# A Reference Framework for Software Product Management

*Inge van de Weerd, Sjaak Brinkkemper, Richard Nieuwenhuis, Johan Versendaal, Lex Bijlsma*

Department of Information and Computing Sciences
Utrecht University
Technical Report UU-CS-2006-014
www.cs.uu.nl
ISSN: 0924-3275

# A Reference Framework for Software Product Management

*Inge van de Weerd, Sjaak Brinkkemper, Richard Nieuwenhuis, Johan Versendaal, Lex Bijlsma*

Department of Information and Computing Sciences
Utrecht University, The Netherlands
{i.vandeweerd, s.brinkkemper, rnieuwen, j.versendaal, a.bijlsma}@cs.uu.nl

## Abstract

*In the last decade, software product management has received much practical attention, though research in this area is still scattered. In this paper, we give a status overview of the current software product management domain by performing a literature study and field studies with product managers. The results are used to develop a reference framework for software product management, in which the key process areas, stakeholders and their relations are modeled. To validate the reference framework, we perform a case study in which we analyze the stakeholder communication concerning the conception, development and launching of a new product at a major software vendor. Finally, we propose the Software Product Management Workbench for operational support for product managers in product software companies.*

## 1. Product management

Software is more and more developed and commercialized as a standard product. In companies specializing in software products, the role of product manager has emerged over the last years, and appears to be of strategic value, but complex to execute. The product manager is responsible for managing requirements, defining releases, and defining products in a context where many internal and external stakeholders are involved [15] [46]. The domain of product management has been established, especially in technical sectors with physical products, since the industrial revolution in the 19th century [29]. Only relatively recently, also software product management has received attention in product software companies like Microsoft [18] and Alcatel [21] [33], and, to a lesser extent, in scientific literature, e.g. [29]. Although several of the existing product management practices can be applied in software product management, specific challenges can be identified in software product management. Software products differ from other products in the fact that the manufacturing and distributing of extra copies do not require extra costs for the company [17]. Also, existing software products can be changed easily, and sold software products can be updated by using patches or release updates. There is also a downside to these advantages. The organization of requirements and the tracking of changes in the design are very complex. Also, due to the ease of making changes, the release frequency is relatively high in comparison with non-software products. Finally, the product manager has a lot of responsibilities regarding the product functionality, but has not the management authority over the development team, so decision making requires consent of many players. We therefore claim that there is a need for an integration of research efforts in this key domain.

Knowledge on software product management for research and educational purposes is very fragmented. In a few (software) product management areas some know-how is available, but there still lacks an integrated body of knowledge, as exists in software development [10] and project management [37]. The goal of this paper is to develop a (preliminary) body of knowledge for software product management, by providing a reference framework for all its activities and deliverables. This reference framework has been based on an extensive overview of state-of-the-art literature, industrial case studies, and by exploring opportunities for operational tool support.

The organization of the paper is as follows. In the next section we elaborate on the rationale for the reference framework, and the research method we have applied to develop it. Then, in section 3, we discuss the basic structure of the reference framework. The four process areas are elaborated on in section 4. In section 5, describe a case study at a major Enterprise Resource Planning software vendor. Subsequently, in section 6, we describe the Software Product Management Workbench, for operational tool support for the product manager. The final section describes our conclusions and future research.

## 2. Rationale and research method

Reference frameworks have proven to be beneficial for research and practice in many fields. We mention the ISO/OSI layers for the layering of network services [26] or the ANSI/SPARC 3-schema architecture for database

management systems [45]. The need for a reference framework for software product management is essentially found in the desire to get an understanding of its complete domain. Varying research contributions on the one hand and all kinds of developments in the software industry on the other hand can be positioned in such a reference framework to interpret their consequences in a uniformed context. Furthermore, such a reference framework also provides a starting point for:

- Definition of key terms in software product management and the identification of open research questions;
- Education of product managers and competence building;
- Development of improved, integrated tool support;

The research method we have applied for the conception of the reference framework is the following:

1. Field interviews and discussions with experienced product managers;
2. Literature review on both non-software product management as well as on software product management;
3. Creation of a draft reference framework;
4. Validation by an extensive case study at a large product software company;
5. Validation with input from an industrial workgroup on product management;
6. Finalization of the reference framework.

The draft framework was amended several times after comments and suggestions from both practitioners as well as researchers. We do not claim that we now have produced the definitive version of the reference framework. Small augments might still be needed, but we are convinced that the basic structure has been established. The framework served furthermore as input for the design of the architecture of the product management workbench.

## 3. Basic framework structure

The basic structure of the reference framework is based upon the objects or artifacts of product management, and upon the set of stakeholders identified in the scope of work of the product manager.

### 3.1. Artifact hierarchy

Professional product management is essentially a matter of well-organized processing of issues related to requirements, products and releases [19] [15]. A hierarchical ordering of these artifacts (see Figure 1) imposes a structure on the process areas.

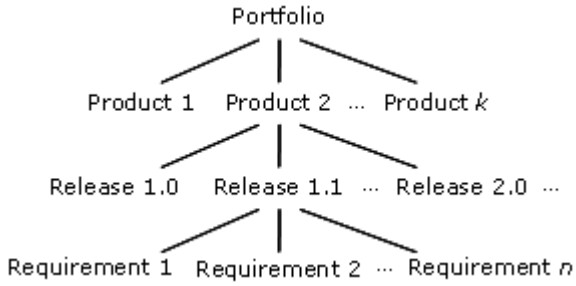


**Figure 1. Artifact hierarchy of product management**

First, the scope of work of product management concerns the complete set of products of the company, the so-called *product portfolio*. Small or young companies may have a portfolio of just one product, whereas larger companies have several, due to acquisitions and/or product derivation.

Each product has a release sequence of past, present and future releases. Several internal versions may exist versus selected externally released versions for the market. The release numbering is usually determined by internal conventions, where major changes in the technical architecture are a reason to call it an X.0 release. Marketing reasons may lead to commercial numbering using the year of release or the same release code as an important customer.

Finally, each release definition consists of a set of selected requirements. Each requirement implies the addition of a technical or functional feature to the product. Non-functional requirements are also considered, such as performance constraints or availability requirements.

As the type of work differs when dealing with artifacts from the distinct hierarchy levels, this hierarchy gives rise to a subdivision of software product management into four process areas: *portfolio management* to deal with the products in the product portfolio; *product roadmapping* to deal with the different releases of each product, also called roadmapping; *release planning* to deal with the collections of requirements of each release; and *requirements management* to deal with the content and administrative data of each individual requirement.

Observe however, that for the sake of diagram clarity, we have swapped the positions of requirements management and release planning in the reference framework (Figure 2). Release planning processes communicate about complete releases to internal stakeholders, whereas requirements management interacts with all stakeholders.

### 3.2. Stakeholder interaction

Product managers are confronted with a large number of requirements, originating from different internal and external stakeholders. We distinguish the following internal stakeholders [15] [19]:

- The *Company board* is responsible for the definition and communication of strategy, vision and mission to the rest of the company. Also, it has the managerial supervision of the different departments, including product management. Occasionally, requirements are communicated through its strategy, but it can occur that a requirement is sent directly to the product manager.
- *Research & innovation* has two core responsibilities: (1) doing research to new opportunities for product innovations and (2) finding ways to incorporate improvements or new features into the existing products. The first one results in requirements in the form of technology drivers that are communicated to the product manager.
- The consultants of the S*ervices* department are responsible for the implementation of the software product at the customer organization. They need to be aware of new release features and they gather new requirements from the customers.
- *Development* has as main responsibility the execution of the release plan. The release definition also includes functional explanation of the product requirements that serve as input for the functional and technical design. It may occur that during the development process new requirements can arise, due to more complex requirements than was anticipated.
- *Support* stands for the helpdesk to answer questions (1st line support) and for small defect repair unit (2nd line support). Large defect repair is usually performed by Development (3rd line support).
- *Sales & marketing* is the first contact with a potential customer. Through these contacts new requirements can be gathered.

The following external stakeholders are recognized [32]:

- The *Market* is an abstract stakeholder, standing for potential customers, competitors and analysts, such as Gartner and Aberdeen. Numerous trends may be recognizable in the market, either in an explicit way by one of the market players, or in an implicit way by product management.
- Most companies have different kinds of *Partners*: (1) implementation partners, who implement the product at a customer; (2) development partners, with whom product components are developed; and (3) distribution partners, selling the product.
- *Customers* often have new feature requests in the process of closing the deal or during the usage of the product. These requests can be communicated to Services, Sales & marketing, Support, but also directly to the product manager.

Observe that the stakeholder names are generic, so that naming or grouping may differ in product software companies. It is obvious that external stakeholders are harder to be influenced in their operational execution and decision making, whereas internal stakeholders should act according to the corporate strategy.

## 4. Reference framework

Except in requirements engineering, there is little literature explicitly addressing the domain of software product management. Vähäniitty [48] found that product portfolio management is largely overlooked in literature, and if it is addressed, it does not mention small and medium sized product software companies. Some papers address release planning [11] [28] [42]. Although software development is largely addressed it adheres to project-related development [24]. In this section we provide an overview of state-of-the-art research on (software) product management.

In Figure 2, a reference framework for software product management is visualized. In the remaining of this section, each of the process areas, defined in section 3.1, is provided with an explanation supported by research contributions.

### 4.1. Portfolio management

Portfolio management entails the decision making about the set of existing products, introducing new products by looking at market trends and the product development strategy, making decision about the product lifecycle, and establishing partnerships and contracts. We also position product line management in this area. A software product line is 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 [14]. Several case studies have shown that introducing product lines organizations improves performance [8] [9] [44]. They are most popular in telecommunication organizations [33], but the last years, also the software industry pays more and more attention to this topic [1] [4] [14]. Some research has been done to tool support for product lines. An example is Laqua [30], who proposes a product line content & knowledge base on top of arbitrary configuration management system. Product lifecycle management is a comprehensive approach for product-related information and knowledge management within an enterprise, including planning and controlling of processes that are

required for managing data, documents and enterprise resources throughout the entire product lifecycle [1]. This is a key process in decision making about the product portfolio. Also partnering and contracting are important issues in product management [7].

Looking at the reference framework, we see Portfolio management on top. It contains four main processes: partnering & contracting, market trend identification, product lifecycle management and product line identification. Input is received from the Company board, Market and Partner companies.

## 4.2. Product roadmapping

In [47], roadmapping is called a popular metaphor for planning and portraying the use of scientific and technological resources, elements and their structural relationships over a period of time. It is complex due to dependencies on other related products (even from partners), technology changes, and the distributed development [13]. Roadmapping has, just like product line management, its origins in a sector distinct from the software industry, the manufacturing industry, where it is used for business oriented long-term planning and technology forecasting [32], however, also in the product software industry roadmaps are used for planning purposes [47]. In [27] the term roadmapping is used in two perspectives: forecasting and planning. Forecasting concerns technology or market trends; and planning concerns products, product lines, resources or the entire company. We use the definition of [39]: a roadmap is a document that provides a layout of the product releases to come over a time frame of three to five years. It is written in terms of expectations, plans and themes and core assets [34] of the product.

In the reference framework, product roadmapping receives input regarding product lines from portfolio management. This input is used to identify themes and core assets that can be used later on in the requirements organization. This information is gathered and described in the product roadmap.

## 4.3. Requirements management

Requirements management entails the activities of gathering, identifying and revising incoming requirements and organizing them by keeping in mind mutual dependencies, existing core assets, product lines and themes. Sources are customers, sales and marketing, development, support, R&D and the company's management.

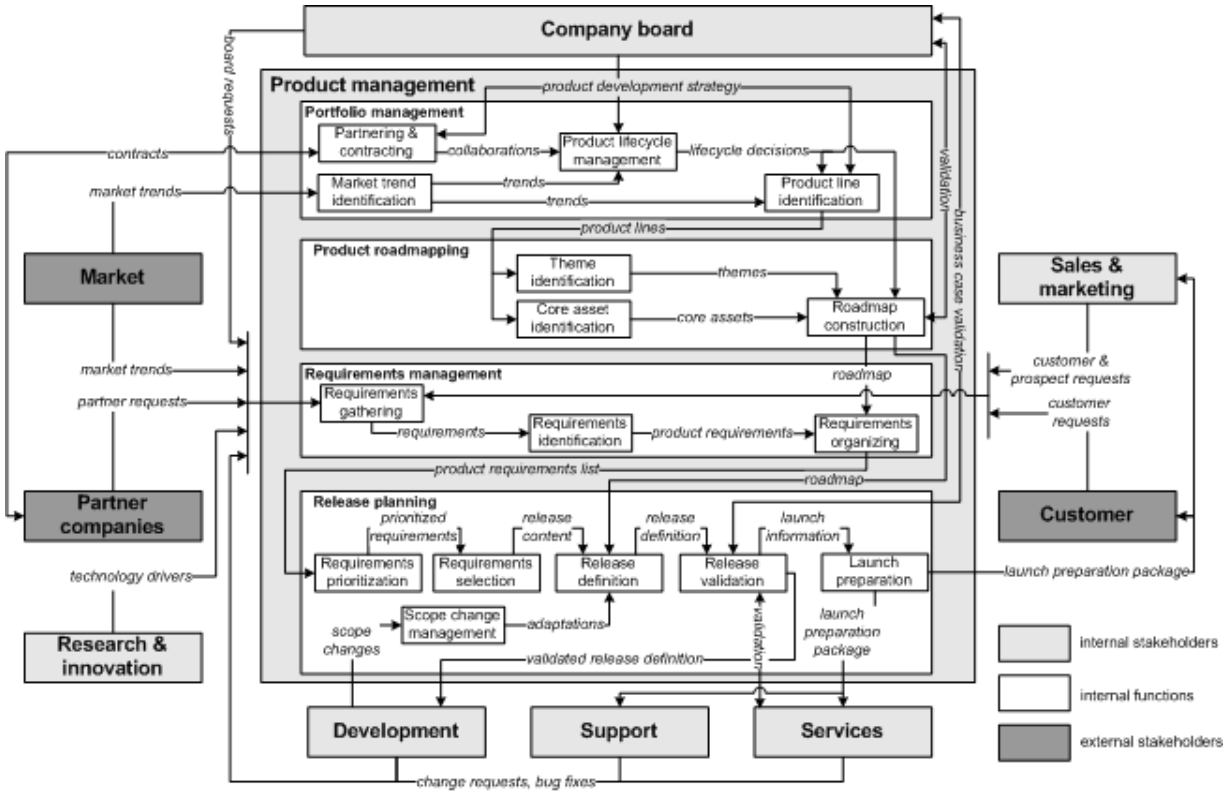


**Figure 2. Reference framework for software product management**

Requirements management is a key area in product software companies [12], but [38] already recognized that requirements engineering for product software is different than for customized software. In [36], the following core requirements engineering activities are recognized: eliciting requirements, modeling and analyzing requirements, communicating requirements, agreeing requirements, and evolving requirements. Especially analyzing requirements costs a lot of time in product software companies, due to the (often) high requirements rate, and the different sources of requirements. An example is the use of linguistic engineering to link customer wishes to requirements [35]. Another problem is the integration of a software product with other systems. Customers cannot expect that all their requirements are met, which may lead to a software product that does not integrate with their existing systems. In [31] several improvements are suggested to this practice. In [20], the requirements process in 246 industry projects is investigated and the results show that four techniques improve schedule performance, if used in parallel: installing of an effective core team for each product release; focusing on the product-lifecycle on upstream gate reviews; evaluating requirements from various perspectives; and assuring a dependable portfolio and release planning implementation.

The position of requirements management in the reference framework is between product roadmapping and release planning. The process starts with gathering all requirements from within the company and from external stakeholders. The requirements gathered and organized into product requirements. Product requirements are identified by removing the duplicates, connecting requirements that describe a similar functionality, and by rewriting the requirements in understandable product requirements. Then, the requirements are organized per product and core asset. Also, the mutual dependencies between the different product requirements are described. In [35], a distinction is made between *market requirements*, which refer to wishes related to future products, defined in the customer's perspective and context; and *business requirements*, a product requirement to be covered by the company's products, described in the company's perspective and context. We use a similar distinction. However, we make a distinction between *requirements* and *product requirements*. Requirements refer to all incoming wishes and change requests. This are not only market requirements, but also service requirement, board requests, technological drivers form research & innovation, etc.

### 4.4 Release planning

Software release management is the process through which software is made available to, and obtained by, its users [25]. Core functions in this process are requirements prioritizing; release planning; constructing and validating a release requirements document; and scope management

Especially on the area of release planning, where the set of requirements for the next release is determined, much research has been carried out. Examples are release planning using integer linear programming [1], the analytical hierarchy process [41], stakeholders' opinions on requirements importance [40] and linear programming techniques using requirement interdependencies [11]. More techniques can be found in [2] and [5].

In the reference framework, release planning starts with the product requirements prioritization. Not only the product management is responsible for this, but also the other stakeholders can influence this process. After the prioritization, product requirements are selected that will be implemented in the next release. This can be done in multiple ways: one can choose the product requirements with the highest priority or use integer linear programming to estimate the best set of requirements. During this process, also the resources have to be applied in the calculations. When the product requirements are selected, a release definition is written that is validated by different stakeholders. A business case is sent to the company board. When this has been approved by the board, a launch preparation package is constructed and sent to the stakeholders.

## 5. Case study

In finding confirmation for the validity of the identified context, activities and relations depicted in the reference framework, we analyzed the conception, development and launching of a new product at a major Enterprise Resource Planning (ERP) software vendor during the period September 2000 to June 2002. The responsible product manager at this company provided us with all incoming e-mail traffic regarding this new product as a source for our analysis. In the mentioned period the product manager received about 1,200 emails related to this product.

### 5.1 ERP vendor case

After an organizational repositioning, the management board of the ERP vendor decided to focus on providing add-on products, so-called solutions, next to ERP products. So, from September 2000 onwards, an

integrated procurement product was planned, including direct materials purchasing, indirect materials purchasing, e-procurement, e-invoicing and e-kanban, to be integrated via one Supplier Trading eXchange (STX). Note that at the start, some of the functionality was already available in existing products (e.g. direct materials purchasing in the ERP-product, e-procurement in the E-Procurement product), while other functionality needed to be created. Existing and new functionality needed to be disclosed through STX.

As for portfolio management, a number of e-mails represented the assignment of solutions, including the STX solution. Although the board indicated (based on market signals) the necessity of solutions, the product manager verified the need for a specific procurement solution through industry analysts, important customers and competitor analysis. Specifically the successful implementation at Komatsu of a predecessor application of the STX, i.e. the E-Collaboration tool, encouraged the product manager to further prepare development of the STX. In one of the e-mails the product manager was invited by someone from the ERP vendor's consultancy department to attend a knowledge transfer on E-Collaboration based on of the successful implementation at Komatsu: "I spoke with Komatsu today just to see how things are going and to ask permission to access their site tomorrow for a knowledge transfer session that I am doing for some of the consulting folks and Sales Managers; you are most welcome to call-in". Note that this particular implementation has been described in a case study in a separate paper [49].

A potential partner company was approached to further enhance functionality regarding the so-called 'round-trip' requisitioning (i.e. linking into suppliers' item catalogues at the suppliers' websites in order to purchase goods from suppliers' sites directly). Integration between the partner's product and STX would make this possible, as one of the e-mails states: "Supplier's product information is dynamically available through agent technology in the partner product's Java code".

Regarding product roadmapping, it became clear that not all topics and themes for an (according to the product manager) ideal procurement solution through the STX could be covered in one release. An example was the late discussion of e-kanban (a Just-In-Time purchasing strategy solution) and its incorporation in the future: in one of the e-mails the product manager asked a colleague to "provide me with some compelling arguments why it is good to develop E-Kanban in STX from the business perspective". In general, in many e-mails dealt with themes projection over future anticipated releases of the STX. This included communication with the management board of the company.

Many of the 1,200 e-mails dealt with requirements management and release definition. A number of detailed requirements became clear from the previous implementation at Komatsu. In addition, communication with the support and consultancy departments provided other requirements for the STX. At the end of 2000, an early version of a release definition was communicated with a number of internal departments, including marketing & sales, development, and the release management department. Later on, the architect of the development department interpreted the requirements in a functional design document: "Here is the first draft of functional design document" (e-mail of 9 March 2001). Subsequent e-mails from the development department mainly dealt with requirement clarification ("I need clarification about the off-line purchase in the STX") and scope changes ("shouldn't we support RosettaNet message exchange?").

In cooperation with other departments and associated country organizations the product manager prepared the launch of the STX: e.g. a white paper was written on the product with involvement of marketing and sales ("Sure thing! I'll make sure this is in the plan and we can work together to get it done".).

## 5.2. Case analysis

In the case study on STX we note that all main product management areas (portfolio management, product roadmapping, requirements management and release management) were addressed. Some areas and some topics within each of the areas were more subject in e-mails than others. For example, product lifecycle management in portfolio management was not so much addressed, as it concerned the first releases of STX, therefore roadmap construction was more extensively addressed. Also, requirements prioritization and selection was not addressed extensively, the scope of the STX, and the list of all requirements was rather small However, proposed scope increases were weighed carefully in order to either include or exclude them: the product manager had to balance between allowing scope creep for development and satisfying sales & marketing.

All identified stakeholders in figure 2 were extensively involved in the communication with the product manager, even for research & innovation: the development department prototyped the round-trip functionality with the STX's partner product.

The largest category of all the 1200 e-mails came from development. This can be explained by the fact that development took place in another country than the country of origin of the product manager. Much communication was through conference calls and e-mails.

## 6. The Software Product Management Workbench

Product management is key to product software companies and should be addressed and supported well. Although there are several tools supporting part of the product management functionality, they do not provide a coherent and complete set of features dedicated to software product management. To support the product manager, we propose the software Product Management Workbench.

### 6.1. Existing support tools

Several portfolio management support tools exist, e.g. ProSight's Application Portfolio Management, supporting top-down portfolio management solutions for a company, and UMT's Portfolio Manager Software Suite, a web-based application for portfolio management.

Few support tools for product roadmapping exists. ReleasePlanner [40] covers part of it. ReleasePlanner is a web-based system solution to enable intelligent planning, priority and road-mapping decisions.

Tools that focus especially on requirements management are Borland's CaliberRM for managing requirements throughout the software delivery process and IBM's RequisitePro, a requirements and use case management tool. ReqSimile [35] is a requirements tool that supports the linkage process in large-scale requirements management, by using a linguistic engineering approach.

Some tools exist in the release planning area. The Accept 360° platform form Accept Software is a product planning and delivery solution that addresses the spectrum of business requirements in all levels of the organization. ReleasePlanner [40], earlier mentioned in this section, is a uses integer linear programming and prioritization of features for purposes of release planning. This tool focuses on (but is not limited to) software companies. In the Release Planner Provotype [11] a selection algorithm is implemented that presents a number of valid and good release suggestions.

### 6.2. An integrated solution

To provide operational support for software product management, we propose a tool: the Software Product Management Workbench. As explained further, it supports portfolio management, product roadmapping, release planning and requirements management, in an integrated way.

The workbench is divided into four main modules, all intended to aid the product manager with his daily routines. The four modules are: *requirements module*, *release planning module*, *roadmap module*, and *product portfolio module*, their names corresponding to the functionality they provide.

The workbench is designed for different user types. The *product manager* is the main user, but there are three other users that are able to login into the system, all with their own privileges. These three users are: *administrator*, *core asset developer*, and *employee*. Product software companies usually have multiple software products all furnished with new releases every once in a while. The main task of the administrator is to start new products or new product releases. When a new core asset has been identified, the core asset developer can login into the system and add this new core asset to the system. In this way the product manager can use the core asset in defining a release, and the development team has always access to information on the latest core assets. An employee can logon to the system for reading the latest news of the development progress or report some news about his work on an upcoming release.

### 6.3. Architecture

The Software Product Management Workbench is a so-called *enterprise application*. Building enterprise applications is a hard and taunting task [23], because they deal with a lot of persistent data, concurrent data access, multiple users with different roles, and are built in a distributed way. In the workbench the difficulties are found in the great amounts of requirements that have to be persistent, different actors that can login into the system, and more. J2EE is a platform that enables the easy creation of enterprise applications, since J2EE handles all the difficult tasks described above for you. This means that enterprise programmers only have to deal with programming the business logic. For technical information of J2EE see [6]. In [43], Szyperski provides a thorough evaluation of the J2EE platform.

Figure 3 gives a high level overview of the architecture. The tool uses two types of clients: a *web client* and an *application client*. Application clients run on the client machine and offer the ability to perform heavy calculations on the client machine, without affecting the server. Enterprise Java Beans (EJB) form the core of the J2EE platform that makes the life of an enterprise programmer easier. Two types of EJBs are used in the architecture, namely *entity* and *session* beans. One entity bean represents one row in a database table (or a row in the result of a join operation). Two types of *session* beans exist, which are *stateful* and *stateless* session beans. A stateful session bean can maintain

conversational state for one client. A stateless session bean offers its services to multiple clients.

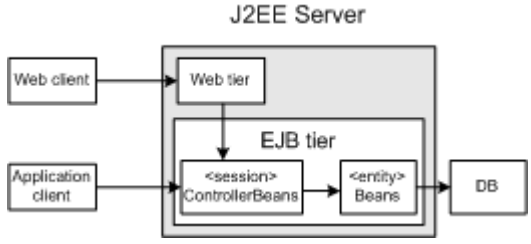


**Figure 3. High level architecture**

The response time, the amount of time it takes for the system to process a request from the outside, is of great importance [23]. The product manager uses functionalities of the tool that require a lot of processing time, so he is the only one able to login into the application client to execute these calculations. The web tier handles all the requests generated by the web client and directs these requests to the controller beans that are deployed into the EJB tier, which provide coarse grained access to the entity beans. The application client accesses the controller beans directly. Note that the web tier and the EJB tier do not have to reside on the same machine.

The extendibility of the tool is also an important issue. As mentioned before there are now four main modules, but the tool should be able to be extended with minimum effort to provide other kinds of functionality. Figure 4 shows a small part of the full architecture, but captures some of the patterns used.

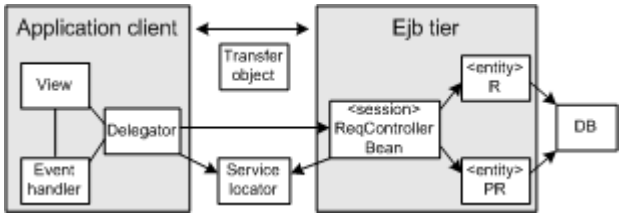


**Figure 4. Requirements administrator module**

The figure shows part of the requirements module, where incoming requirements are connected to product requirements. Remote calls and calls from the web tier to the EJB tier are relatively very slow, so this number should be minimized. The system uses transfer objects that capture as much data that the client possibly wants to get his hands on, instead of getting one piece of data at the time at the cost of one remote call every time. The different components of the system have to be located with so called “look-ups”. It is efficient to extract this code from all the components and put all the look-up code in a service locator object. In this way, references to components can be cached for other components that may need a reference to that component, minimizing the look-ups. There is no tight coupling between the components, which makes the tool easy to change. If for example the presentation logic has to be adapted, only the view has to be changed leaving the other components unharmed.

### 6.4. Prototype

Figure 5 shows a screenshot of the prototype of the Software Product Management Workbench. It shows the requirements window, in which the product manager can link requirements with product requirements that refer to the same functionality [35]. At the top of the screen, a list of product requirements is depicted. A product requirement can be selected in order to find matching requirements from the requirements list at the bottom of the screen. After the system has found all the possible candidates, the requirements are displayed together with the source, similarity ratio and the option to link this requirement to a product requirement. When the preferred requirements are selected, the linkage can be saved.

## 7. Conclusions and further research

In this article we discussed the difference between product management and software product management, and the need for operational support for the latter. By performing field interviews and discussions with product managers and by doing a literature review on (software) product management, we developed a reference framework for software product management. Furthermore, we provided an overview of state-of-the-art literature on software product management.

By performing a case study, we found confirmation of the validity of the identified context, processes and relations in the reference framework for software product management.

Finally, we proposed the Software Product Management Workbench, which integrates several software product management areas. This workbench is currently being developed. When it is finished, several industrial case studies will be performed to test the functionality.

We are convinced that the proposed reference framework for product software management is a first step to position this important industrial domain in the filed of scientific research in requirements engineering. In the future, we hope to contribute to further refinements of the reference framework and to its application in various domains.

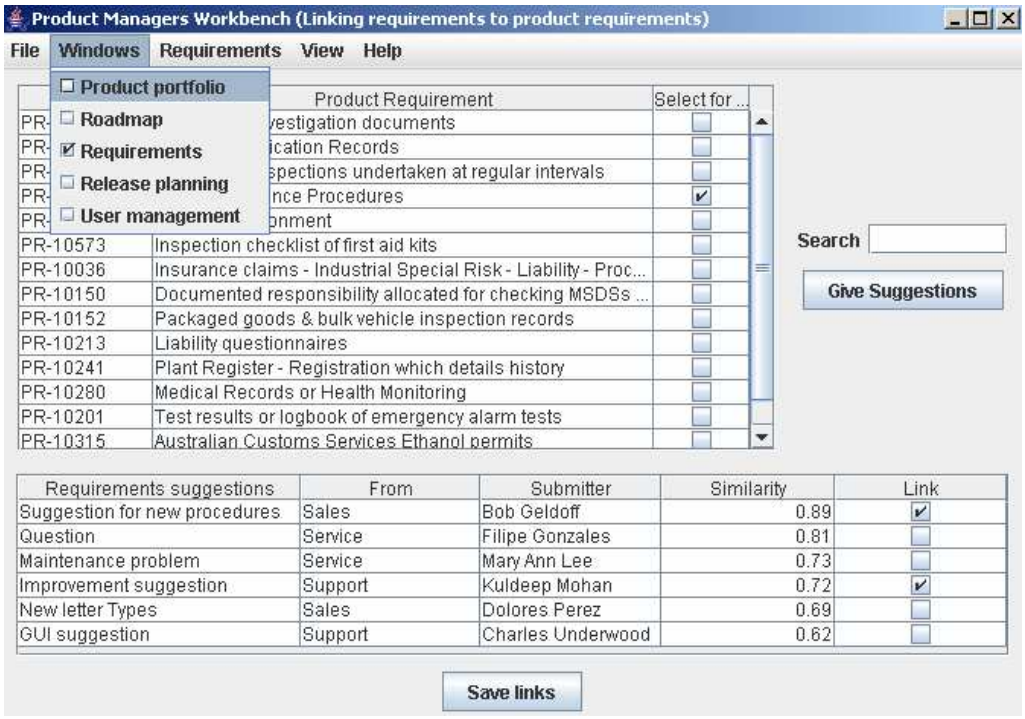


**Figure 5. Screenshot of the Software Product Management Workbench**

## References


[1] M. Abramovici and O.C. Soeg, *Status and Development Trends of Product Lifecycle Management Systems*, Ruhr-University Bochum, Chair of IT in Mechanical Engineering (ITM), Germany, 2002.

[2] M. van den Akker, S. Brinkkemper, G. van Diepen, and J. Versendaal, "Flexible Release Planning Using Integer Linear Programming", *Proceedings of the 11th International Workshop on Requirements Engineering for Software Quality (REFSQ'05),* 13-14 June 2005, Porto, Portugal, Essener Informatik Beitrage, Band 10.

[3] D. Alur, J. Crupi, and D. Malks, *Core J2EE Patterns*, Prentice Hall PTR, 2001.

[4] M. Ardis, N. Daley, D.M. Hoffman, H. Siy, and D. Weiss, "Software Product Lines: a Case Study", *Software - Practice and Experience*, 2000, 30(7), pp. 825–847.

[5] P. Berander and A. Andrews, "Requirements Prioritization", *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin (eds.), Berlin, Germany, Springer Verlag, 2005, pp. 69-94.

[6] S. Bodoff, D. Green, K. Haase, E. Jendrock, M. Pawlan, and B. Stearns, *The J2EE Tutorial*, Addison Wesley Professional, 2002.

[7] A. Bonaccorsi and A. Lipparini, "Strategic Partnerships in New Product Development: An Italian Case Study", *Journal of Production Innovation Management 11*, 1994, pp. 134–145.

[8] J. Bosch, "Product-Line Architectures in Industry: A Case Study," *ICSE Proceedings*, Los Angeles, CA, 1999, pp. 544-554.

[9] L. Brownsword, P. Clements, *A Case Study in Successful Product Line Development,* Technical Report CMU/SEI-96-TR-016, Carnegie Mellon, 1996.

[10] P. Bourque and R. Dupuis, (ed.), *Guide to the Software Engineering Body of Knowledge*, 2004 edition, IEEE Computer Society, Los Alamitos, California, USA, 2004.

[11] P. Carlshamare, "Release Planning in Market-Driven Software Product Development Provoking and Understanding", *Requirements Eng. 7*, 2002, pp. 139-151.

[12] P. Carlshamre, B. Regnell, "Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes", *11th International Workshop on Database and Expert Systems Applications (DEXA'00)*, 2000, p. 961.

[13] E. Carmel, *Global Software Teams,* Prentice Hall: Upper Saddle River, NK, 1999.

[14] P. Clements and L. Northrop, *Software Product Lines: Patterns and Practice*. Reading, MA: Addison Wesley, 2001.

[15] D. Condon, *Software Product Management,* Aspatore Books, Boston, 2002.

[16] R.G. Cooper, S.J. Edgett, and E.J. Kleinschmidt, "Portfolio Management for New Product Development: Results of an Industry Practices Study", *R&D Management 31*, 2001, pp. 361-380.

[17] M.A. Cusomano, *The Business of Software,* Free Press: New York, 2004

[18] M.A. Cusumano and R.W. Selby, *Microsoft Secrets,* Simon and Schuster, New York, 1995.

[19] A.S. Dver, *Software Product Management Essentials,* Anclote Press, 2003.

[20] C. Ebert, “Requirements Before the Requirements: Understanding the Upstream Impacts”, *Proceedings of the 13th IEEE international requirements engineering conference,* IEEE Comp. Soc., Paris, France, 2005, pp. 117-134.

[21] C. Ebert and J. De Man, “e-R&D – Effectively Managing Process Diversity”, *Annals of Software Engineering (14)* 1, 2002, pp. 73 – 91.

[22] C. Ebert and M. Smouts, “Tricks and Traps of Initiating a Product Line Concept in Existing Products” *Proceedings of the 25th International Conference on Software Engineering (ICSE '03)*, IEEE Comp. Soc., Portland, Oregon, USA, pp. 520-525.

[23] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, Boston, MA, USA, 2003.

[24] R.L. Glass, I. Vessey, and V. Ramesh, “Research in Software Engineering: an Analysis of the Literature”, *Information and Software Technology 44*, 2002, pp. 491–506.

[25] A. van der Hoek, “Software release management”, *Proceedings of the Sixth European Software Engineering Conference together with the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering*. Springer: Heidelberg, Germany, 1997, pp. 159–175.

[26] Information Technology - Open Systems Interconnection - *Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1. 2nd ed. Geneva: ISO, 1994.

[27] T. Kappel, "Perspectives on Roadmaps: How Organisations Talk about the Future", *IEEE Engineering Management Review, 29*(3), 2001, pp. 36-48.

[28] J. Karlsson and K. Ryan, “A Cost-Value Approach for Prioritising Requirements”, *IEEE Software,* 1997, pp. 67-74.

[29] T. Kilpi, “Product Management Challenge to Software Change Process: Preliminary Results from Three SMEs Experiments”, *Software Process - Improvement and Practice, 3*(3), 1997, pp. 165-175.

[30] R. Laqua, “Concepts for a Product Line Knowledge Base & Variability”, *Proceedings of NetObjectDays 2002*, October, 2002, pp. 30-39.

[31] S. Lauesen, "COTS Tenders and Integration Requirements", *12th IEEE International Requirements Engineering Conference (RE'04),* 2004, pp. 166-175.

[32] L. Lehtola, M. Kauppinen, M., and S. Kujala, ”Linking the Business View to Requirements Engineering: Long-Term Product Planning by Roadmapping”, *Proceedings of the 13th IEEE international Conference on Requirements Engineering (Re'05).* IEEE Comp. Soc., 2005, pp. 439-446.

[33] J. de Man and C. Ebert, “A Common Product Life Cycle in Global Software Development”, *Eleventh Annual International Workshop on Software Technology and Engineering Practice*, 2003, pp. 16-21

[34] M. Moon and K. Yeom, “An Approach to Develop Requirement as a Core Asset in Product Line”, *Lecture Notes in Computer Science*, *3107*, 2004, pp. 23 – 34.

[35] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell, “A Linguistic-Engineering Approach to Large-Scale Requirements Management” *IEEE Softw. 22*(1), 2005, pp. 32-39.

[36] B. Nuseibeh and S. Easterbrook, “Requirements Engineering: A Roadmap”, *The Future of Software Engineering,* A. Finkelstein (ed.), ACM Press: New York, 2000, pp. 37-46.

[37] *PMBOK, A Guide to the Project Management Body of Knowledge*, first ed., Project Management Institute, Pennsylvania USA, 2000.

[38] C. Potts, "Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software," *Second IEEE International Symposium on Requirements Engineering (RE'95),* 1995, p. 128.

[39] B. Regnell and S. Brinkkemper, “Market-Driven Requirements Engineering for Software Products”, *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin (eds.), Berlin, Germany, Springer Verlag, 2005, pp 287-308.

[40] G. Ruhe and M.O. Saliu, “The Art and Science of Software Release Planning”, *IEEE Softw. 22*(6), 2005, pp. 47-53.

[41] T.L. Saaty, *The Analytic Hierarchy Process*, McGraw-Hill, New York, NY, 1980.

[42] M.O. Saliu and G. Ruhe, "Supporting Software Release Planning Decisions for Evolving Systems," *29th Annual IEEE/NASA Software Engineering Workshop*, 2005, pp. 14-26.

[43] C. Szyperski, *Component Software: Beyond Object-Oriented Programming, 2nd ed*., ACM Press and Addison-Wesley, 2002

[44] S. Thiel, S. Ferber, T. Fischer, A. Hein, and M. Schlick,, “A Case study in Applying a Product Line Approach for Car Periphery Supervision Systems”, *Proceedings of In-Vehicle Software, SAE 2001 World Congress SP-1587*, Detroit, Michigan, USA, 2001, pp. 43-55.

[45] D. Tsichritzis and A. Klug, “The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems”, *Information Systems 1*, 1978, pp. 173–191.

[46] S. Unger, “Ten Marketing Challenges that Can Make or Break your Business… and How to Address Them”, *Productmarketing.com, 1*(1), 2003.

[47] J. Vähäniitty, C. Lassenius, and K. Rautiainen, "An Approach to Product Roadmapping in Small Software Product Businesses", *Quality Connection - 7th European Conference on Software Quality (ECSQ2002), Conference Notes*, Center for Excellence Finland, Helsinki, Finland, 2002, pp. 12-13

[48] J. Vahaniitty and K. Rautiainen, "Towards an Approach for Managing the Development Portfolio in Small Product-Oriented Software Companies”, Proceedings *of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, 2005, p. 314.

[49] J.M. Versendaal and S. Brinkkemper, “Benefits and Success Factors of Buyer-Owned Electronic Trading Exchanges: Procurement at Komatsu America Corporation”, *Journal of Information Technology Cases and Applications, 5*(4), 2003, pp. 39-52.