A Framework for Process Improvement in Software Product Management

Willem Bekkers[1], Inge van de Weerd[1], Marco Spruit[1] and Sjaak Brinkkemper[1]

[1] Utrecht University, Padualaan 14, Centrum Gebouw Noord, 3508 TB Utrecht, NL, {bekkers, i.vandeweerd, m.r.spruit, s.brinkkemper}@cs.uu.nl

Abstract. This paper presents a comprehensive overview of all the important areas within Software Product Management (SPM). The overview has been created and validated in collaboration with many experts from practice and the scientific community. It provides a list of 68 capabilities a product software organization should implement to reach a full grown SPM maturity. The overview consists of the SPM Competence Model that shows the areas of importance to SPM, and the SPM Maturity Matrix that lists all important activities within those areas in a best practice implementation order. SPM organizations can use this matrix to map and improve their SPM practices incrementally.

Keywords: Software Product Management, Software Process Improvement, SPM Maturity Matrix, SPM Competence Model, Situational Assessment Method

1 Introduction

As confirmed by recent research, Software Product Management (SPM) is a key area within many software companies [1, 2]. A product manager can also be referred to as the "mini-CEO" of an organization [3]. They are positioned at the center of the organization where they keep in contact with all stakeholders to ensure that they all work towards the same goal according to the strategies set out. As such, a large array of skills is expected; ranging from gathering requirements, to constructing roadmaps.

Although the product manager's function is essential in the product software industry, little education exists in this area [4]. To make things worse, no extensive body of knowledge exists, such as PMBOK [5] and SWEBOK [6]. This leads to a situation in which product managers learn their skills 'on-the-job', often starting out of a position as development, sales, or project manager. Problems arise when companies want to professionalize their product management practices, either to support the company's growth, or to make a shift from selling customized software to selling standard product software [7]. Due to the lack of knowledge, lifting the quality of the product by improving the SPM processes is often difficult.

To aid product managers in improving their SPM practices, we proposed the Reference Framework for SPM [4] and the Situational Assessment Method (SAM) [8]. A key component of the SAM is the SPM Maturity Matrix, which is used to

determine an organization's SPM maturity level and identify the areas that need improvement to reach a higher maturity level. All kinds of organizations, including small and medium sized organizations, should be able to use the maturity matrix as a guide for incremental process improvement in SPM. We focused on an incremental, or evolutionary, SPI approach for several reasons: a) it is a fundamental way to reduce risk in complex improvement projects [9], and b) we observe that in many organizations this is the natural way for method evolution [10, 11].

Since the proposal of SAM, the maturity matrix has been evaluated in several case studies. Feedback from these case studies led to a number of significant improvements. This paper presents a detailed overview of the SPM Competence Model and the maturity matrix for SPM and the research method we followed to achieve these improvements.

2 Research Design

This study follows the design science methodology, in which research is done through the processes of building and evaluating artifacts [12]. The artifacts in this research are the SPM Competence Model and the SPM Maturity Matrix. The research is performed as action research, since the lead author is both working as a researcher at Utrecht University as well as a consultant at the Dutch product software company Centric.

During our research we follow the five process steps of the design cycle [13]. This design cycle consists of steps that follow an iterative process. Knowledge produced in the process by constructing and evaluating the artifact is used as input for a better awareness of the problem. The five steps are: (1) *Awareness of the problem* – In section 1, we described the problem and its context. (2) *Suggestion* – The suggestion for a solution to the problem identified in step 1 is developed. In section 2, we describe our approach in tackling the problem and the research methods that we use. 3) *Development* – the artifacts that are developed are the SPM Competence Model and SPM Maturity Matrix, which are presented in section 3 and 4. (4) *Evaluation* – This step comprises the evaluation of the method. We used expert validations, a survey, case studies, and questionnaires to validate the method. The results of these extensive validations lead to a higher level of problem awareness and suggestions for solutions. Three case studies are presented in chapter 9 of [14]. 5) *Conclusion* – Finally, in section 5, conclusions and areas for further research are covered.

During this research, we made use of several data collection sources. Firstly, we performed a *literature study*. The literature study was based on a multitude of papers describing specific processes within the field of SPM (e.g. [15] & [16]). Secondly, a *brainstorm session* was conducted with experts from the scientific community to create a first version of the maturity matrix. The session consisted of two parts: 1) determination of the capabilities; 2) determination of the positioning of the capabilities relative to each other. The literature study was used as input for the brainstorm session. Furthermore, an *expert validation* was held where business professionals validated the results of the brainstorm session: the maturity matrix and the SPM Competence Model. Finally, we performed a *survey*, to fine-tune the positioning of the capabilities in the maturity matrix [17].

The SPM Maturity Matrix was applied by the authors in twelve *case studies* at product software organizations from the Netherlands to test the applicability in day-to-day business environments. The case studies consisted of a series of interviews performed at the organizations and an evaluation on how the organizations looked at the results. Furthermore, during a professional SPM course, under guidance of the authors, thirteen product managers from different organizations filled in a questionnaire where they applied the maturity matrix to their own organization.

Finally, we followed the following iterative process to fine-tune both the SPM Competence Model and Maturity Matrix: (1) make adjustments based on feedback, (2) validate the model with experts from practice, and (3) validate the model with experts from the scientific community. These steps were repeated over a period of 4 months until a consensus was reached among all experts (resulting in a total of twelve iterations). The results of this process are presented in this paper. During this validation process six experts from practice were consulted (each representing another SPM organization), and four experts from the scientific community were consulted.

3 The Software Product Management Competence Model

3.1 Introduction

The SPM Competence Model (Figure 1) presents an overview of all of the areas which are important to the field of the SPM. These areas are called focus areas. The relevant external and internal stakeholders are presented on left and right side of the model. The model does not include the development departments' activities of the product software organization. Development is simply one of the stakeholders that provide input to the SPM processes.

Four main business functions are defined in the model, namely: Requirements management, Release planning, Product planning, and Portfolio management. These business functions are based on the structure where a portfolio consists of products, a product consists of releases, and releases consist of requirements. The portfolio is represented in the Portfolio management function. The products are represented in the Product planning function. The releases are represented in the Release planning function. And finally, the requirements are represented in the Requirements management function.

Each business function consists of a number of focus areas (the white areas in figure 1), each of which represents a strongly coherent group of capabilities within a business function. These focus areas are explained further in the next section.

The model contains arrows between the stakeholders and the different business functions. This indicates the interaction that exists between the stakeholders and the different functions. Adjacent business functions also have strong interactions between them, which is indicated by the arrows in between the business functions. Finally, the arrows between focus areas indicate the main flow of the process and therewith information between the different focus areas. Note that this leaves room for interaction between focus areas that are not directly connected via arrows.

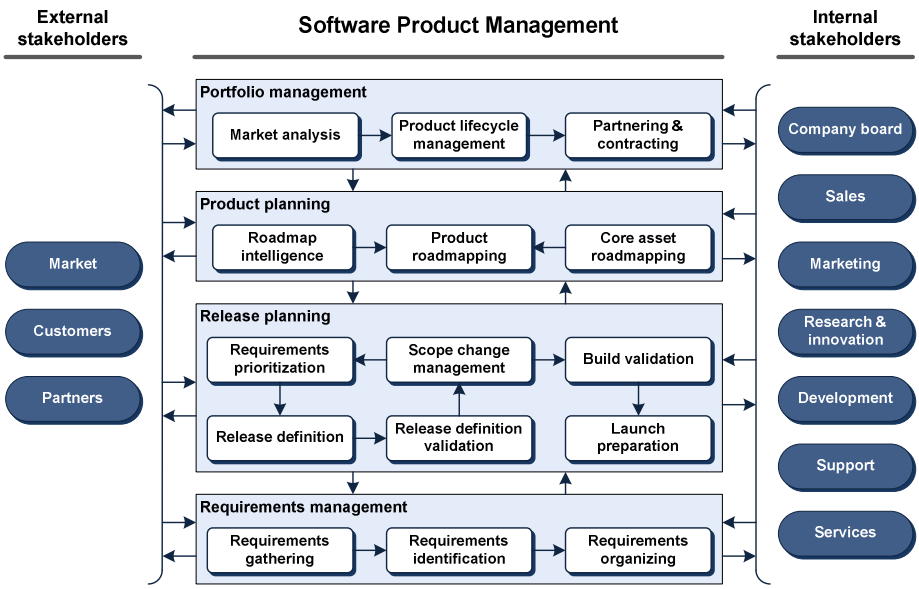

Fig. 1. The Software Product Management Competence Model

3.2 Focus areas

The business function *Requirements management* comprises the continuous management of requirements that are not (yet) part of a release and consists of three focus areas. *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 organizing* structures the requirements throughout their entire lifecycle based on shared aspects, and describes the dependencies between Product Requirements.

Release planning covers the SPM capabilities needed to successfully create and launch a release. *Requirements prioritization* prioritizes the identified and organized requirements. *Release definition* selects the requirements that will be implemented in the next release, based on the prioritization they received in the preceding process. It also creates a release definition based on the selection. *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. *Scope change management* handles the different kinds of scope changes that can occur during the development of a release. *Build validation* is performed after the release has been realized by the development department. It focuses on validating the built release before it is launched. *Launch preparation* prepares the internal and external stakeholders for the launch of the new release. Issues ranging from communication, to documentation,

training, and the preparations for the implementation of the release itself are addressed.

Product planning is focused on the gathering of information for, and creation of a roadmap for a product or product line and its core assets. It consists of three focus areas: *Roadmap intelligence* gathers decision supporting information needed in the creation of the product roadmap. *Product roadmapping* deals with the actual creation of the product roadmap itself. *Core asset roadmapping* concerns the planning of the development of core assets (components that are shared by multiple products).

Portfolio management concerns the strategic information gathering and decision making across the entire product portfolio. Its first focus area is *Market analysis*, which gathers decision support information about the market needed to make decisions about the product portfolio of an organization. Secondly, *Product lifecycle management* concerns the information gathering and key decision making about product life and major product changes across the entire product portfolio. Finally, *Partnering & contracting* focuses on establishing partnerships, pricing, and distribution aspects in which the product manager is involved.

4 The Software Product Management Maturity Matrix

4.1 Introduction

The maturity matrix is a key component of the Situational Assessment Method (SAM) for Software Product Management [8]. It is structured based on the SPM Competence Model introduced in section 3 and presents all of the important practices – called capabilities – in a best practice order for implementation, so that organizations have a guideline for the improvement of those SPM practices. Organizations can thus identify areas of improvement by comparing their organization's processes to the capabilities in the SPM Maturity Matrix. Based on the best practice order provided by the maturity matrix, companies can plan the improvement of their processes.

The maturity matrix depicted in Table 1 is a Focus Area Maturity Model [18, 19]. We chose to develop this type of maturity model because of the shortcomings of other existing models described in [20] and to enable local analysis and incremental improvement. Focus area maturity models are successfully used in the testing domain [21] and the architecture domain [19].

A focus area maturity model consists of a number of focus areas, each with its own number of specific maturity levels. The focus areas are represented in the leftmost column in Table 1. The focus area specific maturity levels are represented by the letters A-F in Table 1 and range from maturity level 1 to 10 (the topmost row in Table 1). Their spread across the overall maturity levels indicates a best practice order, in which capabilities in the maturity matrix are implemented left to right.

The development steps of a Focus Area Maturity Matrix in general, and the maturity matrix specifically are discussed in length in [14].

	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

Table 1. The Software Product Management Maturity Matrix

4.2 Capabilities

This section shortly describes all capabilities of the SPM Maturity Matrix by showing their title and the action required of the SPM organization. The capabilities have more aspects besides the title and action, but these were left out due to the limited space available in this paper. These additional aspects provide supporting information for each capability, consisting of the following: The goal to be achieved by possessing the capability; references to related literature supporting the SPM organization in the implementation and understanding of the capability; and the capability's prerequisites, these are the capabilities that need to be achieved before the capability in question can be achieved. The rest of this section is an overview of all 68 capabilities in the maturity matrix.

Requirements gathering: (A) *Basic registration* – Requirements are being gathered and registered. (B) *Centralized registration* – All incoming requirements are stored in a central database, which is accessible to all relevant stakeholders. (C) *Automation* – All incoming requirements are automatically stored in a central database (e.g. by means of an online helpdesk). (D) *Internal stakeholder involvement* – Requirements are gathered from all relevant internal stakeholders: support, services, development, sales & marketing, research & development. (E) *Customer involvement* – Customer's and prospect's requirements are being gathered and registered, and the customer or prospect is informed of the status of their requirements. (F) *Partner involvement* – Requirements are systematically gathered from partner companies.

Requirements identification: (A) *Uniformity* – Market Requirements are rewritten to Product Requirements using a pre-defined template if the Market

Requirement is applicable to a product. (B) *Requirements validation* – 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. (C) *Connect similar requirements* – Market Requirements that describe similar functionality are grouped together by linking Market Requirements and Product Requirements to each other. (D) *Automatically connect similar requirements* – Similar requirements are automatically connected by using advanced techniques such as linguistic engineering.

Requirements organization: (A) *Requirement organization* – Product requirements are organized based on shared aspects (e.g. type, function, or core asset). (B) *Requirement lifecycle management* – A requirement's history is logged by recording the submitter, submission date, changelog, original description, current status (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 built, so that it can be reused in a new or related product. (C) *Requirement dependency linking* – 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. This linkage can be supported by using advanced techniques, such as linguistic engineering.

Requirements prioritization: (A) *Internal stakeholder involvement* – All relevant internal stakeholders are involved in prioritizing the requirements that should be incorporated in future releases. (B) *Prioritization methodology* – A structured prioritization technique is used (e.g. MOSCOW, Wiegers). (C) *Customer involvement* – 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. (D) *Cost revenue consideration* – 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). (E) *Partner involvement* – Partner companies indicate the requirements that should be incorporated in future releases by assigning priorities to the requirements from their point of view.

Release definition: (A) *Basic requirements selection* – During requirements selection for the next release, constraints concerning engineering capacity are taken into account. (B) *Standardization* – 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. (C) *Internal communication* – The release definition is communicated to the internal stakeholders. (D) *Advanced requirements selection* – The optimal release is automatically calculated based upon the constraints of the requirements. At least the engineering capacity, priorities, cost, requirement dependencies are all taken into account. (E) *Multiple releases* – Multiple releases are included in the requirements selection process.

Release definition validation: (A) *Internal validation* – The release definition is checked by internal stakeholders, before the software is realized. (B) *Formal approval* – Approval standards are determined and verified by the board before the software is realized (turned over to development). (C) *Business case* – A business case (including the ROI) is being written before the software is realized.

Scope change management: (A) *Event notification* – A formal scope change management process is in place, in which all involved stakeholders are informed. (B) *Milestone monitoring* – Key dates and checkpoints are monitored in the product delivery. (C) *Impact analysis* – The impact of problems is determined, and involved stakeholders are informed of the impact. (D) *Scope change handling* – A process is in place to develop alternative plans, with all relevant stakeholders, to react to the effects of the scope change.

Build validation: (A) *Internal validation* – Internal stakeholders perform a functional validation of the build release to verify that it meets the expected outcome. (B) *External validation* – The build is validated by external parties (customers, partners) to verify the builds quality (e.g. by settings up a pilot). (C) *Certification* – Certification by an independent external party is acquired for the release.

Launch preparation: (A) *Internal communication* – 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. (B) *Formal approval* – A formal 'go', based upon standard quality rules, must be obtained from the board before the launch can begin. (C) *External communication* – 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. (D) *Training* – Trainings are organized and documentation is updated for both internal parties) and external parties to help educate them in the new release. (E) *Launch impact analysis* – The time needed to implement the new release at the individual customers is determined, and what type of experts are needed to perform the implementation (e.g. database experts). (F) *Update external expressions* – A checklist of all external expressions 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 is created. The items are checked, and possibly updated before they are made available to external parties (e.g. customers, partners).

Roadmap intelligence: (A) *Product analysis* – The organization's product is analyzed to determine the product's strong and weak points on both functional and technical aspects. Relevant stakeholders, such as the development department for the technical part, are involved in this analysis. (B) *Society trends* – 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 a product's industry. (C) *Technology trends* – 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 the product's market. (D) *Competition trends* –

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 competitors are shown, and the developments of the most important competing products are depicted with special attention. (E) *Partner roadmap* – An overview is created showing what an organization's 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 the organization can or will need to use to support the partner products/components.

Core asset roadmapping: (A) *Centralized registration* – All core assets are registered in a standardized manner, and are stored in a central location. (B) *Core asset identification* – Core assets are systematically identified among the organization's products and deliverables surrounding the product. (C) *Make or buy decision* – A process is in place to actively investigate make-or-buy decisions. This also includes the decision to outsource or subcontract development. (D) *Core asset roadmap construction* – A roadmap is created for the core assets, which shows how they are sustained, upgraded, and enhanced. This roadmap contains both existing core assets, and core assets that are in development.

Product roadmapping: (A) *Short-term roadmap* – A roadmap is developed detailing the short-term plans. (B) *Internal consultation* – Roadmaps are created in consultation with internal stakeholders. (C) *Theme identification* – Release themes are identified and maintained. Themes are decided on together with the internal stakeholders. This results in a list of release themes that is stored centrally, so that requirements, core assets, market trends etc. can be linked to it. (D) *Long-term roadmap* – The roadmap spans a time period of at least four years. (E) *Customer variant* – A (less detailed) variant of the roadmap is created for external parties (e.g. customers, partners).

Market analysis: (A) *Market trend identification* – There is an active search for market opportunities to either expand existing products, or create new products. In this search, market research is carried out in markets related to or similar to the organization's markets, conferences are visited, customers are interviewed, etc. All search findings are documented. (B) *Market strategy* – A plan is created showing which markets will be pursued and products for each segment can be developed. E.g., in year one, a company might plan to enter the automotive market by partnering with another company, or it may want to enter the pharmaceutical market in year two by building products in-house or acquiring products. (C) *Customer win/loss analysis* – A win/loss analysis is performed to research why customers chose or did not choose to buy an organization's products. This capability looks further than just the product features, e.g. the sales process is reviewed. (D) *Competitor analysis* – A competitor analysis is performed on an organizational level to analyze what competitors offer, what their strengths are and what they are going to offer, compared to the own organization. (E) *Custom market trend identification* – External market research parties are used to perform a market analysis specifically for the organization's product portfolio.

Partnering & contracting: (A) *Service level agreements* – (Standard) service level agreements (SLA's) are set up for customers. (B) *Intellectual property management* – Measures are in place to protect the intellectual property of the own organization, and to manage the used intellectual property from other organizations. (C) *Investigate distribution channels* – A process is in place to periodically verify the current distribution channels, and identify alternative distribution channels. (D) *Establish and evaluate pricing model* – A process is in place to establish the pricing model and periodically verify whether it still fits the market. (E) *Monitored Partner network* – A monitored 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.

Product lifecycle management: (A) *Product life cycle analysis* – The current life phase is determined, at least once per year, for each product in the organization's portfolio. This analysis is based on both financial and technical aspects. Information is gathered from all relevant internal stakeholders. (B) *Portfolio innovation* – A decision process is in place to decide whether or not to incorporate trends in the organization's products, and whether to incorporate these trends in the current products or in future products. (C) *Portfolio scope analysis* – A product scope analysis is performed to identify overlaps and gaps between the products in the organization's product portfolio. (D) *Business case* – A business case is performed for major product revisions. (E) *Product lines* – 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.

5 Conclusion and future research

5.1 Conclusion

We believe the Software Product Management Competence Model and the Software Product Management Maturity Matrix presented in this paper to be a solid basis for Software Process Improvement. Our iterative process of improvements and validations in both the field of practice and the scientific community makes this a broadly accepted model.

The organizations which participated in our research all indicated that they have a great need for a model which can be applied in practice at relatively low costs to improve their SPM processes. This paper provides in this need by presenting a model with descriptions for all of the most important SPM practices in SPM organizations. Our case study organizations found the SPM Competence Model and SPM Maturity Matrix presented here very useful models to structure and improve their SPM processes.

5.2 Future research

The models presented in this paper form a base for the assessment method presented in [8]. Our experiences during the case studies made clear that the SPM organizations both need and want such an assessment method. We therefore intend to expand the SAM to a quality instrument with which organizations can periodically evaluate their improvements and set new goals to further improve their maturity.

Further research into the effects of Situational Factors [20] on the capabilities is also very useful. Not all capabilities are relevant to every type of organization. It is therefore important to map which capabilities are relevant to the different types of organizations.

The field of SPM is closely related to development, project management, marketing, and sales. It can be hard to define where SPM ends where another area starts. In many cases, there is cooperation between the product managers performing the SPM activities, and the managers performing the related tasks. We incorporated all activities in which the product managers have a substantial participation. Further research to define the responsibilities in the grey areas would be useful.

The model presented in this paper was developed based on Dutch experts from product software organizations and the scientific community. It might therefore not be fully applicable to organizations outside of the Netherlands. Further international validations must therefore be performed to check the general applicability.

6 References

1. Fricker, S., Gorschek, T., Byman, C., & Schmidle (2010), A. Handshaking: Negotiate to Provoke the Right Understanding of Requirements. *IEEE Software 26*(6).
2. P. Berander, "Evolving Prioritization for Software Product Management," in APS, PhD thesis, Ronneby: Bleking Institute of Technology, 2007
3. Christof Ebert, Sjaak Brinkkemper, Slinger Jansen, Gerald Heller, "2nd International Workshop on Software Product Management," International Workshop on Software Product Management, pp. i-ii, 2008 Second International Workshop on Software Product Management.
4. Weerd, I. van de, Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). Towards a reference framework for software product management. *Proceedings of the 14th International Requirements Engineering Conference*, Minneapolis/St. Paul, Minnesota, USA, 319-322.
5. Project Management Institute. A guide to the project management body of knowledge (PMBOKGuide), 2000 ed. Newtown Square, PA: Project Management Institute; 2000.
6. Abran, A., Moore, J. W., Bourque, P., Dupuis, R., & Tripp, L. L. (2004). *SWEBOK: Guide to the Software Engineering Body of Knowledge.* Los Alamitos, CA: IEEE Computer Society.
7. Artz, P., Weerd, I. van de, & Brinkkemper, S. (2010). Productization: transforming from developing customer-specific software to product software.

Proceedings of the 1st International Conference on Software Business (ICSOB 2010), LNBIP 51, 90–102.

8. Bekkers, W., Spruit, M., Weerd, I. v., & Brinkkemper, S. (2010). A Situational Assessment Method for Software Product Management. *Accepted for the 18th European Conference on Information Systems (ECIS2010)*, June 7-9, Pretoria, South Africa
9. Krzanik, L., & Simila, J. (1997). Is my software process improvement suitable for incremental deployment? *Proceedings of the 8th International Workshop on Software Technology and Engineering Practice*. London, UK, 76-87.
10. Weerd, I. van de, Brinkkemper, S., & Versendaal, J. (2010). Incremental method evolution in global software product management: A retrospective case study. *Accepted for publication in the Journal of Information & Software Technology*.
11. Weerd, I. van de, Versendaal, J., & Brinkkemper, S. (2006). A product software knowledge infrastructure for situational capability maturation: Vision and case studies in product management. *Proceedings of the 12th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'06)*, Luxembourg, 97-112.
12. Hevner, A., March, S., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28, 75-105.
13. Vaishnavi, V., Kuechler, B. (2009, August 01). Design Research in Information Systems. Retrieved Mach 16, 2010, from AISWorld Net: http://www.isworld.org/Researchdesign/drisISworld.htm
14. Weerd, I. van de (2009). *Advancing in Software Product Management: An Incremental Method Engineering Approach*. Doctoral dissertation, Utrecht University, The Netherlands.
15. Abramovici, M., & Soeg, O. (2002). *Status and Development Trends of Product Lifecycle Management Systems*. Germany: Ruhr-University Bochum, Chair of IT in Mechanical Engineering (ITM).
16. Clements, P., & Northrop, L. (2001). *Software Product Lines: Patterns and Practice*. Reading, MA: Addison Wesley, Boston, MA.
17. Weerd, I, van de, Bekkers, W., Brinkkemper, S. (2010). Developing a maturity matrix for software product management. *Proceedings of the 1st International Conference on Software Business (ICSOB 2010), LNBIP 51*, 76–89.
18. Steenbergen, M., Bos, R., Brinkkemper, S., Weerd, van, I., Bekkers, W. (2010). The Design of Focus Area Maturity Models. *Accepted for publication in 5th International Conference on Design Science Research in Information Systems and Technology (DESRIST2010)*.
19. Steenbergen, M., & Brinkkemper, S. (2007). An instrument for the Development of the Enterprise Architecture Practice. *Proceedings of the 9th International Conference on Enterprise Information Systems*, (pp. 14-22).
20. Bekkers, W., Weerd, I. van de, Brinkkemper, S., Mahieu, A. (2008). The Influence of Situational Factors in Software Product Management: An Empirical Study. *Presented at the 21th International Workshop on Software Product Management (IWSPM2008),* Barcelona, Spain, September 9, 2008.
21. Koomen, T., & Baarda, R. (2005). TMap Test Topics. Nederland: Tutein Nolthenius.