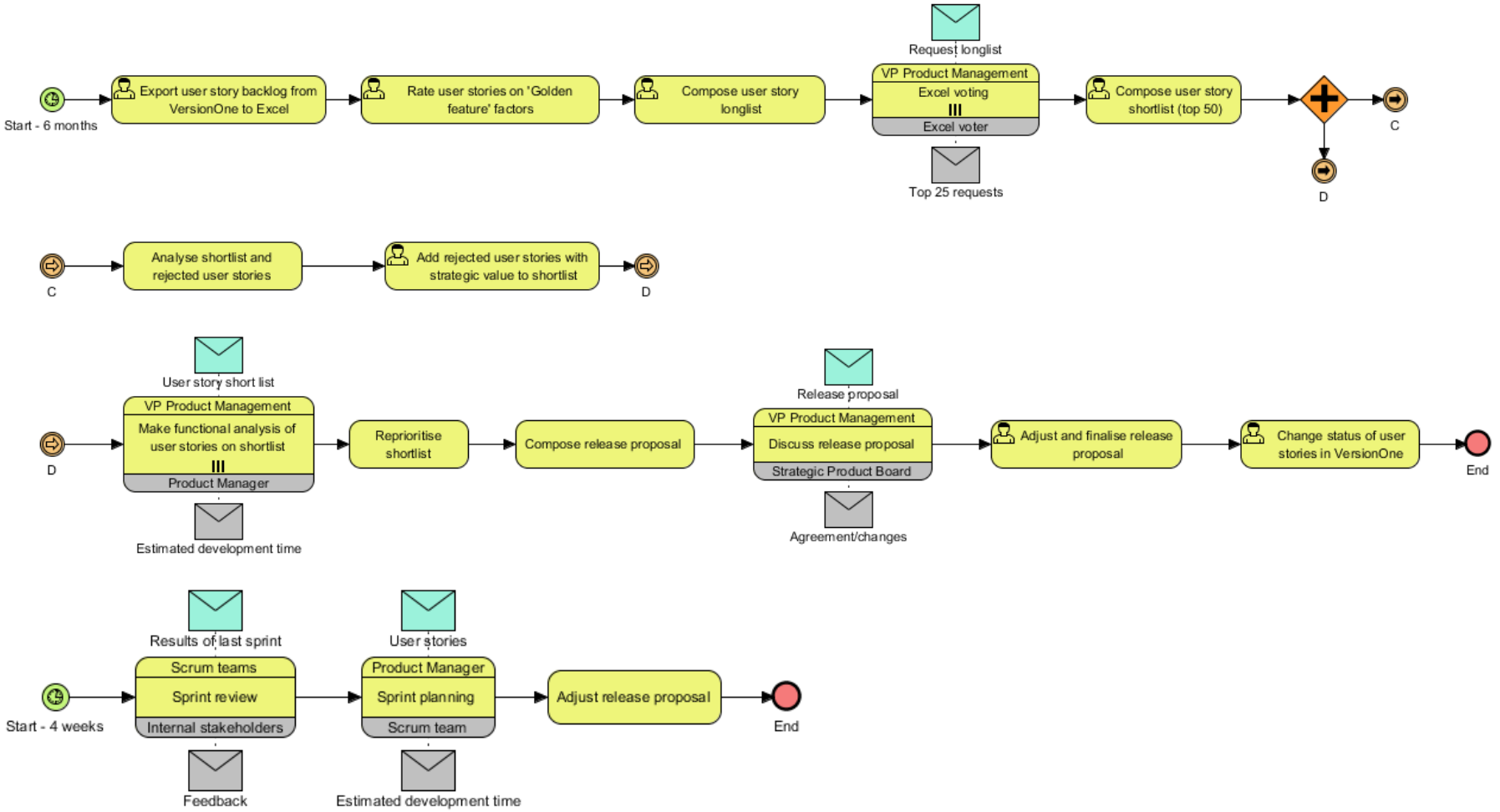


FIGURE 12 RELEASE PLANNING PROCESS AT COMPANY B

Company B has adopted the SCRUM methodology with sprints of 4 weeks. After every sprint, progress of development is discussed in the sprint review meeting. All the VPs and development teams of the different product groups are present here. In these review sessions completed features are presented to all internal stakeholders to validate it was built as intended. Subsequently, planning sessions for the next sprint are held per development team. Based on the results of both sprint sessions the overall planning, the release proposal, is evaluated and adjusted accordingly. The process described above can be seen in figure 12.

After development has finished a new feature enters what is called 'the release clock'. Company B has developed this tool to define the release preparation process. The clock and a summary table of all the checks can be found in Appendix F: Release clock at company B. The table shows that extensive preparation by multiple departments precedes the launch of a new product. The release preparation process is not included in figure 12 because it is elaborately captured by the release clock.

6.3.4. Release planning process maturity at company B

From the process models above the following is derived:

- Considering requirements prioritisation, company B involves the internal stakeholders and uses a formal prioritisation method, the golden features and voting process, and also makes a cost-revenue consideration, thus achieving capabilities A, B and D. They don't involve customers in the process, thus skipping capability C.
- The release proposal compiled by the VP Product Management fits the descriptions of capability A and B of release definition. Capabilities C, D and E are not achieved.
- The release definition has to be approved by the Strategic Product Board thus achieving capabilities A and B of release definition validation. Capability C is not achieved because the release definition is not communicated to internal stakeholders outside the Strategic Product Board and product management. Capability D is also not achieved.
- Having implemented the SCRUM methodology, company B meets capabilities A and B of scope change management. Capabilities C and D are not achieved.
- The release clock at company B defines that new features are first tested internally and next in a field test, thereby achieving levels A and B of build validation.
- The release clock also defines a rigorous launch preparation process which includes all the capabilities of launch preparation.

The resulting release planning process maturity profile at company B is depicted in table 10.

TABLE 10 RELEASE PLANNING PROCESS MATURITY AT COMPANY B

	0	1	2	3	4	5	6	7	8	9	10
Release planning											
Requirements prioritisation			**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**

Using process models in retrospect

The maturity assessment of the release planning processes at company B pointed out a clear disadvantage of using process models as assessment instrument. The build validation and launch preparation processes at company B are very elaborate and would have required a lot of effort to capture in a process model. Given the scope of this project it was therefore decided that the documentation on these processes, the release clock, sufficed to base the maturity rating on. In a normal maturity assessment the extra effort could have been made. It doesn't make it impossible to use process models in the maturity assessment. However, the effort needed to create process models is a concern that has to be kept in mind.

6.4. Evaluation of release planning business function

Using the capabilities defined in release planning business function of the Maturity Matrix, it was very well possible to rate the release planning process maturity of company A and company B. Where they scored equally on requirements management, the process maturity for release planning differs a lot. A remarkable similarity is that both companies have not implemented capability C of requirements prioritisation. This capability concerns the direct involvement of customers. Although, both companies are very oriented on customer input regarding requirements gathering, customers are not involved in the requirements prioritisation process. The requirement priority from a customer's point of view is incorporated in the process by different routing for high priority requirements as described in the requirements management business function section above. Customers are however not represented in the release planning process. This may indicate that capability D, making a cost revenue consideration, comes more naturally to companies than incorporating customers directly in their release planning process. Further research is warranted regarding the ordering of these capabilities.

Another remark on the release planning part of the maturity matrix considers the prioritisation methodology. The description of capability B specifies that a structured prioritisation technique is used, like for example MoSCoW. Capability D in turn specifies that a cost revenue consideration is made. Earlier, in section 2.1.1, this paper showed that scientific literature considers both MoSCoW and cost-revenue considerations to be a prioritisation technique though be it of different complexity. From a literature point of view, cost revenue consideration could thus also be classified as achieving capability B. The capabilities in the requirements prioritisation focus area thus seem to try to capture both an increase in maturity of actors involved in this process as well as maturation on the method used for prioritisation. The current descriptions of the capabilities in this focus area may lead to confusion, since 'using a prioritisation methodology' can differ a lot depending on the method used. It is advised that two considerations be made:

1) If the maturation of involved actors and prioritisation method used are highly correlated, thus making it necessary to capture both in one focus area, the description of at least capability B should be changed. 'Using a structured prioritisation technique' is too general to capture the different complexity of the techniques that are available.
2) It might be worth splitting the requirements prioritisation focus area into two focus areas. One that describes the maturation of involved actors and one that describes the maturation of the prioritisation techniques that is used. This could offer a better guidance for improvement since these are two very distinct topics.

Finally, a remark made in section 3.2.1 has to be revisited. It was remarked there that, from a literature point of view, it seemed odd to include the focus areas scope change management, build

validation and release preparation in the release planning business function. The investigation of the processes at company A and company B partially endorse this view. Within company A no distinct product manager function is present as defined by Bekkers et al. (2010b). Because of the smaller size of the company the responsibilities of a product manager lay with different persons. However, the release planning activities specified in the Maturity Matrix are performed by a defined group of people: the CTO, the Development manager and the Product department. This confirms that for smaller companies, in practice, it makes sense to include scope change management, build validation and release preparation in the release planning business function.

At company B there is a separate product manager function, that of the VP Product Management. The process models show that, in line with the Maturity Matrix, scope change management is indeed the responsibility of the VP Product Management. The descriptions of the build validation capabilities and the processes found at company B however raise some ambiguity. The description of capability A[1] indeed concerns validation, i.e. did we build the right thing. The description of capabilities C[2] however concerns verification, i.e. did we build it right. And the description of capability B[3] can be interpreted to serve both validation and verification. Setting up a pilot can indeed serve to let an external party verify that the right thing was built. At company B, and also at company A, however a pilot serves to test how the new software behaves in practice, thus verifying whether it was built right. In addition, at company B setting up a pilot is not the responsibility of Product management, but falls under the tasks of the QA department which is led by the VP R&D. Furthermore, the release clock at company B defines that multiple departments play a role in the release preparation, but Product management is not one of them.

The fact that literature suggests that scope change management, build validation and release preparation aren't an integral part of release planning and the processes witnessed at the two hosting companies of this project result in the following recommendations:

1) H1, H2 and H3 are confirmed. The application of the Maturity Matrix at the hosting companies has shown that scope change management, build validation and launch preparation are part of the complete product development process. However, they differ too much on a conceptual basis with the other focus areas in the release planning business function, i.e. activities leading up to development and activities during/after development. In addition, the process descriptions above show that they are separated in practice as well. At company B, launch preparation is not even part of the Product management responsibilities. It can thus lead to confusion to call all these activities a part of 'release planning'. Scope change management, build validation and launch preparation should thus be removed from the release planning business function in the Maturity Matrix.
2) The descriptions of the capabilities in the build validation focus area are ambiguous. They capture both validation and verification, which are two distinct processes. As seen within company B, in practice these processes are indeed split over different departments. The descriptions of these capabilities are therefore advised to be reconsidered to indicate this difference.

[1] Internal stakeholders (consultants, etc.) perform a functional validation of the build release to verify that it meets the expected outcome.
[2] Certification by an independent external party is acquired for the release.
[3] The build is validated by external parties (customers, partners) to verify the builds quality (e.g. by settings up a pilot.

The sections above have shown the practical experiences of using the Maturity Matrix to assess the requirements management and release planning process maturity in the two hosting companies of this project. The use of the Maturity Matrix in practice has resulted in a number of recommendations for changes to the Matrix. Also, the process models proved to be of value to determine the maturity profile. These findings are summarised in the next section.

6.5. Using process models in retrospect

The maturity assessments at companies A and B have showed a number of advantages and disadvantages of using process models as assessment instrument. A few occasions showed that the reliability of a maturity rating increases by using process models. This is mainly achieved by the increased ability to communicate about processes and the enhanced understanding of processes that process models provide. The automated requirements gathering system at company B is the best example of how process models make for a more reliable source to base maturity rating on. On the other hand, also a clear disadvantage of process models was encountered. Creating process models requires significantly more effort than completing a questionnaire. The importance of this extra effort is stressed in a commercial environment where 'time is money'. So although the findings of this project indicate that process models have added value in a maturity assessment, future research should investigate the trade-off between on the one hand the increased reliability and on the other hand the extra effort needed of using process models in a maturity assessment. H4 is thus not fully confirmed.

7. Process redesign

The recommendations made in the previous sections put aside, the Maturity Matrix was used to define an improvement proposal for the processes at company A and company B. Besides just improving the processes based on the recommendations coming from the Maturity Matrix, the proposal will also consider issues that came up in the process of constructing the process models. The following sections will discuss the issues found, followed by the improvements based on the Maturity Matrix. The issues and proposal for company A will be discussed first, followed by those of company B. For company A the change proposals have also been captured in process models of the new situation. This was done to show that the use of process models in a maturity assessment can be extended beyond just using them for the maturity rating. As section 4.1 showed, process models can also be of help in the process improvement phase. Because of space considerations process models of improvement proposals haven't been included for company B.

7.1. Issues at company A

Perhaps, the most notable issue at company A was mentioned by the sales personnel and the implementation engineers. They indicated that they have difficulties with informing the customers of the release dates of their requested features or changes that may solve their problem. This can be related back to the fact that an open-ended planning is used to base the release schedule on. Once development has progressed far enough, a release date is formally set and communicated. At that moment, it is fairly clear when the new software will be available. However, in the beginning of the development of a new version it is unclear how long it will take. At that moment a release date is estimated and communicated when asked for. However, in the experience of the employees who have to communicate this to customers, this estimate is highly inaccurate and most of the time too optimistic, i.e. the release date will most likely be later than estimated.

A second issue that was brought up is concerning the projects' priorities. Different persons within the organisation mentioned that prior to the acquisition it was not uncommon for project priorities to change. What happened is that quite short after the Product Committee Board had set the priority of the projects on the list, these would be changed before the top ranked projects had been completed. A previously less important project would become more urgent, often because of a commercial opportunity. As a result the CEO would order for work to be seized on the on-going projects and start work on the now more urgent project. This could happen multiple times within the development of one version. This sometimes led to frustrations with the software engineers since they had to stop and start working on different projects, and could never really finish one. It probably also contributed to the issue described above where the release date was highly uncertain.

A last issue that will be discussed involves the specification of new requirements. Both the Product department as the Development manager indicated that they often have difficulties writing a proper specification because the information they get is not adequate. The request handler, and in turn the customer, are contacted in case there are additional questions, but it can still happen that the requirement remains foggy. This can lead to sketchy specifications, causing the discovery of additional requirements during development or rework when results are different than intended. This again contributes to an inaccurate estimation of development time, impacting the reliability of the estimated release date.

7.2. Company A improvement proposal

The highlighted capabilities in table 11 below are recommended to be implemented at company A. The implementation of these 6 capabilities will mature the requirements management and release planning processes at company A from level 1 to level 3.

TABLE 11 IMPROVEMENT PROPOSAL COMPANY A

	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 organising				**A**		**B**		C			
Release planning											
Requirements prioritisation			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

7.2.1. Requirements identification – capability A

Requirements identification capability A concerns the implementation of a predefined template that is used to rewrite market requirements to product requirements. In section 6.1.1 it is described that the Product department and the Development manager together perform this task. An online questionnaire was used to gather information regarding the items that should be in the template. The online questionnaire asked the Product analyst from the Product department and the Development manager how often their specifications now include the items of the Volere

requirements specification template (Robertson & Robertson, 2005). The complete list of items in the Volere template and their description as the respondents received them can be found in Appendix G: Volere requirements specification template items. The complete results of the questionnaire are depicted in Appendix H: Results of requirements specification questionnaire.

It is recommended to start with a template that includes the items that were indicated to be used most often, these can be found in table 12 . The item 'client or customer' is also included since this information is important in the prioritisation of requirement, as will be discussed in the next section. Relating back to the issues that were found during the interviews, the use of a requirements specification item template can help the Product analyst and Development manager in their task of making the specifications. By communicating the items specified on the template, the sales personnel and implementation engineers also know what information they should look for. This will likely reduce the time that is spent on retrieving all relevant information.

TABLE 12 REQUIREMENTS SPECIFICATION TEMPLATE ITEMS COMPANY A (IN ORDER OF AVERAGE SCORE)

Template items	Average score
The user's problem	4,5
Expected technological environment	4
Partner applications	4
Productization requirements	4
Adaptability requirements	4
Installation requirements	4
Access requirements	4
Goal of the requested feature	3,5
Ease of learning	3,5
Understandability requirements	3,5
Speed and latency requirements	3,5
Precision or accuracy requirement	3,5
Robustness or fault tolerance requirements	3,5
Supportability requirements	3,5
Integrity requirements	3,5
Audit requirements	3,5
Immunity requirements	3,5
Client or customer	3

7.2.2. Requirements prioritisation – capability A

Requirements prioritisation capability A involves all relevant stakeholders to indicate the requirements that should be incorporated in future releases by assigning priorities to the requirements from their point of view. To implement this, it is advised to use a voting system like is in place at company B. Using the same method increases the integration of company B and company A processes. Company B uses their 'golden feature' criteria, these have however evolved from their specific business processes and can therefore not be used identically at company A. An online questionnaire was used to analyse which criteria can be used at company A.

In the process models in sections 6.1.1 and 6.3.1 three important decision moments can be discerned where requirements are essentially prioritised. These are: the decision to accept a

requirement by the request handlers, the decision whether it is more urgent than normal, and the prioritisation of new features in Redmine by the CTO. To determine which criteria are used in these decisions, an online questionnaire was used. In the questionnaire the request handlers and the CTO were asked which of the items identified by Wohlin & Aurum (2005), see table 1, are most important in the decisions listed above. The descriptions of the items that were provided to the respondents and the complete results of the questionnaire are displayed in Appendix I: Descriptions of prioritisation items and Appendix J: Results of prioritisation item questionnaire.

It is recommended representatives from the Sales and Support department, the Product analyst, the Development manager and the CTO rank the new features in Redmine on the criteria displayed in table 13. This can continue to be done on a weekly basis and the 100 dollar game method can for example be used to do determine priorities. Redmine supports exporting of new items to Excel to facilitate the voting process. The results of this effort can then function as basis for dividing the requirements in the high/normal/low priority baskets that are used already nowadays.

TABLE 13 PRIORITISATION CRITERIA COMPANY A (IN ALPHABETICAL ORDER)

Prioritisation criteria	Request handlers' average	CTO
Competitors	4	5
Complexity	4,25	3
Delivery date	4	4
Development cost-benefit	3,75	5
Requirement's issuer	4	3
Resources/competencies	3,75	5
Stakeholder priority of requirement	3,75	5
System impact	4	4

Implementing the prioritisation process in this way will make it more transparent. This makes communication to customers easier for the sales personnel and implementation engineers. At the moment the above can only be implemented for features since projects are already committed in the joint development agreement. When this has expired a similar structure can be implemented for projects as well. Including multiple decision makers in the process will make the priorities less prone to change constantly as was an issue in the past.

7.2.3. Release definition – capability A & B

Release definition capability A concerns requirements selection for the next release considering engineering capacity constraints. Release definition capability B concerns the documentation of this selection in a standard template. Nowadays, the top 4 projects used to be selected as guideline and features and bugs were grouped and added later. However, it was indicated that the priorities of project changed regularly. Given this legacy and given the CTO's intention to remain flexible in defining a release, it's not advised to implement standard release definition processes. Instead it is advised to move away from releasing a new version of the software every time a number of projects is completed, but move towards continuous deployment.

Continuous deployment comes from the website developing domain where new code is pushed to running servers multiple times a day. In the context of company A this would mean that every time a project is finished, a new version of the software is released. This shift in 'release definition

philosophy' will likely increase the dedication to finish a project once it is started because the immediate release of it shows direct results. This isn't release definition as normally intended, but fits better with the corporate culture at company A. Moving to continuous deployment will be a large endeavour since the management of software code and version control in the field will have to be changed radically. However, it will benefit company A more than trying to force them into classic release definition practices.

7.2.4. Scope change management – capability A

Implementing scope change management capability A puts into place a formal scope change management process in which all involved stakeholders are informed. At the moment no such process is in place. The communication of scope changes could be combined with the requirements prioritisation process as discussed in section 7.2.2. Next to informing the representatives from the Sales and Support department and the Product analyst of new features, updates on the work in progress can be communicate on the same occasion. Redmine supports exporting of items that have been updated less than a given number of days ago to support this process

7.2.5. Launch preparation – capability B

Launch preparation capability B implements a formal 'go', based upon standard quality rules, that must be obtained before the launch of new software can begin. Company A could use the release clock at company B as an example to try to formalise when new software is release ready. The number of stages in the release clock and the amount of trucks required in a field test should be adapted to the corporate environment of company A. An initial version of the release clock for company A should be based upon the informal metrics that are now used to decide whether software can be released. The quality standards can be fine-tuned when some experience has been gathered.

7.2.6. Process models new situations

Three of the change proposals above have been elaborated in process models. As can be seen in figure 13 the implementation of requirements identification capability A, the use of a requirements specification template, has no profound impact on the process as it is now. The process steps currently in place can still be followed with the exception that a formal template is now used to write the specification for new requirements.

Figure 14 depicts the implementation of scope change management capability A. This is a completely new part of the process model since no scope change management system is in place at the moment. Redmine supports exporting recently changed items which allows for easy gathering of change information, assuming that items in Redmine are updated adequately.

The most profound change in the process models results from the implementation of requirements prioritisation capability A, as can be seen in figure 15. The CTO can still filter new requirements on their potential value to the company. Subsequently, the Redmine items of new features must now be exported to Excel and spread amongst the employees included in the prioritisation process. These results will have to be gathered again and processed to find the final priorities of the new features. Finally, these priorities will have to be entered in Redmine.

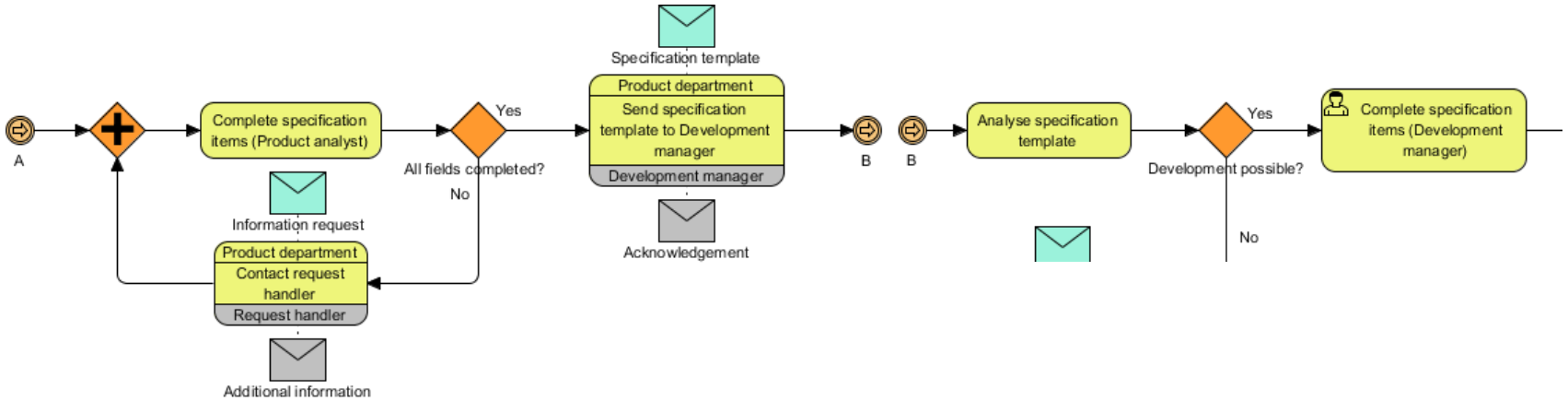


FIGURE 13 IMPLEMENTATION OF REQUIREMENTS IDENTIFICATION CAPABILITY A AT COMPANY A

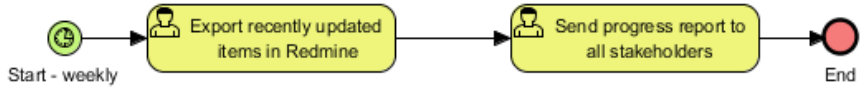


FIGURE 14 IMPLEMENTATION OF SCOPE CHANGE MANAGEMENT CAPABILITY A AT COMPANY A

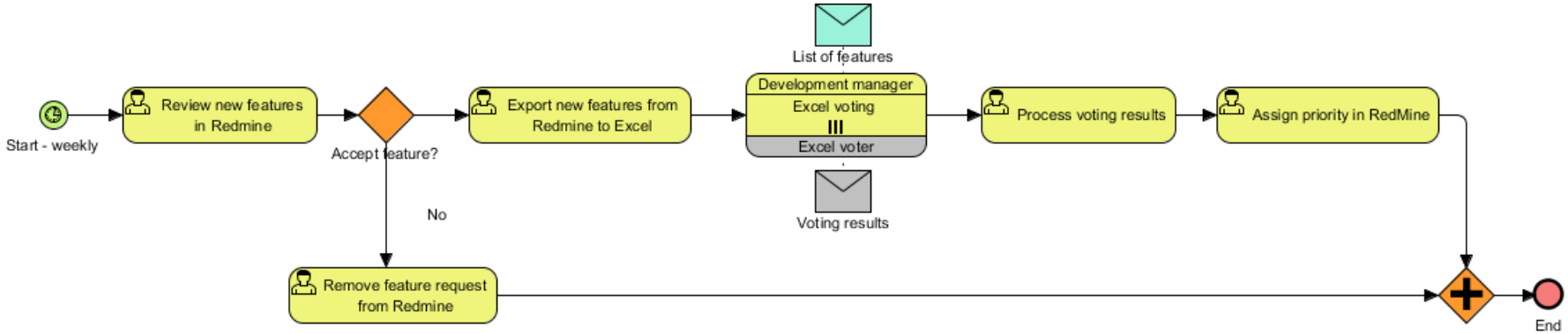


FIGURE 15 IMPLEMENTATION OF REQUIREMENTS PRIORITISATION CAPABILITY A AT COMPANY A

Figures 13 through 15 show that process models provide an excellent tool to indicate how processes will change as result of process improvement. This can support process maturity assessments and the related improvement effort, because management support is likely to increase when it can be shown how the assessment will affect the organisation. Also, process models can be used to train and mentor employees in the implementation of the new processes. As was shown in chapter 4.1, this is one of the success factors of software process improvement (SPI).

The above affirms the notion of H4, that process models are better suited as assessment instrument than is a self-assessment yes/no-questionnaire. The additional benefits of using process models when specifying the improvement proposal are significant. This leads to the conclusion that, for a complete process improvement effort, H4 is confirmed.

7.3. Issues at company B

Firstly, the Operations manager for the Netherlands at company B remarked that there is a high delay between the time of a customer request and the actual release of it in a new version of the product. This is combined with low transparency of the release planning process. The contents of the release proposal are not communicated outside the Product Management. The Operations managers only get to know about the status of a requirement when it is absolutely sure it will be in the next version. So, it is only communicated when development on a feature has been finalised or when the release date is near. Together, this makes the Operations managers unable to reliably tell the customer about when a requirement can be expected if a new release date is still far away.

Secondly, there were some remarks regarding the prioritisation process. The complete process itself receives support because the employees appreciate the fact that they can influence the priority of requirements. It is also valued that the views of different departments from all regions in Europe are incorporated. However, the respondents to the online questionnaire commented that the top 25 voting is not always clear cut. The list of requirements is often very long and only choosing 25 of them doesn't always cover the full load of expectations and/or demands. Furthermore, the list of requirements only contains the titles of the user stories as descriptions for the voters to base their votes on. It is not always fully clear what a user story is about and the voters don't have access to a more comprehensive description. This may lead to misleading results, since a high or low ranking can be based on an ill-conceived perception of a requirement.

7.4. Company B improvement proposal

In table 14 above, the capabilities that are recommended to be implemented company B are highlighted. The implementation of these 4 capabilities will mature the requirements management and release planning processes at company B from level 1 to level 5.

7.4.1. Requirements gathering – capability C

Requirements gathering capability C specifies that all incoming requirements are automatically stored in a central database. Having *all* requirements gathered through automated means seems far away for company B since the whole requirements gathering process is now based on personal contact. A first step in automatically gathering requirements is already implemented in the web services application of company B. This contains a link called 'Improve PRODUCT X' which opens and sends an email to a special mailbox. The customers can specify the request in free text in the body of the email. These messages are checked first by the Customer Service desk, to filter requests that concern existing functionalities that are unknown to the customer sending the request. Genuine

TABLE 14 IMPROVEMENT PROPOSAL COMPANY B

	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 organising				**A**		**B**		C			
Release planning											
Requirements prioritisation			**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**

new requests are forwarded to the responsible Product manager. This option is however rarely used and as a result also not included in the process models discussed earlier. The low rate of use likely results from the fact that using it is completely voluntary. Customers can still submit their requirement to the Project leaders or Operations managers. Company B should move towards gathering a larger part of the requirements through automated means since this can significantly reduce the workload and throughput time of the requirements gathering process. Based on the experiences with the current automated requirements gathering option, the functionality can be adjusted to the point where the largest part of the automatically gathered requirements is clear enough to process them in the release planning. The requirements specification template defined for requirements identification capability A in the next section below can help in this process. By letting the customers complete a number of specific items on a template, the requirements will become clearer than when using free text.

7.4.2. Requirements identification – capability A

Like company A, company B has skipped the implementation of pre-defined template to rewrite a market requirement to product requirements. In section 6.3.3 it is described that the Product managers at company B perform this task. The same online questionnaire as used at company A was used to ask them which items of the Volere template are normally included in their specifications. The complete results can be found in Appendix H: Results of requirements specification questionnaire. The highest scoring items are displayed in table 15. Political requirements were removed from this table because two Product managers marked this item not applicable. Like is advised for company A, company B should start the use of a specification template with these items. Subsequently, after some experience with the template, it can be adjusted to their experiences.

7.4.3. Requirements prioritisation – capability C

Requirements prioritisation capability C involves customers and prospects (or representatives thereof) to 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 in a delegation, a select group of customers, or in other more manageable forms. Given the large number of customers at company B, it is advised to define a group of most important/lead user customers and include them in the prioritisation process. The group should be representative for the general customer base, so there should be a mix of, for example, small and big customers. But also a

mix of different types of transport should be guaranteed since requirements can different throughout the transportation sector. It is also important that the delegation consists of so-called lead customers. This group of customers experiences needs that become general in the market place

TABLE 15 REQUIREMENTS SPECIFICATION TEMPLATE ITEMS COMPANY B (IN ORDER OF AVERAGE SCORE)

Template items	Average score
Look and feel requirements	4,7
Understandability requirements	4,7
Compliance requirements	4,5
Goal of the requested feature	4,3
Reliability and availability requirements	4,0
Access requirements	4,0
The user's problem	3,7
Functional requirements	3,7
Ease of learning	3,7
Speed and latency requirements	3,7
Scalability or extensibility requirements	3,7
Partner applications	3,7
Supportability requirements	3,7
Standards requirements	3,7

already in an earlier stage. By including lead customers in the requirements prioritisation process, company B can improve their innovative power which can help enlarge their competitive advantage. The delegation of customers can, for example, be included in the top 25 voting and be invited to take part in the Strategic Product Board meetings to align the strategic heading of company B.

7.4.4. Release definition – capability C

Release definition capability C institutes communication of the release definition to the internal stakeholders. These days, the release proposal is not communicated to internal stakeholders outside the product management department. Employees such as the sales personnel and the operations managers, only hear from features once they have been developed. This is the result of experiences in the past where development of features was committed and communicated in an early stage. Experience has learned that these commitments keep company B from being able to be flexible enough to adjust product development activities to certain events. In the past Product management thus felt committed to the development of features once it had been communicated that these were planned to be developed. This is, however, a matter of perception. Product management should start communicating about the release proposal again since this will comfort internal personnel. It should be emphasised that the development plans that are communicated are only plans and thus susceptible to change. By stressing the fact that the plans can, and likely, will change internal stakeholders will understand the release proposal is not a commitment to develop the requirements included in it. After each sprint it can be communicated what has changed in the release proposal and what is already finished.

8. Discussion

8.1. Using the Maturity Matrix to guide SPI

The sections above discussed the capabilities that should be implemented at company A and company B according to an SPI effort based on the Maturity Matrix. The choice of which capabilities

to implement first was very straightforward. Adopting the rationale that the capabilities missing in the lowest maturity level have to be implemented first, gives clear guidance on the order of implementing the capabilities. There remains, however, the question whether every company should strive for the highest maturity level possible. It may not be worthwhile for each company to implement the capabilities is the higher maturity levels. This is also why Bekkers et al. (2010a) are working on a method to determine what they call the *optimal capability profile* (OCP). As mentioned in section 4, the Maturity Matrix is part of a larger research effort into the Situational Assessment Method (SAM). It is the aim of this method to determine whether some capabilities in the Maturity Matrix should be ignored based on Situational Factors. Unfortunately, the Situational Factor Effects that are used to determine the OCP are still in an early stage of development (Bekkers & Spruit, 2010) and could thus not be used in this project. The use of Situational Factors however might have proven valuable in this project. As discussed in section 6.2, both company A and company B have skipped the same capabilities in the requirements management business function. Since these companies are active in the same industry and offer comparable products, it is highly likely that they experience the same effects of situational factors. It would have been interesting to see whether Situational Factor Effects would have disabled these capabilities or that another explanation has to be found for this curious fact that both companies skipped the same capabilities. Future research on the Situational Factor Effects is thus encouraged.

Furthermore, Bekkers et al. (2010a) defined two criteria that must be satisfied for a capability to reach the implemented status in the Maturity Matrix:

- **All capabilities must be reoccurring**; the process must be executed on a reoccurring and planned basis, and not ad hoc. If a capability is not executed on a regular, predetermined basis, then the capability is not satisfied.
- **All capabilities must be documented**; a detailed description of the processes must be described in a document for all parties involved in the capability. All parties involved in the capability must at least have access to the (part of) the process describing the actions that are required of them.

These two criteria pose a number of problems in combination with the use of a yes/no-questionnaire. The first criterion, all capabilities must be reoccurring, contrasts with answering yes or no to the question 'Is this capability implemented in your organisation?'. It can happen that a capability is executed often or most of the time, but not always. At what point does the reoccurrence of a capability justify to answer 'yes this capability is implemented'? Using process models can overcome this issue. This will be discussed in the next chapter. However, changes to the questionnaire proposed by Bekkers et al. (2010a) can also address this issue. Authors of other maturity frameworks, for example, propose to use a scale to indicate how often an activity is conducted (see for example Cater-Steel et al., 2006 and Habra et al., 2008).

The second criteria by Bekkers et al. (2010a), all capabilities must be documented, is also not feasible in practice. If the assessment in this project would have adhered to this criterion, company A would have reached none of the capabilities and also the capabilities reached by company B would have decrease dramatically. At company A, there is no documentation available that provides descriptions on how any of the processes in the Maturity Matrix must be performed. The creation of the process models however showed that there is an accepted way of performing certain tasks and that all employees are acquainted with this practice. Also at company B, there are accepted ways of performing certain activities without having this documented. The condition that a capability must

be documented before it can be said to be implemented is thus not tenable in practice. This may especially be true for companies with low process maturity.

8.2. Using process models as assessment instrument

This project investigated the use of process models to provide the information needed to fill in the Maturity Matrix. As discussed in section 4, Bekkers et al. (2010a) propose the use of a self-assessment yes/no-questionnaire to determine the currently implemented capabilities. However, section 4.1 showed this method has quite a number of disadvantages. It was also shown that, instead, process models could be used to overcome these disadvantages and that process models fulfil critical success factors for software process improvement (SPI).

A number of the disadvantages mentioned earlier for the use of a yes/no-questionnaire in a maturity assessment concern the disability of yes/no-questionnaires to provide the rich information needed to accurately portray software processes. The process modelling approach used in this project has proven to be able to capture and display software processes at a detailed enough level to perform a high quality maturity assessment. Sections 6.1.2, 6.1.4, 6.3.2 and 6.3.4 showed that the elaborate information provided by process models enables a well-considered decision on the capabilities implemented by an organisation.

Maturity assessment by means of process models might even be of higher quality than by means of a questionnaire. By separating the assessment in a modelling and analysis phase, the assessment is based purely on actual processes that are implemented in a company. Going through a list of capabilities and asking whether these are implemented, can create bias. The practitioner answering the questions projects the provided descriptions of the capabilities onto the processes in the company influencing the perspective of his analysis. When a practitioner is first asked to describe the processes in a company there is no predisposition towards descriptions in the maturity framework that is used. Subsequently, as advised in section 4.1, by letting an external assessor compare the process descriptions provided by the practitioner with the descriptions in the maturity model, all bias can be eliminated from the assessment.

Additionally, one of the disadvantages mentioned of using yes/no-questionnaires is that it contrasts with the iterative nature of collecting information on the software process. This was also witnessed in this project. Section 5.2.1 showed that multiple rounds of interviews were needed to get the process models to accurately portray the processes as implemented in both the hosting companies. An important implication of this iterative nature is that using only one moment in time, the completion of a questionnaire can overlook substantial pieces of information that are vital for a reliable maturity assessment.

The use of process models has furthermore proven to be valuable to communicate about the maturity assessment. In the discussion sessions with the company supervisors, the models functioned as a tool to explain how certain conclusions about process maturity were derived. It increased the ability of showing how the results of the assessment came about.

A clear disadvantage of using process models is the required effort to make them. The time and resources needed to complete a self-assessment yes/no-questionnaire are substantially lower than those needed for process model based assessment. It must be considered whether the advantages of using process models mentioned above are worth this difference. This is ultimately up to the company that is being assessed. Future research should look into this matter. Perhaps guidelines can

be formulated as to when a process modelling approach is preferred over the use of a questionnaire and vice versa.

9. Conclusions

This paper provides the output of a master thesis project that was conducted at two companies that provide telematics solutions for the transportation industry. The objectives of this project were to investigate an advanced method of data collection for the purposes of process maturity assessment and the evaluation of a novel maturity framework.

9.1. Managerial implications

Literature research has shown that release planning is of importance to the success of a company, however, in practice there is little guidance on developing a proper release planning process. Popular maturity frameworks like the CMMI and SPICE are too broad to specifically target the release planning process. In addition, these models require too many resources for SMEs to use them. Two novel maturity frameworks targeted at release planning were found. The Software Process Management (SPM) maturity framework by Bekkers et al. (2010a) was chosen to be used in this project.

The use of the Maturity Matrix at the two companies participating in this project turned out to be very helpful. The company supervisors were positive about results of the release planning process maturity assessment. It provided them with information about the current state of the processes and it also gave them structure to base improvement directions on. Software product managers are thus encouraged to look at the Maturity Matrix to help them guide their process improvements.

9.2. Theoretical implications

The assessment instrument proposed by Bekkers et al (2010a), a self-assessment yes/no-questionnaire, was shown to have of number of serious disadvantages. A more advanced assessment instrument thus had to be found. A literature search turned up a lead to use process models and further research confirmed process modelling could serve as advanced data collection instrument for maturity assessment purposes. The BPMN notation was selected as modelling language because of its high ontological completeness.

The use of the Maturity Matrix in combination with a process modelling approach, instead of the questionnaire proposed by Bekkers et al. (2010a), proved to be successful to complete the maturity assessments in this project. A number of advantages of using process models to base maturity assessment on have been discussed. The primary benefit of using process models is the increased reliability of the maturity rating. As section 8.2 discussed, the separation of data collection and process analysis in a maturity assessment, prohibits measurement bias towards maturity levels defined in the maturity framework that is used. Also, the use of process models was shown to have advantages in putting together the improvement proposal. By adjusting the process model created for the maturity assessment according to the proposed improvements, it can more easily be shown what the impact of the improvements is. This enhances management support and also supports employee training. A clear disadvantage of using process models is the extra effort that is needed to complete the maturity assessment. However, given the promising benefits of using process models, future research is warranted.

Also, a critical theoretical evaluation of the Maturity Matrix was conducted. This was needed because of the early stage of validation of the model. The theoretical evaluation indicated a number of minor issues that had to be kept in mind during the maturity assessments at the hosting companies. The practical use of the Maturity Matrix in this research project endorsed some of the issues and a number of redesign recommendations were made. Most important is the recommendation to consider the removal of the scope change management, build validation and release preparation focus areas from the release planning business function. These activities are very important in practice, but do not fit the theoretical concept of release planning.

The practical use of the Maturity Matrix also revealed a strong benefit of its design. The use of a focus area oriented model instead of a fixed-level approach has strong advantages for the guidance of software process improvements. By spreading the maturation of the single focus areas over a larger number of overall maturity levels, a fine-grained improvement plan can be put together. However, only very few maturity models have used this design of a focus area oriented model. Future research into the development of maturity models should investigate the advantages and disadvantages of different designs of maturity models.

9.3. Limitations and future research

Of course this research project also has a number of limitations, which lead to directions for future research.

Firstly, regarding the use of process models as data collection instrument in a process maturity assessment, a lot of work remains to be done. This project only provided qualitative evidence that using process models enhances the reliability of maturity assessment. Future research will have to show in a quantitative manner that this is indeed true.

In addition, this project investigated the use of the BPMN modelling notation to create the process models. BPMN notation was chosen because of its high ontological completeness. However, as remarked in section 4.2, a large number of different notations exist. Different characteristics of the available languages can make it less or more suitable in a maturity assessment. Future research should investigate what is required from a modelling language to be used in a maturity assessment.

A last limitation and direction for future research regards the recommendations done for the redesign of the Maturity Matrix. The work by Bekkers et al. (2011b) is still in early stages of validation. This project increased the pool of practical experiences with implementing the model to assess process maturity. A number of redesign recommendations turned up as consequence. However, this is only based on experiences within two companies within the same industry. More research should be done to validate the findings done in this project and increase the validation of the Maturity Matrix.

References

Aguilar-Savén, R. (2004). Business process modelling: Review and framework. *International Journal of Production Economics, Vol 90*, 129 - 149.

Bagnall, A., Rayward-Smith, V., & Whittley, I. (2001). The Next Release Problem. *Information and Software Technology, Vol. 43, No. 14*, 883 - 890.

Bandinelli, S., Fuggetta, A., Lavazza, L., Loi, M., & Picco, G. (1995). Modeling and Improving an Industrial Process. *IEEE Transactions on Software Engineering, Vol. 21, No. 5*, 440 - 454.

Becker, J., Knackstedt, R., & Pöppelbuß, J. (2009). Developing Maturity Models for IT Management - A Procedure Model and its Application. *Business & Information Systems Engineering, Vol. 1, No. 3*, 213 - 222.

Bekkers, W., & Spruit, M. (2010). The Situational Assessment Method put to the test: Improvements based on case studies. *2010 Fourth International Workshop on Software Product Management (IWSPM)*, (pp. 7 - 16). Sydney, NSW.

Bekkers, W., Spruit, M., van de Weerd, I., van Vliet, R., & and Mahieu, A. (2010a). A Situational Assessment Method for Software Product Management. *ECIS 2010 Proceedings. Paper 22.*

Bekkers, W., Van de Weerd, I., Spruit, M., & Brinkkemper, S. (2010b). A Framework for Process Improvement in Software Product Management. *Communications in Computer and Information Science, Vol 99*, 1 - 12.

Berander, P., & Andrews, A. (2005). Requirements Prioritization. In A. W. Aurum, *Engineering and Managing Software Requirements* (pp. 69 - 84). Berlin: Springer Verlag.

Bollinger, T., & McGowan, C. (1991). A Critical Look at Software Capability Evaluations. *IEEE Software*, 25 - 41.

Brewerton, P. (2001). *Organizational research methods.* London: Sage.

Brodman, J., & Johnson, D. (1994). What small businesses and small organizations say about the CMM. *Proceedings of the 16th International Conference on Software Engineering ICSE-16*, (pp. 331 - 340).

Carlshamre, P. (2002). Release planning in market-driven software product development: provoking an understanding. *Requirements Engineering, Vol. 7*, 139 - 151.

Cater-Steel, A., Toleman, M., & Rout, T. (2006). Process improvement for small firms: An evaluation of the RAPID assessment-based method. *Information and Software Technology, Vol. 48*, 323 - 334.

Cattaneo, F., Fuggetta, A., & Lavazza, L. (1995). An experience in process assessment. *ICSE '95 Proceedings of the 17th international conference on Software engineering.* Seattle, Washington, USA: ACM.

CMMI Product Team. (2010). *CMMI for Development, Version 1.3.* Technical report CMU/SEI-2010-TR-033. SEI, Carnegie Mellon University.

Curtis, B., Kellner, M., & Over, J. (1992). Process Modeling. *Communications of the ACM, Vol. 55, No. 9*, 75 - 90.

Dangle, K., Larsen, P., Shaw, M., & Zelkowitz, M. (2005). Software process improvement in small organizations: a case study. *IEEE Software, Vol. 22, No. 6*, 68 - 75.

De Bruijn, T., Freeze, R., Kulkarni, U., & Rosemann, M. (2005). Understanding the Main Phases of Developing a Maturity Assessment Model. *ACIS 2005 Proceedings.* Paper 109.

Dyba, T. (2005). An Empirical Investigation of the Key Factors for Success in Software Process Improvement. *IEEE Transactions On Software Engineering, Vol. 31, No. 5*, 410 - 424.

Eisenberg, I. (2011). Lead-User Research for Breakthrough Innovation. *Research-Technology Management, Vol. 54, No. 1*, 50 -58.

Fraser, P., Moultrie, J., & Gregory, M. (2002). The use of maturity models/grids as a tool in assesing product development capability. *IEMC '02. IEEE International Engineering Management Conference, 2002.* (pp. 244 - 249). IEEE.

Gillham, B. (2000). *Developing a Questionnaire.* Londen, New York: Continuum.

Gray, E., Sampaio, A., & Benediktsson, O. (2005). An Incremental Approach to Software Process Assessment and Improvement. *Software Quality Journal, Vol. 13*, 7 - 16.

Habra, N., Alexandre, S., Desharnais, J., Laporty, C., & Renault, A. (2008). Initiating software process improvement in very small enterprises: Experience with a ligth assessment tool. *Information and Software Technology, Vol. 50*, 763 - 771.

Hevner, A., March, S., Park, J., & Ram, S. (2004). Design science in information system research. *MIS Quarterly, Vol. 28, No. 1*, 75 - 105.

Karlsson, J., & Ryan, K. (1997). A cost-value approach for prioritizing requirements. *IEEE Software, Vol. 14, No. 5*, 67 -74.

Karlsson, J., Wohlin, C., & Regnell, B. (1998). An evaluation of methods for prioritising software requirements. *Information and Software Technology, Vol. 39*, 939 - 947.

Kavakli, E., & Loucopoulos, P. (2004). Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods. In J. Krogstie, T. Halpin, & K. Siau, *Information Modeling Methods and Methodologies* (pp. 102 - 124). IDEA Group.

Khokhar, M., Zeshan, K., & Aamir, J. (2010). Literature review on the software process improvement factors in the small organizations. *4th International Conference on New Trends in Information Science and Service Science (NISS)*, (pp. 592 - 598). Gyeongju.

Koomen, T., & Pol, M. (1999). *Test Process Improvement, a practical step-by-step guid to structured testing.* Boston: Addison-Wesley.

Kotonya, G., & Sommerville, I. (2002). *Requirements Engineering: Processes and Techniques.* New York: John Wiley & Sons, Inc.

Lindgren, M., Land, R., Norström, C., & Wall, A. (2008). Towards a Capability Model for the Software Release Planning Process — Based on a Multiple Industrial Case Study. In A. Jedlitschka, & O. Salo, *Product-Focused Software Process Improvement* (pp. 117 - 132). Springer Berlin/Heidelberg.

Maier, A. M., Moultrie, J., & Clarkson, P. J. (2011). Assessing Organizational Capabilities: Reviewing and Guiding the Development of Maturity Grids. *IEEE Transactions on Engineering Management*, 1 - 22.

Maurice, S., Ruhe, G., Saliu, G., & Ngo-The, A. (2006). Decision Support for Value-Based Software Release Planning. In S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, & P. Grünbacher, *Value-Based Software Engineering* (pp. 247 - 261). Springer Berlin/Heidelberg.

Ngo-The, A., & Ruhe, G. (2008). A systematic approach for solving the wicked problem of software release planning. *Soft Computing - A Fusions of Foundations, Methodologies and Applications, Vol. 12, No. 1*, 95 - 108 .

Niazi, M., Cox, K., & Verner, J. (2008). A measurement framework for assessing the maturity of requirements engineering process. *Software Quality Journal, Vol. 16*, 213 - 235.

Niazi, M., Wilson, D., & Zowghi, D. (2006). Critical success factors for software process improvement implementation: an empirical study. *Software Process: Improvement and Practice, Vol. 11, No. 2*, 193 - 211.

OMG. (2011). Business Process Model and Notation (BPMN). Object Management Group, Inc.

Paulk, M., Curtis, B., Chrissis, M., & Weber, C. (1993). Capability maturity model, version 1.1. *IEEE Software, Vol. 10, No. 4*, 18 - 27.

Pettersson, F., Ivarsson, M., Gorschek, T., & Öhman, P. (2008). A practitioner's guide to light weight software process assessment and improvement planning. *The Journal of Systems and Software, Vol. 81*, 972 - 995.

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

Pino, F., Pardo, C., García, F., & Piattini, M. (2010). Assessment methodology for software process improvement in small organizations. *Information and Software Technology, Vol. 52*, 1044 - 1061.

Rainer, A., & Hall, T. (2002). Key success factors for implementing software process improvement: a maturity-based analysis. *Journal of Systems and Software, Vol. 62, No. 2*, 71 -84.

Recker, J., Indulska, M., Rosemann, M., & Green, P. (2005). Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. *16th Australasian Conference on Information Systems.* Sydney, Australia: Australasian Chapter of the Association for Information Systems.

Regnell, B., & Brinkkemper, S. (2005). Market-driven requirements engineering for software products. In A. Aurum, & C. Wohlin, *Engineering and Managing Software Requirements* (pp. 287 - 308). Berlin Heidelberg: Springer.

Robertson, S., & Robertson, J. (2005). *Requirements-Led Project Management: Discovering David's Slingshot.* Londen: Addison-Wesley.

Saaty, T. (1980). *The Analytical Hierarchy Process.* McGraw-Hill, Inc.

Saliu, O., & Ruhe, G. (2005). Supporting Software Release Planning Decisions for Evolving Systems. *Proceedings of the 2005 29th Annual IEEE/NASA Software Engineering Workshop (SEW'05)*, (pp. 14 - 24). Greenbelt, MD.

Shaikh, A., Ahmed, A., Memon, N., & Memon, M. (2009). Strengths and Weaknesses of Maturity Driven Process Improvement Efforts. *CISIS '09. International Conference on Complex, Intelligent and Software Intensive Systems* (pp. 481 - 486). Fukoaka: IEEE Computer Society.

Sommerville, I., & Ransom, J. (2005). An Empirical Study of Industrial Requirements Process Assessment and Improvement. *ACM Transactions on Software Engineering and Methodoloy, Vol. 14, No. 1*, 85 - 117.

Stelzer, D., & Mellis, W. (1998). Success Factors of Organizational Change in Software Process Improvement. *Software Process: Improvement and Practice, Vol. 4, No. 4*, 227 - 250.

Stevens, S. (1946). On the Theory of Scales of Measurement. *Science, New Series, Vol. 103, no. 2684*, 667 - 680.

Van Aken, J., Berends, H., & Van Der Bij, H. (2007). *Problem Solving in Organizations.* New York: Cambridge University Press.

Van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support. *International Workshop on Software Product Management, 2006. IWSPM '06.* (pp. 3 - 12). Minneapolis, MN, USA: IEEE.

Van Steenbergen, M., Brinkkemper, S., & van den Berg, M. (2007). An instrument for the development of the enterprise architecture practice. *Proceedings of the 9th International Conference on Enterprise Information Systems*, (pp. 14 - 22).

Van Steenbergen, M., van den Berg, M., & Brinkkemper, S. (2009). A Balanced Approach to Developing the Enterprise Architecture Practice. In J. Filipe, J. Cordeiro, & J. Cardoso, *Lecture Notes in Business Information Processing: Enterprise Information Systems, Vol. 12* (pp. 240 - 259). Spingern Berlin / Heidelberg.

Weber, R. (1997). *Ontological Foundations of Information Systems.* Melbourne: Coopers & Lybrand.

Wiegers, K. (1999). First Things First: Prioritizing Requirements. *Software Development, Vol. 7, No. 10*, 24 -30.

Wiegers, K., & Sturzenberger, D. (2000). A Modular Software Process Mini-Assessment Method. *IEEE Software*, 62 - 69.

Wohlin, C., & Aurum, A. (2005). What is Important when Deciding to Include a Software Requirement in a Project or Release? *2005 International Symposium on Empirical Software Engineering*, (pp. 246 - 255).

Wohlin, C., & Aurum, A. (2006). Criteria for Selecting Software Requirements to Create Product Value: An Industrial Empirical Study. In S. Biffl, A. Aurum, B. Boehm, H. Erdogan, & P. Grünbacher, *Value-Based Software Engineering* (pp. 179 -200). New York: Springer Verlag.

Zubrow, D., Hayes, W., Siegel, J., & Goldenson, D. (1994). *Maturity Questionnaire.* CMU/SEI-94-SR-7 Special Report.

zur Muehlen, M., & Indulska, M. (2010). Modeling languages for business processes and busines rules: A representational analysis. *Inofrmation Systems, Vol 35*, 379 - 390.

Appendix A The SPM Competence Model

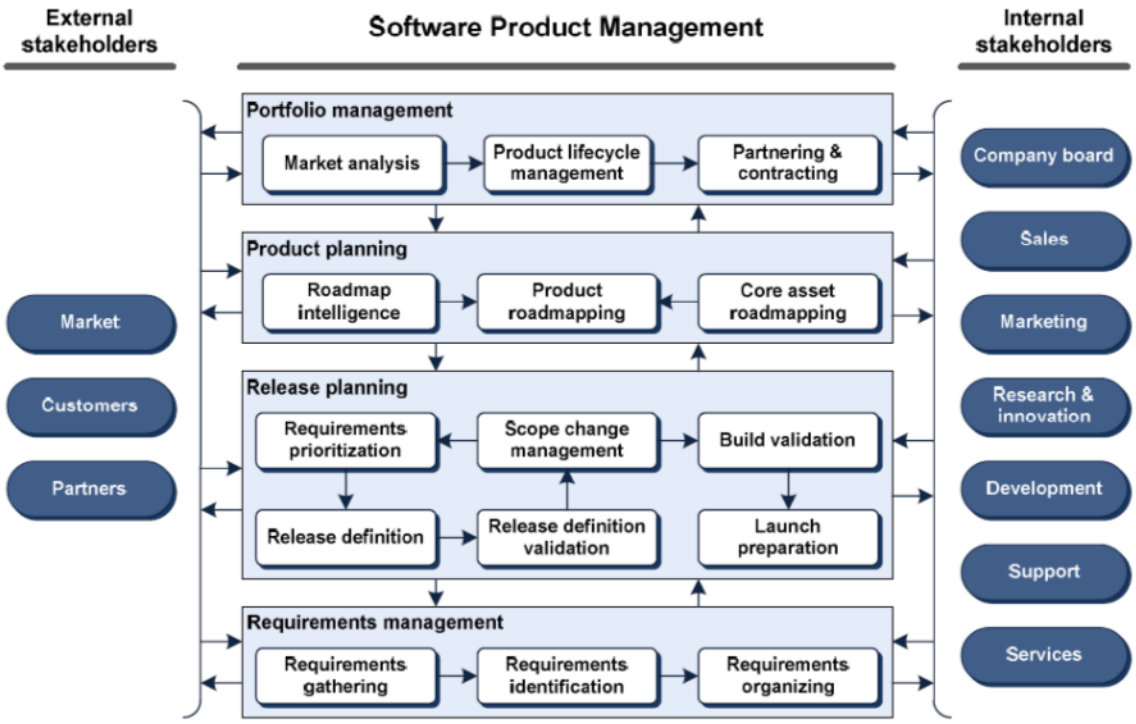


FIGURE 16 THE SOFTWARE PRODUCT MANAGEMENT COMPETENCE MODEL (BEKKERS, VAN DE WEERD, SPRUIT, & BRINKKEMPER, 2010B)

Appendix B: Maturity Matrix Capabilities

Requirements Management

Requirements management concerns the management of the requirements themselves outside of the releases.

Requirements gathering

Requirements gathering concerns the acquisition of requirements from both internal and external stakeholders

A Basic registration

Goal: Create a basis for product development.
Action: Requirements are being gathered and registered.
Prerequisite(s): -
Note(s): -

B Centralized registration

Goal: Structuring of requirements registration.
Action: All incoming requirements are stored in a central database, which is accessible to all relevant stakeholders.
Prerequisite(s): Requirements gathering A
Note(s): -

C Automation

Goal: Reduced workload / improved speed of requirements gathering process, reduced error percentage
Action: All incoming requirements are automatically stored in a central database (e.g. by means of an online helpdesk).
Prerequisite(s): Requirements gathering A
Note(s): -

D Internal stakeholder involvement

Goal: Improved product quality & increased involvement of internal stakeholders in the product management process.
Action: Requirements are gathered from all relevant internal stakeholders: support, services, development, sales & marketing, research & development (parties not present in your organization can be ignored).
Prerequisite(s): -
Note(s): -

E Customer involvement

Goal: Incorporation of customer needs and wishes in the product.
Action: Customer and prospect requirements are being gathered and registered, and the customer or prospect is informed of the developments concerning their requirements.
Prerequisite(s): -
Reference(s): Dieste, Juristo, & Shull (2008) [22]
Note(s): -

F Partner involvement

Goal: Improved product quality & increased involvement of external stakeholders in the product management process.
Action: Requirements are systematically gathered from partner companies.
Prerequisite(s): -

Note(s): -

Requirements identification

Requirements identification identifies the actual Product Requirements by rewriting the Market Requirements to understandable Product Requirements, and connecting requirements that describe similar functionality.

A Uniformity

Goal: Identification of the essence of the requirements, this provides clarity to all involved, enables a meaningful comparison of requirements.
Action: Market requirements are rewritten to product requirements using a pre-defined template if the market requirement is applicable to a product.
Prerequisite(s): Requirements gathering A
Note(s): -

B Requirements validation

Goal: Validation of the requirements to prevent rework.
Action: The correctness ("Is the definition correct?"), completeness ("Does the requirement describe all relevant aspects?"), and unambiguousness ("Can the requirement only be interpreted in one way?") of the requirement is validated.
Prerequisite(s): Requirements gathering A
Note(s): -

C Connect similar requirements

Goal: Identify the true need for requirements (e.g. two requirements that individually are not valued high enough could be valued high enough when merged), prevention of double requirements.
Action: Group together market requirements which describe similar functionality by linking market requirements and product requirements to each other.
Prerequisite(s): Requirements gathering A, Requirements gathering B
Note(s): -

D Automatically connect similar requirements

Goal: Reduce the workload of the connecting of similar requirements.
Action: Automatically connect similar requirements by using advanced techniques such as linguistic engineering.
Prerequisite(s): Requirements gathering A, Requirements gathering B
Note(s): -

Requirements organizing

Requirements organizing organizes the requirements throughout their entire lifecycle based on shared aspects, and describes the dependencies between Product Requirements.

A Requirement organization

Goal: Increase potential of requirements by identifying value outside of the original boundaries, and provide insight into the planning concerning the requirement.
Action: Product requirements are organized based on shared aspects (e.g. type, function, or core asset).
Prerequisite(s): Requirements gathering A
Note(s): -

B Requirement lifecycle management

Goal: Make requirements reusable for other projects, adds traceability for a requirements (easy to gather extra information, discover mistakes)

Action: A requirements history is logged by recording who submitted it, when it was submitted, what changes were made to it, what the original description of the requirement was, what the current status of the requirement is (e.g. new, rewritten, validated, organized, scheduled for release X, tested, released in release X, etc.). A requirement remains in the database after it has been build so that it can be reused in a new or related product.

Prerequisite(s): Requirements gathering A

Reference(s): Arendsen, Cannegieter, Grund, Heck, Klerk, & Zandhuis (2008) [23]

Note(s): -

C Requirement dependency linking

Goal: The existence of requirements interdependencies means that requirements interact with and affect each other. Requirement dependency linking prevents problems that result from these interdependencies, and therewith enables better planning of the development process.

Action: Dependencies between market and product requirements are determined and registered. A dependency exists when a requirement requires a specific action of another requirement. E.g. a requirement requires that another requirement be implemented too, or that another requirement is not implemented in case of conflicting requirements. The linkage can be supported by using advanced techniques, such as linguistic engineering.

Prerequisite(s): Requirements gathering A

Reference(s): Dahlstedt & Persson (2003) [24]

Note(s): -

Release Planning

Release planning consists of the SPM capabilities needed to successfully create and launch a release.

Requirements prioritization

The identified and organized requirements are prioritized.

A Internal stakeholder involvement

Goal: Improved product quality & increased involvement of internal stakeholders in the product management process.

Action: All relevant internal stakeholders (e.g. the product manager, support, services, development, sales & marketing, research & development) indicate the requirements that should be incorporated in future releases by assigning priorities to the requirements from their point of view.

Prerequisite(s): Requirements gathering A

Note(s): -

B Prioritization methodology

Goal: Structure the requirement prioritization process and therewith provide a solid prioritization which is balanced, and clear to all parties involved.

Action: A structured prioritization technique is used (e.g. MOSCOW, Wiegers).

Prerequisite(s): Requirements gathering A

Note(s): -

C Customer involvement

Goal: Incorporation of customer needs and wishes in the product.

Action: Customers and prospects (or representatives thereof) 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 in a delegation, select group of customers, or in other more manageable forms.
Prerequisite(s): Requirements gathering A , Requirements gathering B
Note(s): -

D Cost revenue consideration

Goal: Create a financial basis for the prioritization.
Action: Information about the 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).
Prerequisite(s): Requirements gathering A
Note(s): -

E Partner involvement

Goal: Improved product quality & increased involvement of external stakeholders in the product management process.
Action: Partner companies indicate the requirements that should be incorporated in future releases by assigning priorities to the requirements from their point of view.
Prerequisite(s): Requirements gathering A
Note(s): -

Release definition

During the 'Release definition' process, the requirements that will be implemented in the next release are selected, based on the prioritization they received in the preceding process. And the release definition is created based on the selection.

A Basic requirements selection

Goal: Create a realistic release selection.
Action: During requirements selection for the next release, constraints concerning engineering capacity are taken into account.
Prerequisite(s): -
Note(s): -

B Standardization

Goal: Create clarity, enable comparison of releases.
Action: A standard template is used to write the release definition. The release definition contains aspects such as an overview of the requirements that will be implemented, a time path, and the needed capacity.
Prerequisite(s): -
Note(s): -

C Internal communication

Goal: Inform the internal stakeholders of the upcoming development.
Action: The release definition is communicated to the internal stakeholders.
Prerequisite(s): Release definition A
Note(s): -

D Advanced requirements selection

Goal: Optimize the release selection

Action: The optimal release is automatically calculated based upon the constraints of the requirements. At minimum the engineering capacity, priorities, cost, requirement dependencies are all taken into account.
Prerequisite(s): Release definition A, Requirements organizing C
Note(s): -

E Multiple releases

Goal: Create a more detailed mid-term vision the product.
Action: Multiple releases are included in the requirements selection process.
Prerequisite(s): -
Note(s): -

Release definition validation

The 'Release definition validation' is performed before the release is built by the development department. It focuses on the validation of the release definition by internal parties.

A Internal validation

Goal: Increase quality of releases, generate awareness among internal stakeholders.
Action: The release definition is checked by internal stakeholders, before the software is realized.
Prerequisite(s): Release definition A
Note(s): -

B Formal approval

Goal: Increase release quality, improve internal acceptance.
Action: Approval standards are determined and verified by the board before the software is realized (turned over to development).
Prerequisite(s): Release definition A
Note(s): -

C Business case

Goal: Verify real world viability of release.
Action: A business case (including the ROI) is being written before the software is realized.
Prerequisite(s): Release definition A, Requirements prioritization D
Note(s): -

Scope change management

Scope change management handles the different kinds of scope changes that can occur during the development of a release.

A Event notification

Goal: Create awareness of the problem, learn from the problem for future projects.
Action: A formal scope change management process is in place, in which all involved stakeholders are informed.
Prerequisite(s): -
Note(s): -

B Milestone monitoring

Goal: Create more insight into the development process by introducing milestones.

Action: Key dates and checkpoints are monitored in the product delivery.
Prerequisite(s): -
Note(s): -

C Impact analysis

Goal: Determine the impact of the problems to be able to inform all stakeholders.
Action: An impact analysis is performed to determine the effects of the scope change.
Prerequisite(s): -
Note(s): -

D Scope change handling

Goal: Minimize effects of scope change.
Action: A process is in place to develop alternative plans, with all relevant stakeholders, to react to the effects of the scope change.
Prerequisite(s): Scope change management C
Note(s): -

Build validation

The Release build validation is performed after the release has been built by the development department. It focuses on the validation of the built release before the release candidate is launched.

A Internal validation

Goal: Improve product quality.
Action: Internal stakeholders (consultants, etc.) perform a functional validation of the build release to verify that it meets the expected outcome.
Prerequisite(s): -
Note(s): -

B External validation

Goal: Improve product quality.
Action: The build is validated by external parties (customers, partners) to verify the builds quality (e.g. by settings up a pilot).
Prerequisite(s): -
Note(s): -

C Certification

Goal: Improve product quality, get independent confirmation of product quality to prove the quality of your product.
Action: Certification by an independent external party is acquired for the release.
Prerequisite(s): -
Note(s): -

Launch preparation

Launch preparation prepares the internal and external stakeholders for the launch of the new release. It addresses issues ranging from communication, to documentation, training, and the preparations for the implementation of the release itself.

A Internal communication

Goal: Inform all internal parties involved of the upcoming release.
Action: Information about the upcoming new release is communicated to the internal stakeholders. This information contains a description of the most

important changed and added features, the estimated release date, possible costs involved, information about how the new release can be obtained, possible training dates, etc.

Prerequisite(s): -
Note(s): -

B Formal approval

Goal: Higher quality of releases.
Action: A formal 'go', based upon standard quality rules, must be obtained from the board before the launch can begin.
Prerequisite(s): -
Note(s): -

C External communication

Goal: Inform all external parties involved of the upcoming release.
Action: Information about the upcoming new release is communicated to the external stakeholders. This information contains a description of the most important changed and added features, the estimated release date, possible costs involved, information about how the new release can be obtained, possible training dates, etc.
Prerequisite(s): -
Note(s): -

D Training

Goal: Ensure a smooth transition to the new version, enable optimal use of the new version.
Action: Trainings are organized and documentation is updated for both internal parties (e.g. service desk, consultants) and external parties (e.g. customers, partner companies) to help educate them in the new release.
Prerequisite(s): -
Note(s): -

E Launch impact analysis

Goal: Ensure a smooth transition to the new version (on time, without problems).
Action: Determine how much time it is going to take to implement the new release at the individual customers, and what type of experts are needed to perform the implementation (e.g. database experts).
Prerequisite(s): -
Note(s): -

F Sales & marketing support

Goal: Ensure external corporate expressions are correct.
Action: Create a checklist of all external expression of the product (e.g. fact sheets, demo's, presentations) that may need to be updated by changes made in latest release of the product. These items must be checked, and possible updated before they are available to external parties (e.g. customers, partners).
Prerequisite(s): -
Note(s): -

Product Planning

Product planning is concentrated around the gathering of information for, and creation of a roadmap for a product or product line, and its core assets.

Roadmap intelligence

Roadmap intelligence gathers decision supporting information needed in the creation of the product roadmap and presents it in summary style suited for management information. It does not include the requirements gathered in Requirements management.

A Product analysis

Goal:	Show how your product responds to / fits the trends, how you will take advantage of the momentum.
Action:	A plan is created showing which markets you will be going after and how you plan to develop the products for each segment. Eg., in year one you may plan to enter the automotive market by partnering with another company, or you may want to enter the pharmaceutical market in year two by building products inhouse or acquiring products.
Prerequisite(s):	-
Reference(s):	-
Notes: -	

B Society trends

Goal:	Show how your product responds to / fits the trends, how you will take advantage of the momentum.
Action:	An overview is created showing the big picture of important trends in society in the coming years. This picture contains a general view and a view specific for your products industry.
Prerequisite(s):	-
Reference(s):	-
Notes:	One way to identify the important topics for the society roadmap is to perform a PEST analysis and then show the development regarding these topics.

C Technology trends

Goal:	Making sure and being able to show how your product is staying up-to-date and is taking advantage of opportunities provided by current and up-and-coming technologies.
Action:	An overview is created showing the big picture of important developments in terms of technology in the coming years. This picture contains a general view and a view specific for your products industry.
Prerequisite(s):	-
Reference(s):	-
Note(s):	Following reports of research firms such as Gartner and Forrester can be a very useful sources of information for trends.

D Competition trends

Goal:	Making sure and being able to show how your product is staying up-to-date and is taking advantage of opportunities provided by your partners.
Action:	An overview is created showing what competing products are doing in terms of their product development in the coming years. The general developments trends among your competitors are shown, and the developments of the most important competing products are depicted with special attention.
Prerequisite(s):	-
Reference(s):	Roadmap intelligence A

Notes: Porters Five forces model can be used to determine the different types of competitors

E Partner roadmap

Goal: Show how your organization responds to developments of partner products and which your own products rely.

Action: An overview is created showing what your partners will be developing the coming period. Examples of partner products are operating systems, development environments, database, etc. The overview shows what will be happening with the core platform software as well as what the partner organization will be delivering in terms of their own products and development tools that your organization can or will need to use to support the partner products/components.

Prerequisite(s): -

Reference(s): -

Note(s): -

Core asset roadmapping

Core asset roadmapping concerns the planning of future development of core assets (components that are shared by multiple products).

A Centralized registration

Goal: Enable the reuse of components.

Action: All core assets are registered in a standardized manner, and are stored in a central location.

Prerequisite(s): -

Note(s): -

B Core asset identification

Goal: Increase and simplify the reuse and maintenance of components.

Action: Common components/functionality (core assets) is systematically identified among the organizations products and deliverables surrounding the product.

Prerequisite(s): -

Note(s): -

C Make or buy decision

Goal: Cost and time savings by using external parties.

Action: A process is in place to actively investigate make-or-buy decisions: external sources are investigated based on ROI in the search for core asset acquisition: partners, outsourcing or subcontracting of development.

Prerequisite(s): -

Note(s): -

D Core asset roadmap construction

Goal: Provide insight in the future plans for the core assets to ensure that this is incorporated in the product roadmap in a realistic and optimal form.

Action: A roadmap is created for the core assets, this roadmap shows how the core assets are sustained, upgraded, and enhanced. This roadmap contains both existing core assets, and core assets that are in development.

Prerequisite(s): Core asset registration A

Note(s): -

Product roadmapping

Product roadmapping deals with the actual creation of the product roadmap itself.

A Short-term roadmap

Goal: Development of a short-term vision of the product(s).
Action: A roadmap is developed detailing the short-term plans. The plans span more than one release.
Prerequisite(s): -
Reference(s): -
Note(s): Internal roadmaps can be used for defining development priorities, communicating to upper management and other departments and for use in obtaining funding for the company.

B Theme identification

Goal: Structuring of releases and roadmaps: themes are used give a clear direction to the roadmap and later on to structure the requirements.
Action: Release themes are identified and maintained. Themes are decided on together with the internal stakeholders. Identification of the themes results in a list of release themes that are stored centrally, so that requirements, core assets, market trends etc. can be linked to it.
Prerequisite(s): -

C Internal consultation

Goal: Organization wide acceptance of the product roadmap. Optimal use of all knowledge in the organization to create more rich and realistic product roadmaps
Action: Product roadmaps are created in consultation with all relevant internal stakeholders.
Prerequisite(s): -

D Long-term roadmap

Goal: Development of a long-term vision of the product(s).
Action: The roadmap spans a time period of at least four years.
Prerequisite(s): Product roadmapping A

E External variants

Goal: Informing of customers/managing customers expectations, marketing tool. Informing external parties using information they want.
Action: Less detailed variants of the internal roadmap are created for specific external parties (e.g. customers, partners, investors).
Prerequisite(s): Product roadmapping A
Reference(s): -
Note(s): External roadmaps are used for communicating to customers, partners, press and analysts. External roadmaps are based on the corresponding internal roadmaps, but should be less detailed. An external roadmap should be fine-tuned for the specific party it is intended for.

Portfolio Management

Portfolio management concerns the strategic information gathering and decision making across the entire product portfolio.

Market analysis

Market analysis gathers decision supporting information about the market needed to make decisions about the product portfolio of an organization.

A Market trend identification

Goal: Widen your product base.
Action: There is an active search for market opportunities to either expand existing products to, or create new products for. This search exists of doing market research in markets related to or similar to your organizations markets, visiting conferences, listening to customers, etc. All search findings are documented.
Prerequisite(s): -
Reference(s): -
Note(s): -

B Market strategy

Goal: Plan which markets you will target and how you will enter them.
Action: A plan is created showing which markets your product will be going after and how you plan to develop the products for each segment. E.g., in year one you may want to enter healthcare by partnering with another company. Or you may want to enter the financial market in year two by building products in-house or acquiring products.
Prerequisite(s): -
Reference(s): -
Note(s): -

C Customer win/loss analysis

Goal: Learn about your customers/prospects, to generate more future customers by tuning product development to them.
Action: A win/loss analysis is performed to research why customers chose or did not choose to buy your organizations products. This capability looks further than just the product features, e.g. the sales process is reviewed.
Prerequisite(s): -
Reference(s): -
Note(s): -

D Competitor analysis

Goal: Learn from competitors and do not fall behind product-wise.
Action: A competitor analysis is performed on an organizational level to analyze what competitors offer, what their strengths are and are going to offer compared to your organizations.
Prerequisite(s): -
Reference(s): -
Note(s): A SWOT analysis is a very useful way to map your own strengths/weaknesses and the opportunities and threats that follow from them against your competitors

E Custom market trend identification

Goal: Gain unique information (that your competition does not have) specific to your own organization. Gain an unbiased insight into your market and/or operations.
Action: External market research parties are used to perform a market analysis specifically for the organizations product portfolio.

Prerequisite(s): -
Reference(s): -

Partnering & contracting

Partnering & contracting focuses on establishing partnerships, pricing, and distribution aspects in which the product manager is involved.

A Service level agreements

Goal: Manage customer expectations.
Action: (Standard) service level agreements (SLA's) are set up for customers.
Prerequisite(s): -
Reference(s): Trienekens, et al. (2004) [25]
Note(s): -

B Intellectual property management

Goal: Protection of the organizations intellectual property, and prevention of problems due to misuse of the intellectual property of other organizations.
Action: Measures are in place to protect the intellectual property of the own organization, and to manage the used intellectual property from other organizations.
Prerequisite(s): -
Reference(s): -
Note(s): -

C Investigate distribution channels

Goal: Improve sales process.
Action: A process is in place to periodically verify the current distribution channels, and identify alternative distribution channels.
Prerequisite(s): -
Reference(s): -
Note(s): -

D Establish and evaluate pricing model

Goal: Improve sales process.
Action: A process is in place to establish the pricing model and periodically verify whether it still fits the market.
Prerequisite(s): -
Reference(s): -
Note(s): -

E Monitored partner network

Goal: Set up partner networks to gain synergetic advantages.
Action: A partner network and/or partner portals are used to regulate partnering. Key performance indicators are set up to monitor the performance of partners on a regular basis.
Prerequisite(s): -
Reference(s): -
Note(s): -

Product lifecycle management

Product lifecycle management concerns the information gathering and key decision making about product life and major product changes across the entire product portfolio.

A Product life cycle analysis

Goal:	Ensure that there is a healthy balance between new and old products in the product portfolio, create awareness of the products life expectations.
Action:	The current life phase is determined, at least once per year, for each product in the organizations portfolio. This analysis is based on both financial and technical aspects. Information is thus gathered from all relevant internal stakeholders (e.g. company board, sales, development).
Prerequisite(s):	-
Reference(s):	-
Note(s):	The Boston Consultancy Group Matrix is an easy tool to create an overview of your products life phase(s). And your financial risks.

B Portfolio innovation

Goal:	Balance the products in the product portfolio to make sure that products do not become competitors.
Action:	A decision process is in place to decide whether or not to incorporate trends in one of the current products or in newly to be developed products.
Prerequisite(s):	-
Reference(s):	-
Note(s):	-

C Portfolio scope analysis

Goal:	Balance products in the portfolio and identify opportunities for reuse (overlap) and discover possible new market segments (gaps).
Action:	A product scope analysis is performed to identify overlaps and gaps between the products in the organizations product portfolio.
Prerequisite(s):	-
Reference(s):	-
Note(s):	-

D Business case

Goal:	Validation of major future plans before they are put into practice.
Action:	A business case is performed for major product revisions (revisions spanning multiple release) or when the product strategy is changed. We use Kittlaus & Clough (2009) definition in which a business case is the “comparison of the costs associated with the product or project to the quantified economic benefits or value to be derived”.
Prerequisite(s):	-
Reference(s):	Kittlaus & Clough (2009) [26]
Note(s):	-

E Product lines

Goal:	Enable maximum reuse of resources and simplify the creation of new products.
Action:	Product lines are developed. The architecture of the product line is documented, and its goal is clearly defined. A software product line is defined as a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way (Clements & Northrop, 2002).
Prerequisite(s):	-
Reference(s):	Clements & Northrop (2002) [27]
Note(s):	-

Appendix C: List of interviewed persons

Company A

- Development Manager
- Founder and Chief Technology Officer
- Implementation Engineers (three different persons)
- Product Analyst
- Support Manager
- Sales Manager
- Salesman

Company B

- Business Process Manager
- Customer Service Manager
- Operations Manager Netherlands
- Product Manager
- QA Manager
- Sales Manager Benelux
- Vice President Product Management
- Vice President Research & Development

Appendix D: Semi-structured interview protocol

Introduction

- Introduce myself, the interviewer, to the interviewee
- Explain the goal of research: "This research is part of my master thesis project. I want to determine the release planning process maturity level within Company A and Company B and adapt/improve where needed".
- Explain the goal of the interview: "I want to model the release planning processes in order to determine their maturity. You are one of the stakeholders in this process, so I would like to ask you some questions about it."
- Recording permission: "I would like to record this interview so I can hear it back at a later moment. Is that alright? No one besides me will have access, or get to hear these recordings."

General questions

- The interviewee's name
- The interviewee's department/function
- How long have you been employed within company A or B?
- How long have you been working in your current function?

Specific questions

- Explain what 'the release planning process' means within the context of this project:
  "Release planning is the process of making the decision about what new functionalities or changes will be implemented in which release of the software. [Show the figure below] The release planning process thus contains the activities of gathering all the new requirements for a product and deciding which ones will get into the product and in which version of the project."

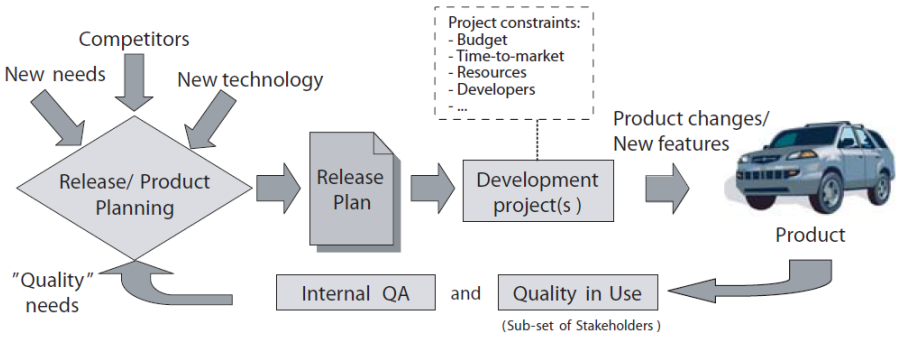


FIGURE 17 OVERVIEW OF RELEVANT FACTORS OF RELEASE PLANNING (LINDGREN, LAND, NORSTRÖM, & WALL, 2008)

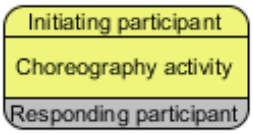
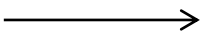
- What is your role is the release planning process?
- Who are other relevant actors in the release planning process?
- What activities are performed in the release planning process?
  - Which ones do you perform?
- What are the inputs and outputs of the activities?
- What documentation is used in these activities?
  - What are the inputs?
  - What are the outputs?
- Which tools/software packages are used to support the activities?

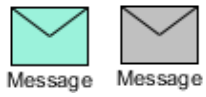
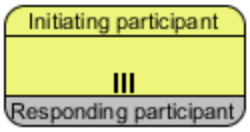
Closing

- Is there anything else you might want to add to this interview?
- Thank the interviewee for his time and effort.

Appendix E: BPMN modelling elements

TABLE 16 BMPN MODELLING ELEMENTS

Notation	Element	Description
	Event	An Event is something that "happens" during the course of a Process or a Choreography. These Events affect the flow of the model and usually have a cause (*Trigger*) or an impact (*Result*). Events are circles with open centers to allow internal markers to differentiate different *Triggers* or *Results*. There are three types of Events, based on when they affect the flow: Start, Intermediate, and End. As the name implies, the Start Event indicates where a particular Process or Choreography will start. As the name implies, the End Event indicates where a Process or Choreography will end.
	Off-page connector	Generally used for printing, this object will show where a Sequence Flow leaves one page and then restarts on the next page. A Link Intermediate Event can be used as an Off-Page Connector
	Activity	An Activity is a generic term for work that company performs in a Process. An Activity can be atomic or non-atomic (compound). The types of Activities that are a part of a Process Model are: Sub-Process and Task, which are rounded rectangles. Activities are used in both standard Processes and in Choreographies.
	User Task	A User Task is a typical "workflow" Task where a human performer performs the Task with the assistance of a software application .
	Choreography activity	A Choreography Task is an atomic Activity in a Choreography. It represents a set of one (1) or more Message exchanges. Each Choreography Task involves two (2) *Participants*. The name of the Choreography Task and each of the *Participants* are all displayed in the different bands that make up the shape's graphical notation. There are two (2) or more *Participant* Bands and one Task Name Band.
	Gateway - Exclusive	A Gateway is used to control the divergence and convergence of Sequence Flows in a Process and in a Choreography. This Decision represents a branching point where Alternatives are based on conditional Expressions contained within the *outgoing* Sequence Flows. Only one of the Alternatives will be chosen.
	Fork / merge	BPMN uses the term "fork" to refer to the dividing of a path into two or more parallel paths (also known as an AND-Split). It is a place in the Process where activities can be performed concurrently, rather than sequentially. BPMN uses the term "merge" to refer to the exclusive combining of two or more paths into one path (also known as an OR-Join).
	Sequence flow	A Sequence Flow is used to show the order that Activities will be performed in a Process and in a Choreography.

	Message	A Message is used to depict the contents of a communication between two *Participants* of a choreography task. A blue envelop indicates communication from the initiating participant. A grey envelop indicates communication from the responding participant.
	Multiple instances	A set of three vertical lines will be displayed at the bottom-center of a choreography activity to indicate sequential Multi-Instances (i.e. more than one (1) responding participant is involved in the choreography task).

Appendix F: Release clock at company B

The release clock at company B, figure 18, defines a number of stages that a new feature has to pass before it can be released. It starts counting at 12 o'clock, where the development of a new functionality begins. Effectively, the stages start at 3 o'clock where the development has finished and the software is handed over from development to testing. At 7 o'clock software is released to all customers. 11 o'clock indicates the end of life of a feature. At certain stages on the clock, indicated by the letters, a number of checks have to be done to guarantee quality of the release process. The checks that have to be done and which departments have to do them are specified in table 17 on the next page.

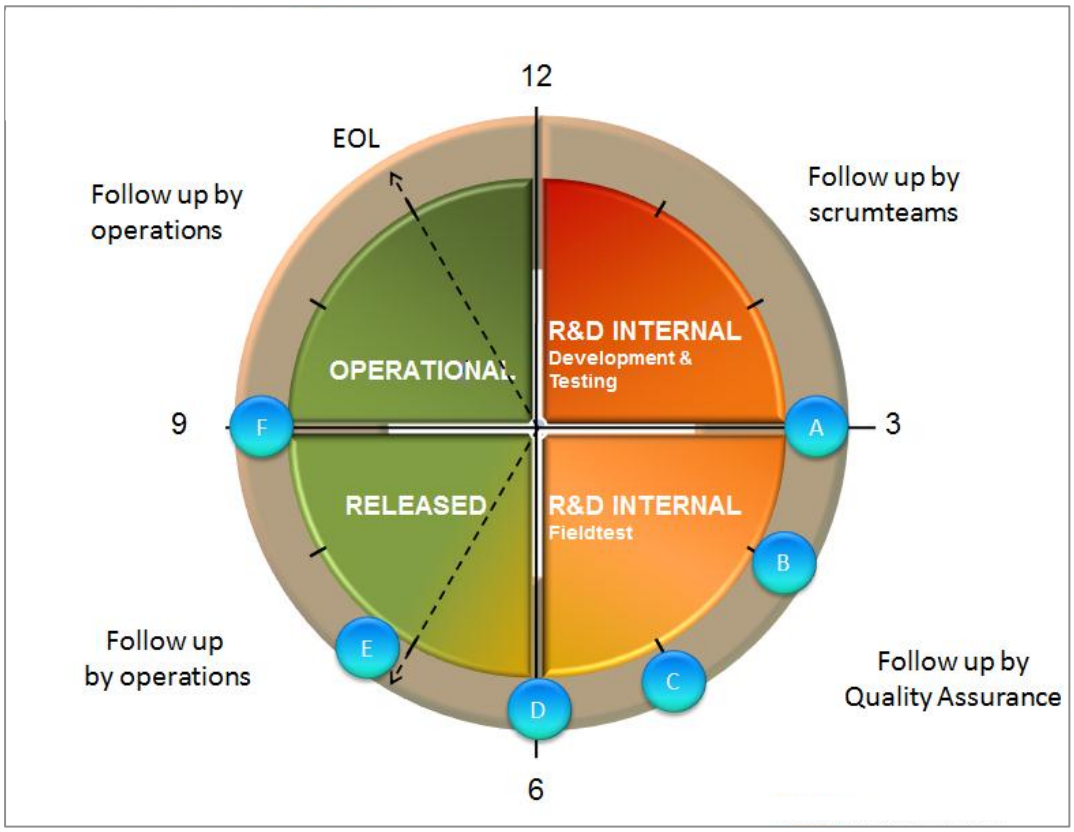


FIGURE 18 RELEASE CLOCK AT COMPANY B

TABLE 17 CHECKLIST OF RELEASE CLOCK

Department	TO DO	Deadline hour
QA	HW-combinations test	3
QA	Field test-environment set up	3
Product Marketing	Prices product defined	4
Product Marketing	CCP product defined	4
Product Marketing	Prices recurring defined	4
Product Marketing	CCP recurring defined	4
Product Marketing	Buying price product is clear	4
QA	Field test successful at 1 customer	4
OPERATIONS	Define customer training needs	4
Product Marketing	Translations done	5
QA	Manuals created	5
Product Marketing	Sales deliveries created	5
QA	Hardware installation guide created	5
CBO	Operational activities created in Ax	5
CBO	Sales items created in Ax	5
QA	Field test successful at 2 customers	5
OPERATIONS	Define project setup	5
QA	Online help function updated	5
Product Management	Operations people trained	6
Product Marketing	Sales people informed	6
QA	Field test installations reached and stable (50)	6
Product Marketing	One pager created	6
QA	Communication field test finished	6
QA	Field test successful at 3 customers	6
HOSTING	Demo-environment set up	6
HOSTING	Logins for sales and Operations created	6
ServiceDesk Manager	Product support package ready	6
SALES	Selling strategy defined	6
OPERATIONS	Create training deliverables	6
OPERATIONS	OPS tester planned for initial OPS release customer	6
Legal	If needed: adaptation of terms and conditions	6
LOGISTICS	Check availability of unit	6
QA	Add items to compatibility matrix	6
Marketing Communication	Information to customers	7
Marketing Communication	Information on website	7
Product Marketing	Product added to the offer template	7
Product Marketing	Recurring added to the offer template	7
Product Marketing	Services added to the offer template	7
HOSTING	Operational release installations reached (200)	7
ServiceDesk Manager	Communication item operational released	7
HOSTING	Customer and pilot-environment set up	7
Product Marketing	Webcast done	7

Appendix G: Volere requirements specification template items

The user's problem

A short description of the work context and the situation that triggered the development effort. It should also describe the work that the user wants to do with the delivered product.

Goal of the requested feature

This boils down to one, or at most a few, sentences that say "What do we want this product for?" In other words, the real reason that the product is being developed.

Examples

"We want to give immediate and complete response to customers ordering our goods over the telephone."

"We want to be able to forecast the weather."

Client or customer

This item must give the name of the client. It is permissible to have several names, but more than three negates the point.

Users of the feature to be developed

A list of the potential users of the product.

Examples

Users can come from wide, and sometimes unexpected, sources. Consider the possibility of your users being clerical staff, shop workers, managers, highly-trained operators, general public, casual users, passers-by, illiterate people, tradesmen, students, test engineers, foreigners, children, lawyers, remote users, people using the system over the telephone or Internet connection, emergency workers, and so on.

Functional requirements

A specification for each individual functional requirement. Specifications of detailed functional requirements that must be supported by the product.

Data requirements

A specification of the essential subject matter/business objects/entities/classes that are germane to the system. This might take the form of a first-cut data model, an object model or a domain model.

Look and feel Requirements

Requirements relating to spirit of the interface. Your client may have given you particular demands/requirements for the interface.

Ease of learning

A statement of how easy it should be to learn to use the product. This will range from zero time for products intended for placement in the public domain (for example a parking meter or a web site) to a considerable time for complex, highly technical products.

Examples

"The product shall be easy for an engineer to learn."

"A clerk shall be able to be productive within a short time."

"The product shall be able to be used by members of the public who will receive no training before using it."

"The product shall be used by engineers who will attend 5 weeks of training before using the product."

Understandability requirements

This specifies the requirement for the product to be understood by its users. While usability refers to ease of use, efficiency etc., understanding determines whether the users instinctively know what the product will do for them. In other words, the product fits into their view of the world.

Examples

"The product shall use symbols and words that are naturally understandable by the user community".

"The product shall hide the details of its construction from the user."

Speed and latency requirements

Specifies the amount of time available to complete specified tasks. These often refer to response times. They can also refer to the product's ability to fit into the intended environment.

Examples

"Any interface between a user and the automated system shall have a maximum response time of 2 seconds"

"The response shall be fast enough to avoid interrupting the user's flow of thought"

"The product shall poll the sensor every 10 seconds"

"The product shall download the new status parameters within 5 minutes of a change"

Safety critical requirements

Quantification of perceived risk of possible damage to people, property and environment. Note that different countries have different standards so the fit criteria must specify precisely which standards the product must meet.

Examples

"The product shall not emit noxious gases that damage people's health."

"The heat exchanger shall be shielded from human contact."

Precision or accuracy requirements

Quantification of the desired accuracy of the results produced by the product.

Examples

"All monetary amounts shall be accurate to 2 decimal places."

"Accuracy of road temperature readings shall be within + or – 2 degrees centigrade."

Reliability and availability requirements

This quantifies the necessary reliability of the product. This is usually expressed as the allowable time between failures, or the total allowable failure rate. It also quantifies the expected availability of the product.

Examples

"The product shall be available for use 24 hours per day, 365 days per year."

"The product shall be available for use between the hours of 8:00am and 5:30pm."

"The escalator shall run from 6am until the last flight arrives at 10pm."

"The product shall achieve 99% up time."

Robustness or fault tolerance requirements

Robustness specifies the ability of the product to continue to function under abnormal circumstances.

Examples

"The product shall continue to operate in local mode whenever it loses its link to the central server."

"The product shall provide 10 minutes of emergency operation should it become disconnected from the electricity source."

Capacity requirements

This specifies the volumes that the product must be able to deal with and the numbers of data stored by the product.

Examples

"The product shall cater for 300 simultaneous users within the period from 9:00am to 11:am. Maximum loading at other periods will be 150."

"During a launch period the product shall cater for up to 20 people to be in the inner chamber."

Scalability or extensibility requirements

This specifies the expected increases in size that the product must be able to handle. As a business grows (or is expected to grow) our software products must increase their capacities to cope with the new volumes.

Examples

"The product shall be capable of processing the existing 100,000 customers. This number is expected to grow to 500,000 within three years."

"The product shall be able to process 50,000 transactions an hour within two years of its launch."

Longevity requirements

This specifies the expected lifetime of the product.

Examples

“The product shall be expected to operate within the maximum maintenance budget for a minimum of 5 years”.

Expected physical environment

This specifies the physical environment in which the product will operate.

Examples

“The product shall be used by a worker, standing up, outside in cold, rainy conditions.”
“The product shall be used in noisy conditions with a lot of dust.”
“The product shall be able to fit in a pocket or purse.”
“The product shall be usable in dim light.”
“The product shall be not add to the noise in the environment.”

Expected technological environment

Specification of the hardware and other devices that make up the operating environment for the new system.

Partner applications

Description of other applications with which the product must interface.

Examples

“We must be able to interface with any html browser.”
“The new version of the spreadsheet must be able to access data from the previous 2 versions”
“Our product must interface with the applications that run on the remote weather stations”

Productization requirements

Any requirements that are necessary to make the product into a distributable or saleable item. It is also appropriate to describe here the operations to be performed to have a software product successfully installed.

Examples

“The product shall be distributed as a ZIP file. ”
“The product shall be able to be installed by an untrained user without recourse to separately-printed instructions.”
“The product shall be of a size that it can fit onto one CD.”

Maintenance requirements

A quantification of the time necessary to make specified changes to the product.

Examples

“New MIS reports must be available within one working week of the date the requirements are agreed”
“A new weather station must be able to be added to the system overnight”

Special conditions that apply to the maintenance of the product

Specification of the intended release cycle for the product and the form that the release will take.

Examples

“The maintenance releases will be offered to end-users once a year.”
“Every registered user will have access to our help site via the Internet.”

Supportability requirements

This specifies the level of support that the product requires. This is often done using a help desk. If there are to be people who provide support for the product, is that considered part of the product and are there any requirements for that support. You might also build support into the product itself, in which case this is the place to write those requirements.

Adaptability requirements

Description of other platforms or environments to which the product must be ported.

Examples

“The product is expected to run under Windows XP and Linux”

“The product might eventually be sold to the Japanese market”
“The product is designed to run in offices, but we intend to have a version which will run in restaurant kitchens”

Installation requirements

Description of the effort needed to install the product.

Examples

“The product shall be able to be installed in the specified environment within 2 working days.

Access requirements

Specification of who has authorized access to the product (both functionality and data), and under what circumstances that access is granted, and to what parts of the product access is allowed.

Examples

“Only direct managers can see the personnel records of their staff.”
“Only holders of current security clearance can enter the building.”

Integrity requirements

Specification of the required integrity of databases and other files, and of the product itself.

Examples

“The product shall prevent its data from incorrect data being introduced.”
“The product shall protect itself from intentional abuse.”

Privacy requirements

Specification of what the product has to do to insure the privacy of individuals that it stores information about. The product must also ensure that all laws about privacy of an individual’s data are observed.

Examples

“The product shall make its user aware of its information practices before collection data from them.”
“The product shall notify customers of changes to its information policy.”
“The product shall reveal private information only in compliance with the organization’s information policy.”

Audit requirements

Specification of what the product has to do (usually retain records) to permit the required audit checks. This may have legal implications. You are advised to seek the approval of your organization’s auditors for what you write here. You should also consider whether the product should retain information on who has used it. The intention is to provide security in the form that a user may not later deny having used the product, or participated in some form of transaction using the product.

Immunity requirements

The requirements for what the product has to do to protect itself from infection by unauthorized or undesirable software programs, such as viruses, worms, Trojan horses and others.

Cultural requirements

This contains requirements that are specific to the sociological factors that affect the acceptability of the product. If you are developing a product for foreign markets then these requirements are particularly relevant.

Examples

“The product shall not be offensive to religious or ethnic groups.”
“The product shall be able to distinguish between French, Italian and British road numbering systems.”
“The product shall keep a record of public holidays for all countries in the European Union and for all states in the United States.”

Political requirements

This contains requirements that are specific to the political factors that affect the acceptability of the product.

Examples

“The product shall be installed using component X.”
The product shall make all functionality available to the managing director.”
“The product shall be developed using XYZ standards.”

Compliance requirements

A statement specifying the legal requirements for this system.

Examples

“Personal information shall be implemented so as to comply with the data protection act.”

Standards requirements

A statement specifying applicable standards and referencing detailed standards descriptions. This does not refer to the law of the land, think of it as an internal “law” imposed by your company.

Example

“The product shall comply with MilSpec standards.”
“The product shall comply with insurance industry standards”.
“The product shall be developed according to SSADM standard development steps.”

Appendix H: Results of requirements specification questionnaire

Please rate the following items on how often they are present in you requirements specifications. Ranging from 1 being never, to 5 being always. If in your opinion an item is never included because it isn't required in these documents please select 'Not Applicable'.

TABLE 18 REQUIREMENTS SPECIFICATION QUESTIONNAIRE RESULTS AT COMPANY A

Specification items	1	2	3	4	5	N/A	Average
The user's problem				1	1		4,5
Goal of the requested feature			1	1			3,5
Client or customer		1		1			3
Users of the feature to be developed		2					2
Functional requirements		1	1				2,5
Data requirements	1	1					1,5
Look and feel requirements		1		1			3
Ease of learning		1			1		3,5
Understandability requirements		1			1		3,5
Speed and latency requirements		1			1		3,5
Safety critical requirements		1		1			3
Precision or accuracy requirements		1			1		3,5
Reliability and availability requirements		1			1		3,5
Robustness or fault tolerance requirements		1			1		3,5
Capacity requirements		1			1		3,5
Scalability or extensibility requirements		1			1		3,5
Longevity requirements		1			1		3,5
Expected physical environment		1		1			3
Expected technological environment			1		1		4
Partner applications			1		1		4
Productization requirements			1		1		4
Maintenance requirements		2					2
Special conditions that apply to the maintenance of the product		1		1			3
Supportability requirements			1	1			3,5
Adaptability requirements			1		1		4
Installation requirements			1		1		4
Access requirements			1		1		4
Integrity requirements		1			1		3,5
Privacy requirements		1		1			3
Audit requirements			1	1			3,5
Immunity requirements		1			1		3,5
Cultural requirements		1	1				2,5
Political requirements		2					2
Compliance requirements		1		1			3
Standards requirements		1	1				2,5

TABLE 19 REQUIREMENTS SPECIFICATION QUESTIONNAIRE RESULTS AT COMANY B

Specification items	1	2	3	4	5	N/A	Average
The user's problem			1	**2**			4,5
Goal of the requested feature				**2**	1		3,5
Client or customer		**1**	**1**	**1**			3
Users of the feature to be developed		**2**		1			2
Functional requirements			1	**2**			2,5
Data requirements		**2**		1			1,5
Look and feel requirements				1	**2**		3
Ease of learning		**1**		**1**	**1**		3,5
Understandability requirements				1	**2**		3,5
Speed and latency requirements		**1**		**1**	**1**		3,5
Safety critical requirements	**1**			**1**		**1**	3
Precision or accuracy requirements		**2**		1			3,5
Reliability and availability requirements				**2**		1	3,5
Robustness or fault tolerance requirements	**1**			**1**	**1**		3,5
Capacity requirements	1			**2**			3,5
Scalability or extensibility requirements			1	**2**			3,5
Longevity requirements	**1**	**1**		**1**			3,5
Expected physical environment	**1**			**1**		**1**	3
Expected technological environment	**1**			**1**		**1**	4
Partner applications		**1**		**1**	**1**		4
Productization requirements		**2**		1			4
Maintenance requirements		**2**		1			2
Special conditions that apply to the maintenance of the product		**2**		1			3
Supportability requirements			1	**2**			3,5
Adaptability requirements		1		**2**			4
Installation requirements		**2**		1			4
Access requirements			**1**	**1**	**1**		4
Integrity requirements		**2**		1			3,5
Privacy requirements	**1**			**1**	**1**		3
Audit requirements	**1**			**1**		**1**	3,5
Immunity requirements	**1**			**1**	**1**		3,5
Cultural requirements		**1**		**1**		**1**	2,5
Political requirements				1		**2**	2
Compliance requirements				**1**	**1**	**1**	3
Standards requirements		**1**		**1**	**1**		2,5

Appendix I: Descriptions of prioritisation items

Competitors

Explanation: The status of the competitors with respect to the requirement. In other words, it is taken into account whether a competitor has the implied functionality implemented or not.
Motivation: We may feel forced to include a requirement if our competitors have the functionality, or we may want to implement something that is considered to be leading edge functionality (functionality competitors do not have).

Complexity

Explanation: The estimated complexity of the requirement and the associated challenges in implementing it.
Motivation: We may not want to include a requirement that is judged to be very complex to implement and as a consequence the risk of failure as too high.

Delivery date

Explanation: The ability to meet the deadline.
Motivation: We may not want to introduce a requirement that may affect the deadline of other projects negatively.

Development cost-benefit

Explanation: The actual cost-benefit for implementing the requirement.
Motivation: We may not want to include a requirement if the implementation cost is judged to be high in relation to the expected benefit.

Evolution

Explanation: The impact on the future evolution of the system.
Motivation: We may not want to implement a requirement if it is believed to make long-term evolution of the system more complicated.

Maintenance

Explanation: The impact on the maintenance of the current system.
Motivation: We may not want to implement a requirement if it is believed that the requirement may cause many problems in terms of maintenance.

Requirements dependencies

Explanation: The dependencies between this specific requirement and other requirements, either already implemented or other posed requirements.
Motivation: The dependency to other requirements (already implemented, scheduled to be implemented, or deferred to later release) may affect our decision regarding the current requirement.

Requirement's issuer

Explanation: The actual issuer of the requirement is taken into account, i.e. which stakeholder (internal or external) generated the requirement.
Motivation: We may judge some issuers as more important than others, for example, a very important customer or representative for an important market.

Requirement volatility

Explanation: This criterion is related to whether the requirement is likely to change or not.
Motivation: We may want to handle highly volatile requirements differently.

Resources/competencies

Explanation: The availability of resources with the right competencies to implement the requirement.
Motivation: We may not want to implement a requirement unless we are sure that we have the right people available for the job.

Stakeholder priority of requirement

Explanation: The priority of the requirement is taken into account.
Motivation: We may want to prioritize the requirements that our customers or markets think are of particular importance.

Support for Education/Training

Explanation: The ability and possibility to provide technical support, education, and training to customers, markets and so forth with respect to the requirement.
Motivation: We may not want to implement functionality unless we could provide the appropriate technical support, education and training in relation to the requirement.

System impact

Explanation: The impact of the requirement on the existing system.
Motivation: We may not want to implement a requirement if we judge that the actual impact in terms of changes to the existing system is too large.

Appendix J: Results of prioritisation item questionnaire

Please rate the following items on how important they are in the prioritisation of requirements, ranging from 1 being not important, to 5 being very important.

TABLE 20 RESULTS PRIORITISATION ITEMS QUESTIONNAIRE NORMAL REQUIREMENTS AT COMPANY A

Prioritisation items	Request handler average	CTO
Competitors	4	5
Complexity	4,25	3
Delivery data/Calendar time	4	4
Development cost-benefit	3,75	5
Evolution	3,25	4
Maintenance	3,5	3
Requirement volatility	3,5	3
Requirement's issuer	2,75	5
Requirements dependencies	3,25	3
Resources/competencies	3,75	5
Stakeholder priority of requirement	2	5
Support for Education/Training	3,25	2
System impact	4	4

Please rate the following items on how important they are in the decision to determine whether a requirement is of higher priority than normal, ranging from 1 being not important, to 5 being very important.

TABLE 21 RESULTS PRIORITISATION ITEMS QUESTIONNAIRE FOR URGENT REQUIREMENTS AT COMPANY A

Prioritisation items	Request handler average	CTO
Potential sales volume	4,75	5
Delivery data/Calendar time	4	4
Requirement's issuer	4	3
Stakeholder priority of requirement	3,75	5
Competitors	3,5	4