Software Product Management

Samuel A. Fricker

Abstract

Software organizations evolve and maintain software solutions with more than a single development project. The delta specifications and artifacts that result from each project make reuse difficult and challenge a company's ability to innovate. Software product management is a growing discipline for understanding how to productize and align software with company strategy, how to evolve software, and how to coordinate product stakeholders. With product focus, in addition to project focus, planning accuracy can be improved, time-to-market reduced, product quality enhanced, and economic success sustained. This chapter provides an overview on software product management and discusses what today is known about this discipline.

1 Introduction

In modern economy the companies that succeed and survive are those able to develop and market winning products. Products like phones, cars, and airplanes are increasingly software-based, rather than being electro-mechanical devices. Services like banking, information provision, and entertainment and infrastructure like telecommunications and power networks gradually tend to be run more by software products than by humans. Software is an increasingly important enabler and driver of innovation because of its adaptability and flexibility. At the same time, software allows cutting cost, reducing time, and increasing reliability of services that earlier were performed by humans, enabling humans to focus on complex and little repetitive tasks.

S.A. Fricker
Blekinge Institute of Technology - School of Computing, Karlskrona, Sweden
e-mail: samuel.fricker@bth.se

A. Maedche et al. (eds.), *Software for People*, Management for Professionals,
DOI 10.1007/978-3-642-31371-4_4,

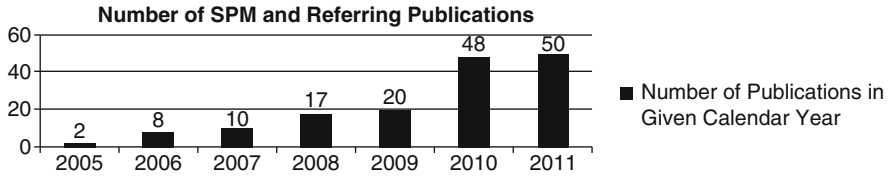

Fig. 1 Growth of the number of scientific publications that have the term "software product management" in the title, abstract, or keywords or that refer to such publications (Source: Scopus (February 26, 2012))

Software organizations see themselves confronted with software solutions evolved and maintained with more than a single development project. They incur a majority of cost not with that first development project, but afterwards (Sneed et al. 2005). The delta specifications and artifacts that result from each project make reuse difficult and challenge a company's ability to innovate. In addition, the ever increasing strategic importance of software requires that solutions are not only technically sound, but are aligned with business strategy.

In 1997, the term *software product management* was coined (Kilpi 1997). Procter & Gamble's idea of letting an individual manager championing and taking responsibility for a product (Gorchels 2006) was applied to a software organization. The software product managers extended configuration management of code and software artifacts with delivery data, customer data, and change requests. An integrated software product management process was established that covered marketing, release planning, software production, sales, customer delivery, and product support in addition to the release projects.

Today software product management is a young, growing discipline that bridges software engineering with business. The first book on software product management appeared in 2002 (Condon 2002). The first workshop of the International Workshop on Software Product Management (IWSPM) series took place in 2006. The first conference on the International Conference on Software Business (ICSOB) series took place in 2010. Figure 1 shows that an increasing number of publications are written on software product management or refer to such publications. These figures are very defensive estimates of the actual amount of activity in the field of software product management.

The aim of this chapter is to provide an up-to-date overview on software product management. It does so by characterizing software products, describing the profile of a software product manager, and introducing software product management practices and interfaces to other company functions. The characterization covers product management both for commercial software and for software solutions used to provide commercial services. The overview gives selected entry points to academic literature and to books about the topic and concludes with a short discussion of the state of knowledge.

The remainder of the chapter is structured as follows. Section 2 introduces the software product concept. Section 3 characterizes the software product manager and company-internal stakeholders. Section 3.2 surveys software product

management references models and describes software product management practices. Section 5 discusses state of knowledge. Section 6 summarizes and concludes.

2 Software as a Product

2.1 Software Products

The term *product* is a central concept in marketing. It designates *anything that can be offered to a market for attention, acquisition, use, or consumption that might satisfy a want or need* (Kotler et al. 2010). The intention behind a product is the satisfaction of a set of people or organizations with comparable needs against some form of compensation. Products can be physical objects, services, people, places, organizations, ideas, or mixes of these entities.

The set of people or organizations the product is made available for is called a *market*. The market-orientation where one product may be instantiated many times differentiates product development with bespoke development where one software instance is developed for one specific customer. Characteristic for something to qualify as a product is its offering to a market, and not the concluded compensation. Even a thing that is made available for potential sales but never sold is called product. The compensation can be, but does not necessarily need to be of commercial or financial nature. Attention, use, or consumption suffices for the thing to be a product.

A "software product" is a *product whose primary component is software* (Kittlaus and Clough 2009). Software is an information good that manifests human know-how in bits and bytes. This characteristic makes a software product special in comparison to other goods.

Software becomes whatever function or application it addresses (Cusumano 2004). The utility of a software product is determined by the functionality it provides at its interfaces. Value is generated as a result of such functionality. For example, an online-banking solution may offer the possibility to enter a payment. The banking solution then generates value by initiating an account movement. The generic character of software makes businesses in many industries depend on software products, provided that the generated value is understood by those targeted by these software products.

An information good like software can be easily copied, shared, resold, or rented (Variant 2000). Production and distribution of copies require little cost in comparison to the development of the software product. This characteristic allows software product sales to achieve fascinating profit margins (Cusumano 2004). The duplication and spreading of commercial software needs to be managed, however. Defined rights need to be licensed (Kittlaus and Clough 2009). Such rights include the right to use, the right to own, and the right to resell the software.

Software can be changed or updated relatively easily by using patches or release updates (van de Weerd et al. 2006b). This flexibility make incremental product

development possible, where a rapid break-even can be reached and a high return of investment achieved (Denne and Cleland-Huang 2003). The high release frequency, however, makes requirements organization highly complex, especially for large and complex software products that offer integration with other software.

The software business has also its dark sides when compared to other kinds of product businesses (Cusumano 2004; Xu and Brinkkemper 2007). Many developers consider themselves to be artists, rather than engineers or scientists. Their productivity differs up to a factor 20. Software product development, hence, is highly unpredictable and risky. Seventy-five to eighty percent of the projects routinely are late and over budget. Planning accuracy of 20 % is considered best practice. However, a successful software product has the potential be considered a "license to print money" with marginal production cost and customers locked in to the vendor. The product business model can be maintained as long as the software company succeeds to sell new products to new customers, rather than selling services and upgrades to their existing customer base.

Three types of software products are often differentiated: packaged software, software as a service, and embedded software (Kittlaus and Clough 2009). These types mainly differentiate themselves in how the product is duplicated and made available to the customer.

Packaged software is commercial software that comes ready-made (Xu and Brinkkemper 2007). Packaged software is often customized when deployed. The larger such software is the more significant customization and deployment effort can be. A typical large-scale example is an Enterprise Resource Planning (ERP) system whose adaptation cost usually outweighs the product cost. Packaged software may be offered in a standardized format as commercial off-the-shelf software, as shrink-wrapped software, or as an app. *Commercial off-the-shelf (COTS) software* is packaged software with limited configurability offered to a whole market. COTS is often integrated into other software (Basili and Boehm 2001; Jaccheri and Torchiano 2002). *Shrink-wrapped software* is COTS software that can be bought in stores or downloaded from the web (Flammia and McCandless 1997). An *app* is a form of shrink-wrapped software that typically is of small size and offered through a market place, an *app store*, dedicated for a given operating system (Miller 2010).

Software products may be offered as a service instead of being offered as packaged software (Hayes 2008). Such software is called *Software as a Service (SaaS)*. SaaS is not installed on a user's local PC, but run at a distant server managed by the supplier. The user typically accesses the software through a web browser. One of the benefits of SaaS is runtime binding, where the software decides ad-hoc to access other software services to deliver its functionality to the user (Turner et al. 2003). The SaaS software development model is likely to lead to higher software quality, greater profits, and higher social welfare than traditional forms of software (Choudhary 2007). On-demand software provision through the Internet, also called *cloud computing*, is a new paradigm shift in the software industry (Buyya et al. 2009) that can change the way software is developed and composed (Gold et al. 2004).

A third important form of software products is software embedded in microcontroller-based systems (Lee 2000; Ebert and Salecker 2009). Examples of embedded systems include home appliances (Edwards and Grinter 2001), mobile phones (Roussos et al. 2005), cars (Broy et al. 2007), and power systems (Wu et al. 2005). The end user usually doesn't recognize *embedded software*, but perceives it as a set of functions that the system provides. Embedded software produced by one supplier can be prepared for use in the products of another company that usually is called Original Equipment Manufacturer, OEM (Kittlaus and Clough 2009).

Software companies often do not sell one single product but offer *portfolios* of products tailored for different market segments (Kittlaus and Clough 2009). Quality concerns, total cost of portfolio ownership, and time-to-market of product development drives these companies towards developing platforms and *product lines* that enable planned reuse of software artifacts (Clements and Northrop 2001; Pohl et al. 2005). Usually, higher-level or senior product management is responsible for a product line, the product manager for one or a few of the products, and analysts or requirements engineers for product features or releases.

2.2 Parts of Software Products

A product is a combination of goods and services (Kotler et al. 2010). The *core product* corresponds to the problem-solving services or benefits that customers buy when they obtain a product. The *actual product* consists of various parts, functional and quality features, styling, brand name, and packaging combined to deliver the core product's benefits. The *augmented product*, finally, delivers additional services and benefits such as delivery, installation, after-sales service, and warranty.

The software domain knows engineering, business, and legal interpretations of what a software product consist of. These interpretations characterize the concerns the respective viewpoints are concerned with.

The engineering perspective focuses on the core product. Here, a software product consists of the *complete set of computer programs, procedures, and associated documentation and data designated for delivery to users* (IEEE 1990). These artifacts support planning, design, construction, and quality assurance of the product artifacts, including code development and configuration control (Bersoff 1984). A product repository is used to manage these artifacts across development projects (Kilpi 1997). It contains baselines of consistent product development plans, requirements, architecture, design, code, configuration data, test plans, test cases, test data, and user documentation (Clements and Northrop 2001; Sneed et al. 2005).

The business interpretation focuses on the actual and augmented products. Here, a software product is a *packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market* (Xu and Brinkkemper 2007). The packaged components refer to the software delivered to users. The software-based services cover infrastructure, software environment, and application service provision (Yourseff et al. 2008). These components and services form the actual product. The augmented product

consists of auxiliary materials and the activities that support software distribution and trading. The auxiliary materials comprise software documentation, web pages, user manuals, training material, and brochures. Supporting activities include distribution, customer projects that deliver customization and integration for customers, user training, and maintenance.

The legal interpretation adds a set of rights to the services and artifacts of an augmented software product. Right management and transfer are extensively researched in the context of open source software (Lerner and Tirole 2002). The GNU General Public License GPL (GNU 2007) is one well-known open source license model among many others (Open Source Initiative 2012). These license models allow free use, modification, or distribution of software provided that restrictions with respect to copyright and open source status protection are observed (Ruffin and Ebert 2004). Affected by these restrictions is the right to integrate the open source software into other products that then are reproduced or sold (Alspaugh et al. 2010). A bypassed licensor is entitled to force an integrator to end the affected product's production, delivery, and sale and to claim for damages.

2.3 Classification of Software Products

Products can be classified based on their role for the customers (Kotler et al. 2010). Each product class has a particular customer buying and use behaviors that are addressed with a specific pricing, distribution approach, and promotion tactic. Similar differences can be observed between each type of software product: packaged software, embedded software, and SaaS. Table 1 provides an overview and examples of products.

Consumer products target end consumers for personal consumption. Convenience goods are bought relatively frequently with a little comparison and buying effort. Shopping goods are less frequently bought with significant comparison effort to alternatives. Specialty goods have unique characteristics and high brand identification. Here the customer is willing to make a special purchase effort.

Industrial products target other companies or in-house business units. These products are used to again build and offer value added products or services (Jansen et al. 2007). Materials and parts enter the customer's products completely. These products are sold directly to the industrial users who mostly look on price and service. These customers are, in the case of embedded materials and parts usually called Original Equipment Manufacturers, OEMs (Kittlaus and Clough 2009). Capital items help in the customer's production or operations and partly enter the customer's products. Capital items tend to be sold through intermediaries or integrated to the customer's environment with dedicated projects. Supplies and services are needed in the customer's production or operations, but do not enter the customer's products. Supplies are usually purchased with minimal comparison. Services are usually purchased under contract.

Predominantly in the context of SaaS, the term software product is not limited to the world of software vendors, but also encompasses the world of corporate IT

Table 1 Product classifications (Adapted from Kotler et al. 2010), and product types with examples

Product Classification \ Software Product Type		Non-Software	Packaged Software	Embedded Software	SaaS
Consumer Products	Convenience Goods	Food	Web browser	Memory stick	Web-based e-mail
	Shopping Goods	Furniture	PDF reader	Smart phone	Music streaming service
	Specialty Goods	Housing	Office suite	Car	Online banking
Industrial Products	Materials and Parts	Cement	Development libraries and components	Specialized chipsets	*Web services*
	Capital Items	Factory	Development environment (tools)	Server farm	*Billing system (e.g. in telecom)*
	Supplies and services	Management consulting	Enterprise resource planning system	Copier machine	*Web-based CRM system*

organizations (Kittlaus and Clough 2009; Cabinet Office 2011b). Corporate IT organizations deliver and operate software products in-house that enable business units to deliver business services to customers (Ward and Peppard 2002). The corporate IT organizations establish software products usually as *standard software* that is routinely installed on most computers within the business unit.

3 The Software Product Roles

Procter & Gamble has introduced product management to improve the performance of one their product lines (Gorchels 2006). Their key idea was to let an individual manager assigned responsibility for these products compete with others. This system was so successful that it was copied over again since then.

3.1 The Software Product Manager

The *software product manager*, often called *solution owner* in the context of information systems, is a *mini Chief Executive Officer* with full business responsibility of a software (Ebert 2006). A software product manager is a middle manager responsible for developing new products and managing and marketing existing ones (Gorchels 2006; Cabinet Office 2011). He defines strategy of these products, aligns them with company strategy and markets needs, and executes the strategy-implementing plans by coordinating product development, marketing, sales, distribution, service, and support.

Job descriptions for software product managers usually cover the following responsibilities. He participates in innovation and takes leadership for new product

ideas. He develops the product strategy that aims at achieving sustainable economic product success in line with corporate strategy (Kittlaus and Clough 2009). In the context of commercial software, he plans the marketing mix and monitors product success in collaboration with marketing. He plans product scope and monitors product evolution in collaboration with development. He represents the product inside and outside the company and coordinates product operations with company units including sales, distribution, service, and support.

A software product manager has at least a Masters title in informatics or business administration and has developed broad knowledge of relevant technologies and markets. He has proven ability to communicate and negotiate successfully with all important product stakeholders inside and outside the company. He has excellent self-management abilities that allow him to work on a broad spectrum of tasks with changing priorities.

To summarize, the software product manager stands for what the product does, what it is, who it serves, and what it means to the company and customers (Gorchels 2006). He is a champion and driver for successful evolution and revolution of the company's portfolio.

3.2 External and Internal Stakeholders of Software Product Management

Each company acts in an industry, where it is embedded in an ecosystem suppliers, customers, system integrators, partners, competitors, suppliers of substitute products or services, and potential new entrants (Porter 1998; Messerschmitt and Szyperski 2003; Popp and Meyer 2010). *Suppliers* deliver components, platforms, and systems that enable development and running of software products (Jansen et al. 2007). *System integrators* design and implement software-based systems that are used by customers and into which software products are integrated. *Partners* may resell software products, provide referrals, integrate software products into their products, cooperate in product development, and provide certification services (Popp and Meyer 2010). *Competitors, suppliers of substitute products or services, and potential new entrants* are perceived to be threats for a product that are addressed by protecting intellectual property and by innovating continuously (Hyland and Beckett 1994; Pech 2006).

A product cuts across the company's value chain (Porter 1998). To align the product with the company's processes and activities, the product manager coordinates company-internal stakeholders (Lehmann and Winer 2005; Gorchels 2006; van de Weerd et al. 2006; Kittlaus and Clough 2009; Bekkers et al. 2010). Product management ensures that the product and the stakeholders' activities are aligned with company strategy, the market situation, and the stakeholders' interests.

The organizational structure, the position of product management, and the naming and scope of the various company functions differ from company to company and over time (Lehmann and Winer 2005). This limits the validity of any global definition of product management stakeholders. The following stakeholder

characterizations, hence, need to be interpreted and adapted to each organization under consideration.

The interface between a company's marketing and development functions is the most critical in terms of challenges and criticality for product success (Griffin and Hauser 1996). The product manager bridges two these functions while planning and aligning the product with corporate strategy. The product management responsibilities and activities ensure that product-related knowledge and information are adequately shared and that product-related activities are coordinated.

The product manager closely cooperates with *marketing management* that is responsible for market and customer success. Marketing management *builds and maintains beneficial exchanges with target buyers* and *manages demand and customer relationships* (Kotler et al. 2010). The product manager is the company-internal supplier who delivers new and improved products to marketing. Marketing represents the voice of the customer, for example by providing access to marketing research and customers. Marketing also ensures that the products address the needs of target markets. Upon successful implementation of a product release, marketing undertakes promotion and selling efforts for bringing the released products to the customers and satisfy their needs.

For defining the product strategy and planning product evolution, the product manager collaborates with the following additional functions and roles. He collaborates with *research and development* to identify opportunities for product and technology innovation (Garcia and Calantone 2002). He performs triage and aligns the product with corporate strategy whose owner is *higher-level management* (Davis 2005). Higher-level management balances the overall portfolio, prepares and executes measurement and analysis, defines a consistent marketing and sales strategy, and provides funding and resources for product development and operations (Ebert 2006).

The product manager closely cooperates with product development *project managers*. Such a project manager is responsible for successfully executing a project that leads to a product release. A project, in contrast to a product, is characterized by *having a definite beginning and a definite end* and by *creating unique deliverables in a sequence of steps* (Project Management Institute 2004). The product manager is the company-internal customer who contracts each product development project by setting priorities for the targeted product release. A project manager is responsible for one such project. He ensures feasibility of the project scope, implements the agreed scope, and hands the implemented product release over to the product manager.

For launching and delivering an implemented product release, the product manager collaborates with the following additional functions and roles. These roles mostly address operational concerns on a tactical level.

He collaborates with *distribution* who make the software product ready for consumption. In the case of software-as-a-service, distribution may be called *production* and is responsible for operating and hosting the software (Cabinet Office 2011). In the case of packaged software, distribution is responsible for

creating the product package and shipping it to the points-of-sales or the customers. Distribution interfaces with inbound and outbound logistics.

The product manager collaborates with *sales* and with *customer projects* that enable the customers to take advantage of the values delivered by the product. Such customer projects can involve system engineering projects, such as building a substation for power distribution, or organizational and process changes, such as in the deployment of an enterprise resource planning system. He collaborates with *customer service and support* who deliver user training, consultancy, maintenance service, and upgrades (Goffin and New 2001).

The product manager collaborates with *finance* who ensure a sufficient revenue stream for the company (Konig 2009). He collaborates with *legal* who provide advice regarding license use and who protect and defend the product against competitors and potential new market entrants.

4 Software Product Management

A software organization faces a highly competitive environment. A good product idea and a well-executed new product development project are not enough to be successful. Usefulness of a software product in accordance with the organization's strategy can only be achieved and sustained with an interrelated set of competences, practices, and processes for planning, building, marketing, distributing, evolving, and maintaining a software product.

Software product management is the *discipline, which governs a software product from its inception to its close-down to generate as large value as possible for the business*. This definition, adapted from Alcatel (Ebert 2006), encompasses three important facets: governance of a software, coverage of the full software lifecycle, and business value generation.

Software product management encompasses the practices needed for governing a software product. Governance refers to structures, processes, and relational mechanisms implemented to ensure that business goals are achieved, resources responsibly utilized, and risks adequately managed (Meyer et al. 2003; De Haes and Van Grembergen 2004). A software product manager, the mini-CEO, is responsible and accountable for the success of the software product (Ebert 2006). The product manager plans the scope and evolution of the software product and aligns it with user, market, and company needs (Gorchels 2006). To obtain commitment from the organization and to reduce risks, the product manager actively involves important stakeholders from development, marketing, sales, distribution, service, and support (Aurum and Wohlin 2003; van de Weerd et al. 2006).

A software that is used undergoes continual change or becomes progressively less useful (Lehman 1980). This law applies not only for software use, but also for the product's position in the market and its utility for the company. The product needs to be adjusted to changing customer preferences and moves by competitors (Porter 1998). A software product is adapted differently based on its current lifecycle stage (Raijlich and Bennet 2000). Initially the product is evolved to extend

its capabilities and functionalities. Later when changes become difficult, it is serviced with defect repairs and simple functional changes to maintain its usefulness. The ensuing phase-out stage stops servicing, but continues revenue generation until the product is closed down. Each phase requires specific expertise, employs particular architectural tactics, and pursues own economic goals.

Every investment requires return of investment to be generated. Return can be generated with a broad variety of values, including values perceived by customer and users, market value for the company, shareholder value, production value, product differentiation, intellectual capital, value of technology, and innovation value (Khurum et al. 2011). The software product manager is responsible for prioritizing value generation and aligning the software product with company strategy. Strategic management instruments (Kaplan and Norton 1992), business cases (Reifer 2002), and goals and requirements (Gorschek and Wohlin 2006) are means to plan and manage the implementation of such value generation.

4.1 Software Product Management Reference Models

Software product management stands for a broad variety of practices for aligning the product business with corporate strategy, steering software development, obtaining customer interest, generating revenue, and supporting product use. The scope of product management can be derived from textbooks (Lehman 1980; Condon 2002; Gorchels 2006; Dver 2007; Lawley 2007; Cagan 2008; Haines 2009; Kittlaus and Clough 2009). A systematic overview, however, can be gained by studying reference models that describe software product management practices. They have been developed based on empirical research (van de Weerd et al. 2006; Bekkers et al. 2010), personal experience in large-scale companies (Ebert 2006, 2009; Kittlaus and Clough 2009), as a result of industry consultancy and training (Pragmatic Marketing 2012; Steinhardt 2012), and by seeking consensus between academic and industrial opinion leaders (ISPMA 2012).

The software product management reference models are used to benchmark, improve processes, and train practitioners. They refer to 28–68 capabilities, processes, activities, or competencies and are structured according to product lifecycle phases, functional areas, interfaces to company function, abstraction levels, or impact of decision making. Figure 2 shows an overlay of the reference models' structuring mechanisms.

A product artifact hierarchy consisting of portfolio, products, releases, and requirements, is used to structure *van de Weerd's reference framework for software product management* (van de Weerd et al. 2006; Bekkers et al. 2010). Each level in the artifact hierarchy refers to groups of three or six focus areas: portfolio management, product planning, release planning, and requirements management. The total of 15 focus areas again structure 68 capabilities of a software product organization that are implemented in collaboration with company-external and internal stakeholders.

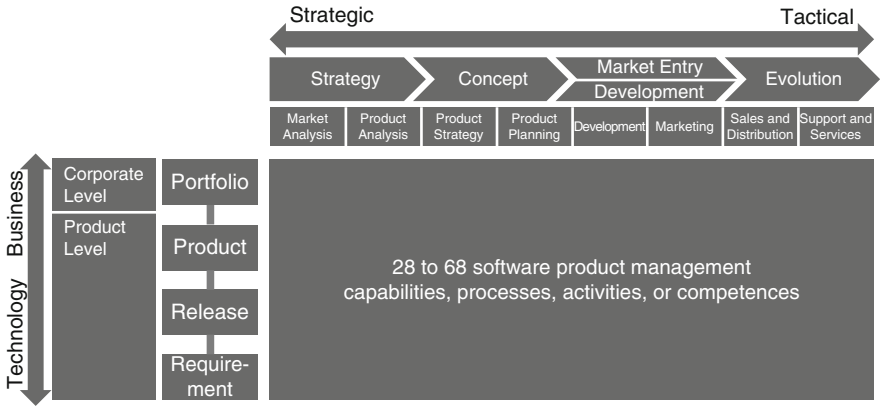

Fig. 2 Overlay of software product management reference models (Ebert 2006; Kittlaus and Clough 2009; Bekkers et al. 2010; Pragmatic Marketing 2012)

Lifecycle phases are the structuring mechanism used in *Ebert's software product management framework* (Ebert 2006, 2009). The phases strategy, concept, market entry and development, and evolution follow the major milestones of a new product development effort (Cooper 2001) and are used to order 18 product management processes and 10 competencies.

Eight functional areas, visualized as columns, are used to structure 49 activities in *Kittlaus' s software product management framework* (Kittlaus and Clough 2009). The areas are market and product analysis, product strategy and planning, and collaboration with development, marketing, sales and distribution, and support and services. The framework also differentiates two abstraction levels that distinguish activities performed at the corporate and the product levels.

A well-known reference model for software product management that emerged from industry consultancy and training is the *Pragmatic Marketing framework* (Pragmatic Marketing 2012). It delineates management and marketing practices for technology products on a continuum of concerns from strategic to tactical: market, strategy, business, planning, programs, readiness, and support. Pragmatic Marketing tailors the framework to characterize special kinds of product managers. The director of product strategy is responsible for strategic business-oriented practices like market definition, product portfolio management, and business planning. The technical product manager is responsible for strategic technology-oriented practices like technology assessment, product roadmapping, and requirements management. The product marketing manager, finally, is responsible for tactical practices like customer acquisition, product sales, and support.

The *software product management body of knowledge (SPMBoK)* of the International Software Product Management Association (ISPMA) is a consensus between academic and industrial opinion leaders (ISPMA 2012). It integrates Van de Weerd's, Ebert's, and Kittlaus's frameworks for product management education purposes. In contrast to other reference models, the scope and contents of the SPMBoK continues to evolve by being discussed and adjusted by any product

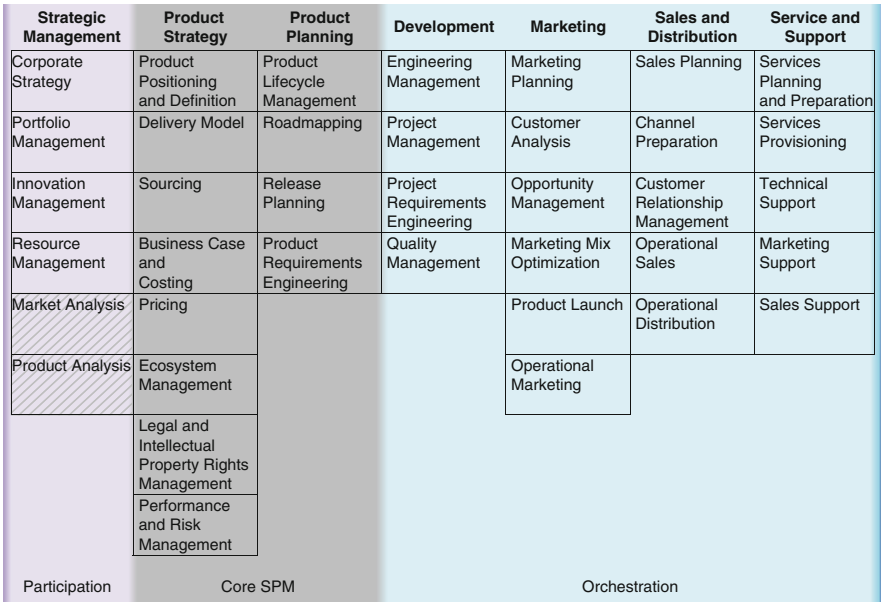

Fig. 3 The software product management body of knowledge (SPMBoK) (ISPMA 2012)

management expert that joins the ISPMA knowledge network. It hence adapts to the evolving understanding of the discipline.

The SPMBoK structures 38 practices again along functional areas. The product manager participates in strategic management, is directly responsible for product strategy and planning, and orchestrates development, market, sales, distribution, service, and support. In small companies, market and product analysis are performed as core activities by the product manager. Figure 3 provides an overview of the SPMBoK.

The various structuring approaches and naming of product management practices make it difficult to provide a single, globally accepted view of software product management. The SPMBoK provides the broadest consensus of software product management opinion leaders and is used here to structure the overview of the various practices.

4.2 Strategic Management

Strategic management is concerned of the question how a company achieves and sustains competitive advantage (Teece et al. 1997). This SPMBoK pillar represents the interface between software product management and the company's higher-level management. The latter sets goals and constraints for the portfolio of company offerings, and the product manager influences and adjusts the company's strategy. The product manager participates but does not take leadership or responsibility.

Corporate strategy consists of the vision and approach a company takes to compete with other firms. The following three questions help creating a strategic vision for a company (McGrath 2001): *Where do we want to go? How will we get there? Why do we think we will be successful?* Executive management answers these questions for the company with a defensible position against competitive forces (Porter 1980), competitive moves (Shapiro 1989), capability development (Wernerfelt 1984), and the adaptation and use of these capabilities (Teece et al. 1997).

Portfolio management is the balancing of risk versus return, maintenance versus growth, and short term versus long term by making choices of markets, products, and technologies to invest in and allocating resources to (Cooper et al. 1999). Portfolio decisions allow a company to focus its resources on a few important activities, hence to ensure its ability to act. A well-known portfolio management approach is the Boston Consulting Group analysis used to decide which products to invest in, which products to use as funding sources, and which products to eliminate (Henderson 1979). A successful product takes different positions in the company's portfolio during its lifetime (Haines 2009). It requires initial funding during the product definition phase, where concepts and feasibility are evaluated. It requires substantial funding during the product introduction phase, where the focus lays on development, launch, and growth. It then acts as a revenue source for other products during maturity and decline before it is eliminated. Each phase requires dedicated product management activities.

Innovation management in a product context refers to the continuous renewal of technologies and product offerings (Trott 2011). Innovation is crucial for a company to achieve competitive advantage and long-term success. The innovation process includes the discovery of new possibilities, the choice and combination of potential innovations, and the delivery of value (Hyland and Beckett 1994; Gorschek and Fricker 2010). Appropriate organizational structure, funding and resources, and an innovation-encouraging climate provide the necessary context for innovation. The degree of desired innovativeness (Garcia and Calantone 2002) determines how product management is involved in innovation. A product manager may be a champion and leader for a radical or really new innovation. A product manager may oversee the whole innovation process in the case of incremental innovations.

Resource management is closely related to corporate strategy and portfolio management. The resources managed by a company include factors of production such as unskilled labor and capital, include firm-specific assets that are difficult to imitate such as trade secrets, specialized production facilities, business knowledge, and engineering experience, and include organizational routines and competences (Teece et al. 1997; Gold et al. 2001). Resources are scarce, hence are increasingly sought outside the boundaries of the company (Chesbrough 2003). If resources are adapted to business needs and utilized in a focused manner, they can be a source for competitive advantage.

Market analysis delivers key inputs to positioning the company and its products in the industry. Market analysis is a basis to determine a company's opportunities and threats. A well-known market approach is PEST analysis, the analysis of

political, economic, social or societal, and technological factors (Ward and Peppard 2002) usually enhanced into the PRESTO analysis with regulatory and other factors (Haines 2009). Porter's five forces analysis represents a complementary analysis approach focused on customers, suppliers, existing and future competitors, and product substitutes (Porter 1998). Inputs for market analysis can be found by studying industry-specific journals and periodicals, attending fairs and conferences, scanning the Internet, or procuring market research reports from commercial analysts such as Gartner (Fenn and Raskino 2011), IDC, and Forrester Research.

Product analysis delivers information about current product performance. Product analysis is one of the inputs to determine a company's strengths and weaknesses. Many data for product analysis can be provided by finance controllers (Kittlaus and Clough 2009). Typical quantitative data are product revenue, development and operations cost, profit, sales, and support requests. Qualitative data are feedback from customers (Anderson and Sullivan 1993; Johnson and Gustafsson 2000), sales channels, and market analysts, and opinions from trade press in articles. Market analysis and product analysis provide inputs for the analysis of strengths, weaknesses, opportunities, and threats (SWOT) that can be used as a basis for defining competitive moves (Piercy and Giles 1989).

4.3 Product Strategy and Planning

Product strategy and *product planning* describe what the software product will be and how and when it will be developed and used. These two SPMBoK pillars represent the core areas of software product management. The product manager here defines the software product, sets goals and constraints for its evolution, enables business, and obtains support and commitment from stakeholders. The product manager takes leadership and responsibility for the decisions taken. The kind of software product, commercial software or in-house information system, determines how much product strategy and planning cover marketing aspects.

4.3.1 Product Strategy

Product positioning and product definition refers to the product vision and characterization of targeted markets, product use, and product scope (Kittlaus and Clough 2009). A product vision answers the three vision questions for the software product (McGrath 2001): *Where do we want to go? How will we get there? Why do we think we will be successful?* The first question can be answered with the target markets (Haines 2009), the intended differentiation with competitive products, and the planned support of company strategy (Gorschek and Wohlin 2006). The second question can be answered with the intended product use, characterized with user personas (Pruitt and Grudin 2003), use scenarios (Cockburn 2001; Cohn 2004), and value (Khurum et al. 2011), and the intended product scope, characterized with a catalogue of product features (Classen et al. 2008). The third question can be answered with the process for developing, evolving, marketing, delivering, and supporting the software product (Cooper 2001) and with a risk management plan

(Miller 1992; Carr et al. 1993). A vision is a short and concise statement with the product's key ideas and is elaborated with supporting documentation. An understood and accepted vision is important for success and short time-to-market (Lynn and Akgün 2001; Tessarolo 2007), but is surprisingly difficult to establish. It hence should be developed in collaboration with stakeholders and peers and be evolved based on results of other product strategy and planning activities (McGrath 2001).

Ecosystem management refers to building or integrating into the product's industry, the business network (Iansiti and Levien 2004). A company becomes part of a value chain in that business network by offering products and services to customers that build on products and services from suppliers (Jansen et al. 2007). The value chain structure and the company's position in that value chain affects the organization's business model (Popp and Meyer 2010). Typical players in the software industry, differentiated by their business model, are vendors, distributors, resellers, OEMs, integrators, and technological alliances (Kittlaus and Clough 2009). Software architecture, services, and competitive moves help disrupting an ecosystem and sustaining the thus obtained position (Iansiti and Levien 2004).

The *delivery model* refers to the approach chosen for delivering the software and the software's services to the users (Kittlaus and Clough 2009). The choice of the delivery model affects the licensing and pricing model and the distribution of activities between the company and the customers. Packaged software implies a sales contract, delivery of the software to the customer, installation of the solution in the premises of the customer, and potentially a customer project for tailoring and integrating the solution. Software-as-a-Service implies hosting the solution, a service level agreement with the customer, application service provision, and on-demand delivery of the software-enabled services to the user through the internet or intranet (Gold et al. 2004).

Pricing refers to the process of setting prices to market offerings. Pricing is a key instrument to generate profitable growth opportunities for the company (Nagle and Hogan 2006). A comprehensive pricing strategy captures the value offered by the product, adapts the price structure to the customer segments, ensures that price and value can be communicated, manages customer and employee expectations with an accepted price policy, and sets price levels to maximize profitability. Pricing models for software are adapted to product lifecycle stage, distribution channels, geographic location, customer importance, and delivery model (Kittlaus and Clough 2009). The pricing approach of a corporate IT product is similar to commercial software, except that the customers are the company's business units and that the price should equal cost.

Sourcing refers to the decisions of making or buying parts of the software solution (Kittlaus and Clough 2009). Buying software can lead to faster time to market, reduced development cost, reduced knowledge and resources needs for development, and increased software quality. At the same time, it can lead to reduced ability to adapt the software, generate integration problems, and build a dependency on a supplier (Boehm and Abts 1999). Sourcing software-as-a-service

enables run-time binding of sourced software, but is perceived to be afflicted with security risks (Benlian and Hess 2010).

A *business case* is a scenario for economic evaluation of an investment. It is used by management to project likely financial results and other business consequences (Brugger 2009). A business case refines the product vision with economic information based on cost, price, competition, and market estimates (Gorchels 2006). *Costing* refers to the development of cost estimates. A business case describes a timeline of expected cash flows, explains the estimation methods and assumptions that were used, characterizes quantifiable business impact and non-quantifiable benefits, and discusses critical success factors and risks (Schmidt 2002). Typical software business cases discuss investments based on contributions to business objectives such as sales and financial performance (Schmidt 2002), efficiency and productivity increases and quality improvements (Reifer 2002), and avoidance of non-compliance problems.

Legal and intellectual property rights management refers to the practices and artifacts for performing business transactions and protecting the business. License or service level agreements are established and used to resolve conflicts between the vendor and the customer about the software product's functionality and quality, permitted amount and kind of product use, and handling of unused but paid product instances (Kittlaus and Clough 2009). Trademarks, trade secrets, copyrights, and patents can at many places be used to protect a software product against piracy. This implies that a software product needs to be free from legal problems before it is released, requiring lawful engineering practices (German et al. 2010) such as auditing its licensing structure (Alspaugh et al. 2010).

Performance and risk management refers to measuring the success of the software product and reacting to problems and risks. The typically monitored performance factors differ between the product lifecycle stages (Anderson and Zeithaml 1984). During the product's inception phase, innovation metrics help to benchmark, diagnose, allocated resources, compensate employees, inform markets, and setting future goals (Kuczmarski 2000). The focus changes towards monitoring buyers, advertising, and purchase frequency during the introduction stage; towards segmentation, production and marketing efficiency, and customer need satisfaction during the growth stage; towards process efficiency, marketing and distribution cost reduction, and product differentiation during the maturity stage; and towards competitive strength during the decline stage (Anderson and Zeithaml 1984). Performance measurements help identifying problems and risks that are addressed with avoidance and mitigation actions (Miller 1992; Carr et al. 1993).

4.3.2 Product Planning

Product lifecycle management in the context of a software product relates to planning the product lifecycle. A software product's initial release requires different expertise, architectural decisions, and economics than its subsequent evolution, servicing, phase-out, and close-down (Raijlich and Bennet 2000). The initial focus on learning the application domain and technology changes towards retaining expertise and understanding the interfaces and operation of the software. The

architecture that initially balanced flexibility with time-to-market is continuously adapted and decays until it becomes too hard to change (Lehman 1980). The initially heavy investments start to generate revenue that is extended as long as possible (Henderson 1979). Similarly the focus of product management collaboration with company functions changes from research and development towards marketing, sales and distribution, and service and support.

Roadmapping refers to planning the evolution of a product. A roadmap shows how product features, technologies, and resource needs evolve (Albright and Kappel 2003; Phaal et al. 2003; Lehtola et al. 2005). Roadmaps are used to translate product strategy into long-term plans for research, development, marketing, sales, distribution, service, and support by capturing on best knowledge of the corresponding company functions and obtaining their commitment and support (Phaal et al. 2007). Roadmapping is adaptable to the needs of small to large-scale companies (Vähäniitty et al. 2002; Phaal and Muller 2009).

Release planning refers to the selection and assignment of requirements to projects for implementing sequences of product releases (Svahnberg et al. 2009). Implementing a product in incremental releases, rather than implementing the full product scope at once, allows reaching earlier break-even and a higher return of investment (Denne and Cleland-Huang 2003). Incremental development requires prioritization of functionality and quality levels (Berander and Andrews 2005; Lehtola and Kauppinen 2006). Release decisions aim at achieving usefulness and competitiveness (Regnell et al. 2008), satisfying capacity, schedule, business, and stakeholder needs (Cohn 2004; Wohlin and Aurum 2005), and accounting for requirements interdependencies (Carlshamre et al. 2001).

Product requirements engineering, also called *market-driven requirements engineering*, refers to the process of collecting stakeholder needs, expectations, and ideas for guiding the implementation of the software product (Regnell and Brinkkemper 2005). The product manager performs triage to reduce the large amount of inputs and to ensure the inputs' relevance and feasibility (Davis 2005; Gorschek and Wohlin 2006). These inputs are translated into product features that represent options for product evolution (Fricker and Schumacher 2012) with understood implications on architecture and implementation (Fricker and Gorschek 2010).

4.4 Orchestration of Company Functions

The software product manager depends on various company functions to realize the product vision and to accomplish the plans defined and agreed during roadmapping. These remaining four SPMBoK pillars represent the interfaces between software product management and these company functions. Development implements the software, marketing identifies and wins customers, sales and distribution generate revenue, and service and support facilitate product use. The product manager acts as company-internal customer and orchestrates the company functions, but delegates responsibility.

4.4.1 Development

Software product management orchestrates development by jointly innovating, communicating functional and quality requirements, procuring and developing technologies, planning and steering software implementation, and accepting the achieved results.

Engineering management refers to managing knowledge and staff and structuring software development to achieve development efficiency at an acceptable level of quality. A software organization needs to develop expertise and enable knowledge sharing and collaboration even when employees come and go (Rus and Lindvall 2002; Bosch 2009). Tapping into knowledge and developer networks and establishing culture and reward systems are common means. Platforms and software product lines enable planned reuse of software artifacts, hence increase development efficiency and software quality (Clements and Northrop 2001; Pohl et al. 2005). Software release management allows maintaining an overview on the many different versions of software artifacts (van der Hoek and Wolf 2003; Jansen and Brinkkemper 2006).

Project management refers to the process and activities for developing a software release (Carmel and Becker 1995). Stage-gates (Cooper 2001), a software development lifecycle model (Wallin et al. 2002), and project management practices (Project Management Institute 2004) are used to structure and control a development project. Agile approaches are increasingly employed to manage the uncertainties inherent in a new product development environment (Pichler 2010). Dedicated practices help acquiring and integrating software components (Brownsword et al. 2000) or outsourcing development (Krishna et al. 2004).

Project requirements engineering refers to the project team's inquiry process of eliciting requirements, specifying the software system the team intends to implement, and validating that specification with stakeholders (Potts et al. 1994). A rich body of techniques can be used to reach a shared understanding of requirements and to manage requirements changes (Pohl and Rupp 2011). Product management plays a key role in the communication of requirements (Fricker et al. 2010), controlling progress, and accepting the developed solution (Martin and Meinik 2008).

Quality management refers to the practices a development organization implements to meet its critical success factors and to mitigate business-critical problems (Kitchenham and Pfleeger 1996). Software quality is achieved and maintained by measuring quality (Ebert and Dumke 2007; Jones 2008) and establishing practices and responsibility for quality management (El Emam 2005). Many organizations implement process improvement programs to improve their engineering and management practices in software product management (Bekkers et al. 2010), software development (CMMI Product Team 2010), and IT-based service provision (Spalding and Case 2007).

4.4.2 Marketing

Software product management orchestrates marketing by jointly analyzing the market, customers, and opportunities, launching products and analyzing their performance, and winning customers.

Marketing planning refers to refining the *product positioning and definition* by defining the marketing goals, the marketing mix, and the approach, budget, and controls for reaching the marketing goals (Kotler et al. 2010). Marketing planning characterizes the current marketing situation by building on market, product, and customer analysis. From the identified strengths, weaknesses, opportunities, and threats are derived target sales, market share, and profits and the broad approach to achieve these objectives. Action programs, budget, and controls describe how that marketing strategy is implemented.

Customer analysis refers to the first part of marketing planning: determining and prioritizing market segments, understanding customer needs, and matching these needs with unique selling points. A company chooses between different levels of market segmentation, ranging from one offering for all customers to segment-specific offerings to tailored offerings for each individual customer (Kotler et al. 2010). Similarity of customer needs and characteristics such as geography, demography, psychography, and behavior are a basis for such market segmentation. Analysis of segments attractiveness and business strengths are a basis for prioritizing segments and for identifying actions needed to serve them (Gorchels 2006; Haines 2009). Segments are often represented with personas (Pruitt and Grudin 2003) whose key needs are matched with a suitable marketing mix differentiated from competitive offerings (Hauser and Clausing 1988).

Opportunity management refers to refining the product positioning by narrowing the market (Haines 2009). Criteria such as resource availability and capability, strategic significance, financial viability, and potential customer satisfaction are used to choose among opportunities. The resulting priorities help optimizing the marketing mix, inform the product launch and operational marketing, and guide sales.

Marketing mix optimization refers to the definition and improvement of product, price, place, and promotion to influence demand (Kotler et al. 2010). The marketing mix is what customers actually see in the marketplace. The product defines what the customer will get: functionality, quality, design, brand name, packaging, services, and warranties that address the customer's needs and wants (Boatwright and Cagan 2010). The price determines the cost to the customer. The place refers to the channels and distribution practices that make the product more or less conveniently available. Promotion refers to activities that communicate the product's merits to persuade customers to buy it. The interplay between the elements of the marketing mix affects the success of a marketing campaign (Haines 2009). Marketing mixes of competitive products can be used to analyze competitors as part of market analysis (Lehmann and Winer 2005).

Product launch refers to the process of preparing the public release of the product (Gorchels 2006; Lawley 2007; Haines 2009). The product launch makes the product visible to the markets, hence initiates the company's interaction with channels and customers and generates competitive responses. Launches are prepared by stabilizing and assuring the quality of the software (Galen 2004), by determining product viability (Kaulio 1998), by evaluating the marketing strategy, by timing and planning the market entry sequence, and by introducing the sales

force, channels, and customer service (Gorchels 2006). The launch is accompanied with measurements to evaluate the new product and to improve the new product development process.

Operational marketing refers to marketing communication and analyzing the effectiveness of that communication (Gorchels 2006). A marketing communications system is established with intermediaries and the public to influence the customers with messages and word of mouth (Kotler et al. 2010). Public relations, press kits, articles, demonstrations at trade shows, press releases, and advertising are used to communication the product's unique selling proposition or most critical benefits for the most critical markets (Gorchels 2006). Measurements are established to determine customer awareness, knowledge, liking, preference, conviction, and purchase (Kotler et al. 2010).

4.4.3 Sales and Distribution

Software product management orchestrates sales and distribution by planning the sales, preparing sales channels, and supporting sales operations and product distribution.

Sales planning refers to defining the market and product profiles, preparing sales channels, and training sales in how to sell the software supporting (Haines 2009). A software company's sales channels include direct sales, telesales, internet sales, sales through partners such as integrators, and sales through resellers (Kittlaus and Clough 2009). Sales materials describe customers, their rational and emotional reasons for buying the product, decision and purchase processes, and the role of influencers (Gorchels 2006). A similar approach is taken for *channel preparation*, with the exception that channels usually do not receive company-internal information (Gorchels 2006). Sales and distribution channels need to be coordinated and position the product favorably compared with alternatives (Kittlaus and Clough 2009).

Customer relationship management refers to managing interactions with customers and sales prospects along the customer lifecycle (Buttle 2008). *Operational sales* refers to setting adequate incentives for salespeople, forecasting and planning customer interactions, and monitoring sales progress and success (Calvin 2004). Consolidated results of the customer interactions are fed back to marketing and software product management to support market and product analysis.

Operational product distribution refers in the case of packaged software to production and shipment, online download, or automated deployment (Humble and Farley 2010) and in the case of software-as-a-service to operating servers and the software that offers services to customers (Rhoton 2010; Cannon and Wheeldon 2011). Software distribution includes updating of existing software installations with updates and patches (Ballintijn 2005).

4.4.4 Service and Support

Software product management has to support users, marketing, sales, and customer projects to facilitate product use and the product's business operations.

Services planning and preparation and *services provisioning* refer to facilitating product deployment and use. Services include bespoke projects are performed to customize, enhance, install, and integrate software (Kittlaus and Clough 2009). Large-scale packaged software or products for complex systems are often performed by consultants or integrators. Other forms of services targeted at users include the provision of helpdesks to provide *technical support* (Bruton 2002).

Marketing support and *sales support* refer to providing sales and customer service trainings, performing events and promotions, and producing brochures and other materials of publicity and sales (Gorchels 2006).

5 State of Knowledge

Software product management is a young discipline that aims at closing the gap between business and software engineering. Much of the discipline has its roots in the corresponding established bodies of knowledge. Nonetheless, a growing part of the discipline, especially in the area of product planning, contributes with research results specific to the management of software products.

Software product management has not succeeded yet to establish a sharp differentiation, neither to traditional product management nor to software engineering. The implications of the special characteristics of software, for example its flexibility, intangibility, and ease of duplication and distribution, on product management are not fully understood yet. Conversely, many software engineering practitioners and scholars think that software product management is too far away from the technical artifact to really affect the domain of software.

Software, however, has the potential to change product management. For example, new software technologies such as cloud and app stores establish new communication paths between the company and the so far anonymous mass of customers and users. The utilization of these channels would allow moving away from imagined stakeholders (Karlsson et al. 2007) towards integrating real people in the market-driven requirements engineering process. Such a change can reduce product development risks by basing product decision-making on facts rather than opinion. Early applications of this concept are emerging: social media complement traditional media for eliciting and communicating information (Mangold and Faulds 2009). Analytics are used to study in user behavior in real-time for building user segments, evaluating feature attractiveness, monitoring product performance, and understanding the impact of marketing campaigns and channels (Phippen et al. 2004). App stores simplify software distribution and allow identifying product and service offering gaps (Kim and Park 2010). The field, however, is still far away from providing an understanding of software-enabled and software-specific product management.

Product management also has the potential to change software engineering. The product manager is the central hub for information exchange and focal point for

decision making for a long-lasting software solution. Aligning software product management practice with software architecture can increase overall engineering efficiency and the value of product ownership (Helferich et al. 2006). Companies have started to align management practices with software architecture by moving from software development projects to software product management (Artz et al. 2010). Unclear, however, is still how effective business-socio-technical congruence is achieved and what its empirically grounded business case is.

A third knowledge issue is software product manager education. Most university curricula focus on project-level software engineering and do not teach management and engineering disciplines such as software product management, evolution, and maintenance that cut across development projects. One of the hindering factors is the lacking understanding of educational needs of software product managers. His scope of responsibilities is initially not as broad as the discussed reference models suggest and changes throughout his career when moving from junior to senior positions. Empirically grounded models of the learning process and of how the learning process can be supported effectively are missing today.

6 Summary and Conclusion

Software solutions undergo continuing adaptation or become progressively less useful. Software product management is the discipline that puts attention on the concepts and approaches for achieving long-term usefulness of software for users, customers, and the software organization. A software product manager, armed with the right competences and responsibilities, is able to position and plan software as a product and lead development, marketing, sales, distribution, service, and support towards developing and sustaining a business aligned with company objectives.

This chapter has provided an overview of the current knowledge landscape of software product management. It has introduced software as a product in the contexts of commercial markets and in-house information systems. It has given an overview on reference models of software product management practices. It has discussed the various elements of software product management based on the ISPMA SPMBoK that represents a consensus between software product management industry and academia. It has highlighted limitations of current software product management knowledge and posed some of the most pressing research questions whose answers are needed for developing the field.

Software product management is still in its infancy. Business and software knowledge need to be further consolidated to provide an integrated and clearly differentiated understanding of the peculiarities of product management for software. This requires even more collaboration between industry and academia than has been seen until now. The results will allow tapping into the value of the technology that has become a driver for innovation and growth.

References

Albright, R. E., & Kappel, T. A. (2003). Roadmapping in the corporation. *IEEE Engineering Management Review, 31*(3), 31–40.

Alspaugh, T. A., Scacchi, W., et al. (2010). Software licenses in context: The challenge of heterogeneously-licensed systems. *Journal of the Association for Information Systems, 11*(11), 730–755.

Anderson, E. W., & Sullivan, M. W. (1993). Antecedents and consequences of customer satisfaction. *Marketing Science, 12*(2), 125–143.

Anderson, C. R., & Zeithaml, C. P. (1984). Stage of the product lifecycle, business strategy, and business performance. *Academy of Management Journal, 27*(1), 5–24.

Artz, P., van de Weed, I., et al. (2010). Productization: Transforming from developing customer-specific software to product software. *1st international conference on software business*, Jyväskylä.

Aurum, A., & Wohlin, C. (2003). The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology, 45*, 945–954.

Ballintijn, G. (2005). A case study of the release management of a health-care information system. *IEEE international conference on software maintenance (ICSM 2005, Industry Track)*, Budapest, Hungary.

Basili, V. R., & Boehm, B. (2001). COTS-based systems top 10 list. *Computer, 34*(5), 91–95.

Bekkers, W., van de Weed, I., et al. (2010). A framework for process improvement in software product management. *European systems & software process improvement and innovation (EuroSPI 2010)*, Grenoble, France.

Benlian, A., & Hess, T. (2010). The risks of sourcing software as a service – An empirical analysis of adopters and non-adopters. *18th European conference on information systems (ECIS 2010)*, Pretoria, South Africa.

Berander, P., & Andrews, A. (2005). Requirements prioritization. In A. Aurum & C. Wohlin (Eds.), *Engineering and managing software requirements*. Berlin: Springer.

Bersoff, E. (1984). Elements of software configuration management. *IEEE Transactions on Software Engineering, 10*(1), 79–87.

Boatwright, P., & Cagan, J. (2010). *Build to love: Creating products that captivate customers*. San Francisco, CA: Berrett Koehler.

Boehm, B., & Abts, C. (1999). COTS integration: Plug and pray? *Computer, 32*(1), 135–138.

Bosch, J. (2009). From software product lines to software ecosystems. *13th international software product line conference (SPLC 2009)*, San Francisco.

Brownsword, L., Oberndorf, T., et al. (2000). Developing new processes for COTS-based systems. *IEEE Software, 17*(4), 48–55.

Broy, M., Krüger, I. H., et al. (2007). Engineering automotive software. *Proceedings of the IEEE, 95*(2), 356–373.

Brugger, R. (2009). *Der IT business case*. Heidelberg: Springer.

Bruton, N. (2002). *How to manage the IT helpdesk*. Routledge: Chapman & Hall.

Buttle, F. (2008). *Customer relationship management*. Routledge: Taylor & Francis.

Buyya, R., Yeo, C. S., et al. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599–616.

Cannon, D., & Wheeldon, D. (2011). *ITIL service operation*. The Stationery Office.

Cabinet Office. (2011). *ITIL service strategy*. The Stationery Office.

Cagan, M. (2008). *Inspired: How to create products customers love*. SVPG Press, Sunnyvale, CA, USA.

Calvin, R. J. (2004). *Sales management*. McGraw-Hill Professional, New York, NY, USA

Carlshamre, P., Sandahl, K., et al. (2001). An industrial survey of requirements interdependencies in software product release planning. *5th IEEE international symposium on requirements engineering*, Toronto.

Carmel, E., & Becker, S. (1995). A process model for packaged software development. *IEEE Transactions on Engineering Management, 42*(1), 50–61.
Carr, M. J., Konda, S. L., et al. (1993). Taxonomy-based risk identification. Technical Report, CMU/SEI-93-TR-6, Software Engineering Institute, Carnegie Mellon University.
Chesbrough, H. W. (2003). The era of open innovation. *MIT Sloan Management Review, 44*(3), 35–41.
Choudhary, V. (2007). Software as a service: Implications for investment in software development. *40th Hawaii international conference on system sciences (HICSS'07),* Hawaii.
Classen, A., Heymans, P., et al. (2008). What's in a feature: A requirements engineering perspective. *11th international conference on fundamental approaches to software engineering*, Budapest, Hungary.
Clements, P., & Northrop, L. (2001). *Software product lines: Practices and patterns*. Boston, MA: Addison-Wesley Professional.
CMMI Product Team. (2010). CMMI for development. Version 1.3, Carnegie Mellon University.
Cockburn, A. (2001). *Writing effective use cases*. Boston: Addison-Wesley Professional.
Cohn, M. (2004). *User stories applied*. Pearson Education, Boston, MA, USA.
Condon, D. (2002). *Software product management*. Aspatore Books, Eagan, MN, USA.
Cooper, R. (2001). *Winning at new products: Accelerating the process from idea to launch*. B&T.
Cooper, R. G., Edgett, S. J., et al. (1999). New product portfolio management: Practices and performance. *Journal of Product Innovation Management, 16*(4), 333–351.
Cusumano, M. (2004). *The business of software*. New York: Free Press.
Davis, A. (2005). *Just enough requirements management*. New York: Dorset House Publishing.
De Haes, S., & Van Grembergen W. (2004). IT governance and its mechanisms. *Information Systems Control Journal 1*. See: http://www.isaca.org/Journal/Past-Issues/2004/Volume-1/Pages/IT-Governance-and-Its-Mechanisms.aspx
Denne, M., & Cleland-Huang, J. (2003). *Software by numbers: Low-risk, high-return development*. Upper Saddle River, NJ: Prentice-Hall.
Dver, A. (2007). *Software product management essentials*. Tampa, FL: Anclote Press.
Ebert, C. (2006). The impacts of software product management. *Journal of Systems and Software, 80*, 850–861.
Ebert, C. (2009). Software product management. *Crosstalk, 22*(1), 15–19.
Ebert, C., & Salecker, J. (2009). Embedded software – Technologies and trends. *IEEE Software, 26*(3), 14–18.
Ebert, C., & Dumke, R. (2007). *Software measurement: Establish – extract – evaluate – execute*. Berlin: Springer.
Edwards, K., & Grinter, R. (2001). *At home with ubiquitous computing: Seven challenges*. Atlanta: Ubiquitous Computing (UBICOMP).
El Emam, K. (2005). *The ROI from software quality*. Boston, MA: Auerbach Publications.
Fenn, J., & Raskino, M. (2011). *Understanding Gartner's hype cycles*. Gartner.
Flammia, G., & McCandless, M. (1997). From software to service: The transformation of shrink-wrapped software on the internet. *IEEE Expert, 12*(2), 4–6.
Fricker, S., Gorschek, T., et al. (2010). Handshaking with implementation proposals: Negotiating requirements understanding. *IEEE Software, 27*(2), 72–80.
Fricker, S., & Schumacher, S. (2012). Release planning with feature trees: Industrial case. *Requirements engineering: Foundations for software quality (RefsQ 2012),* Essen, Germany.
Galen, R. (2004). *Software endgames: Eliminating defects, controlling changes, and the countdown to on-time delivery*. New York: Dorset House.
Garcia, R., & Calantone, R. (2002). A critical look at technological innovation typology and innovativeness terminology: A literature review. *The Journal of Product Innovation Management, 19*(2), 110–132.
German, D. M., Weber, J. H., et al. (2010). Lawful software engineering. *FSE/SDP workshop on the future of software engineering (FoSER 2010)*, Santa Fe, New Mexico.
GNU. (2007). GNU General Public License. GNU. http://www.gnu.org/licenses/gpl-3.0

Goffin, K., & New, C. (2001). Customer support and new product development – An exploratory study. *International Journal of Operations & Production Management, 21*(3), 275–301.
Gold, A. H., Malhotra, A., et al. (2001). Knowledge management: An organizational capabilities perspective. *Journal of Management Information Systems, 18*(1), 185–214.
Gold, N., Mohan, A., et al. (2004). Understanding service-oriented software. *IEEE Software, 21*(2), 71–77.
Gorchels, L. (2006). *The product manager's handbook*. New York: McGraw-Hill.
Gorschek, T., & Wohlin, C. (2006). Requirements abstraction model. *Requirements Engineering, 11*(1), 79–101.
Gorschek, T., Fricker, S., et al. (2010). A lightweight innovation process for software-intensive product development. *IEEE Software, 27*(1), 37–45.
Griffin, A., & Hauser, J. (1996). Integrating R&D and marketing: A review and analysis of the literature. *Journal of Product Innovation Management, 13*, 191–215.
Haines, S. (2009). *The product manager's desk reference*. New York: McGraw-Hill.
Hauser, J., & Clausing, D. (1988). The house of quality. *Harvard Business Review, 66*(3), 63–73.
Hayes, B. (2008). Cloud computing. *Communications of the ACM, 51*(7), 9–11.
Helferich, A., Schmid, K., et al. (2006). Product management for software product lines: An unsolved problem? *Communications of the ACM, 49*(12), 66–67.
Henderson, B. D. (1979). *Henderson on corporate strategy*. Abt Books, Cambridge, MA, USA.
Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Amsterdam: Addison-Wesley Longman.
Hyland, P., & Beckett, R. (1994). Engendering an innovative culture and maintaining operational balance. *Journal of Small Business and Enterprise Development, 12*(3), 336–352.
Iansiti, M., & Levien, R. (2004). *The keystone advantage*. Boston: Harvard Business School Press.
IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology 610.12-1990.
ISPMA, International Software Product Management Association. (2012). Software product management body of knowledge (SPMBoK). Retrieved July 1, 2012, from http://ispma.org/spmbok/
Jaccheri, L., & Torchiano, M. (2002). Classifying COTS products. *7th European conference on software quality (ECSQ 2002)*, Helsinki, Finland.
Jansen, S., Brinkkemper, S., et al. (2007). *Providing transparency in the business of software: A modeling technique for software supply networks*. Virtual Enterprises and Collaborative Networks.
Jansen, S., & Brinkkemper S. (2006). Ten misconceptions about product software release management explained using update cost/value functions. *1st international workshop on software product management (IWSPM'06)*, Minneapolis, MS, USA.
Johnson, M. D., & Gustafsson, A. (2000). *Improving customer satisfaction, loyalty, and profit: An integrated measurement and management system*. New York: Wiley.
Jones, C. (2008). *Applied software measurement: Global analysis of productivity and quality*. New York: McGraw Hill.
Kaplan, R. S., & Norton, D. P. (1992). The balanced scorecard: Measures that drive performance. *Harvard Business Review, 70*(1), 172–180.
Karlsson, L., Dahlstedt, Å., et al. (2007). Requirements engineering challenges in market-driven software development – An interview study with practitioners. *Information and Software Technology, 49*, 588–604.
Kaulio, M. A. (1998). Customer, consumer and user involvement in product development: A framework and a review of selected methods. *Total Quality Management, 9*(1), 141–149.
Khurum, M., Gorschek, T., et al. (2011). *A homogeneous and consolidated view of software value. Decision support for product management of software intensive products*. M. Khurum, Blekinge Institute of Technology, Doctoral Dissertations Series (Vol. 12), 2011.
Kilpi, T. (1997). Product management challenge to software change process: Preliminary results from three SMEs experiment. *Software Process Improvement and Practice, 3*(3), 165–175.

Kim, J. & Park, Y. (2010). Identifying a new service opportunity from potential needs: User-centric service map. *IEEE international conference on industrial engineering and engineering management (IEEM 2010)*, Macao.
Kitchenham, B., & Pfleeger, S. L. (1996). Software quality: The elusive target. *IEEE Software, 13*(1), 12–21.
Kittlaus, H.-B., & Clough, P. (2009). *Software product management and pricing*. New York: Springer.
Konig, S. J. (2009). Finance as a stakeholder in product management. *Third international workshop on software product management (IWSPM 2009),* Atlanta, GA.
Kotler, P., Armstrong, G. (2011). *Principles of marketing*. Pearson Prentice Hall. Upper Saddle River, NJ, USA.
Krishna, S., Sahay, S., et al. (2004). Managing cross-cultural issues in global software outsourcing. *Communications of the ACM, 47*(4), 62–66.
Kuczmarski, T. D. (2000). Measuring your return on innovation. *Marketing Management, 9*(1), 24–32.
Lawley, B. (2007). *Expert product management*. HappyAbout.info.
Lee, E. (2000). What's ahead for embedded software? *Computer, 33*(9), 18–26.
Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE, 68*(9), 1060–1076.
Lehmann, D., & Winer, R. (2005). *Product management*. Burr Ridge, IL: McGraw-Hill.
Lehtola, L., & Kauppinen, M. (2006). Suitability of requirements prioritization methods for market-driven software product development. *Software Process Improvement and Practice, 11*, 7–19.
Lehtola, L., Kauppinen, M., et al. (2005). Linking the business view to requirements engineering: Long-term product planning by roadmapping. *13th IEEE international conference on requirements engineering (RE'05)*, Paris, France.
Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics, 50*(2), 197–234.
Lynn, G. S., & Akgün, A. E. (2001). Project visioning: Its components and impact on new product success. *Journal of Product Innovation Management, 18*(6), 374–387.
Mangold, W. G., & Faulds, D. J. (2009). Social media: The new hybrid element of the promotion mix. *Business Horizons, 52*(4), 357–365.
Martin, R. C., & Meinik, G. (2008). Tests and requirements, requirements and tests: A möbius strip. *IEEE Software, 25*(1), 54–59.
McGrath, M. E. (2001). *Product strategy for high technology companies: Accelerating your business to web speed*. New York: McGraw-Hill.
Messerschmitt, D., & Szyperski, C. (2003). *Software ecosystem: Understanding an indispensable technology and industry*. London: The MIT Press.
Meyer, M., Zarnekow, R., et al. (2003). IT-governance: Begriff, Status quo und Bedeutung. *Wirtschaftsinformatik, 45*(4), 445–448.
Miller, K. D. (1992). A framework for integrated risk management in international business. *Journal of International Business Studies, 23*(2), 311–331.
Miller, R. (2010). Apps: Exploring the next content frontier. *EContent, 33*(5), 18–22.
Nagle, T. T., & Hogan, J. E. (2006). *The strategy and tactics of pricing: A guide to growing more profitably*. Upper Saddle River: Pearson Prentice Hall.
Open Source Initiative. (2012). Open source licenses by category. Retrieved February 12, 2012, from http://www.opensource.org/licenses/category
Pech, E. (2006). Making innovation happen. *Annual Review of Communications, 59*, 169–172.
Phaal, R., & Muller, G. (2009). An architectural framework for roadmapping: Towards visual strategy. *Technological Forecasting & Social Chane, 76*(1), 39–49.
Phaal, R., Farrukh, C., et al. (2003). Technology roadmapping – A planning framework for evolution and revolution. *Technological Forecasting and Social Change, 71*, 5–26.

Phaal, R., Farrukh, C., et al. (2007). Strategic roadmapping: A workshop-based approach for identifying and exploring strategic issues and opportunities. *Engineering Management Journal, 19*(1), 3–12.
Phippen, A., Sheppard, L., et al. (2004). A practical evaluation of web analytics. *Internet Research, 14*(4), 284–293.
Pichler, R. (2010). *Agile product management with scrum*. Addison-Wesley Pearson Education, Boston, MA, USA.
Piercy, N., & Giles, W. (1989). Making SWOT analysis work. *Marketing Intelligence & Planning, 7*(5/6), 5–7.
Pohl, K., Böckle, G., et al. (2005). *Software product line engineering: Foundations, principles and techniques*. Berlin: Springer.
Pohl, K., & Rupp, C. (2011). *Requirements engineering fundamentals: A study guide for the certified professional for requirements engineering exam – Foundation level – IREB compliant*. Rocky Nook Computing.
Popp, K. M., & Meyer, R. (2010). *Profit from software ecosystems*. Books on Demand.
Porter, M. E. (1980). *Competitive strategy*. New York: Free Press.
Porter, M. (1998). *Competitive advantage: Creating and sustaining superior performance*. New York: Free Press.
Potts, C., Takahashi, K., et al. (1994). Inquiry-based requirements analysis. *IEEE Software, 11*(2), 21–32.
Pragmatic Marketing. (2012). Pragmatic marketing framework. Retrieved February 26, 2012, from http://www.pragmaticmarketing.com/pragmatic-marketing-framework
Project Management Institute. (2004). A guide to the project management body of knowledge. PMBOK Guide. ANSI/PMI 99-001-2004.
Pruitt, J., & Grudin, J. (2003). Personas: Practice and theory. *2003 conference on designing for user experience (DUX'03)*, New York.
Raijlich, V. T., & Bennet, K. H. (2000). A staged model for the software life cycle. *Computer, 33*(7), 66–71.
Regnell, B., & Brinkkemper, S. (2005). Market-driven requirements engineering for software products. In A. Aurum & C. Wohlin (Eds.), *Engineering and managing software requirements* (pp. 287–308). Berlin: Springer.
Regnell, B., Svensson, R. B., et al. (2008). Supporting roadmapping of quality requirements. *IEEE Software, 25*(2), 42–47.
Reifer, D. J. (2002). *Making the software business case: Improvement by the numbers*. New York: Addison-Wesley.
Rhoton, J. (2010). *Cloud computing explained: Implementation handbook for enterprises*. Recursive Press, Kent, United Kingdom
Roussos, G., Marsh, A. J., et al. (2005). Enabling pervasive computing with smart phones. *Pervasive Computing, 4*(2), 20–27.
Ruffin, M., & Ebert, C. (2004). Using open source software in product development: A primer. *IEEE Software, 21*(1), 82–86.
Rus, I., & Lindvall, M. (2002). Knowledge management in software engineering. *IEEE Software, 19*(3), 26–38.
Schmidt, M. (2002). *The business case guide*. Boston: Solution Matrix.
Shapiro, C. (1989). The theory of business strategy. *The RAND Journal of Economics, 20*(1), 125–137.
Sneed, H., Hasitschka, M., et al. (2005). *Software-Produktmanagement: Wartung und Weiterentwicklung bestehender Anwendungssysteme*, dpunkt.verlag.
Spalding, G., & Case, G. (2007). *ITIL continual service improvement*. The Stationery Office.
Steinhardt, G. (2012). *PTMK action model*. Retrieved February 26, 2012, from http://www.blackblot.com/pmtk-action-model/
Svahnberg, M., Gorschek, T., et al. (2009). A systematic review on strategic release planning models. *Information and Software Technology, 52*, 237–248.

Teece, D. J., Pisano, G., et al. (1997). Dynamic capabilities and strategic management. *Strategic Management Journal, 18*(7), 509–533.

Tessarolo, P. (2007). Is integration enough for fast product development? An empirical investigation of the contextual effects of product vision. *Journal of Product Innovation Management, 24*(1), 69–82.

Trott, P. (2011). *Innovation management and new product development*. London: Prentice Hall Financial Times.

Turner, M., Budgen, D., et al. (2003). Turning software into a service. *Computer, 36*(10), 38–44.

Vähäniitty, J., Lassenius, C., et al. (2002). An approach to product roadmapping in small software product businesses. *7th international conference on software quality (ECSQ 2002)*, Helsinki, Finland, Portland, Oregon, USA.

van de Weerd, I., Brinkkemper, S., et al. (2006). On the creation of a reference framework for software product management: Validation and tool support. *International workshop on software product management*, Minneapolis.

van de Weerd, Inge, S. B., et al. (2006). Towards a reference framework for software product management. *14th IEEE international requirements engineering conference (RE'06)*. Minneapolis: IEEE Computer Society.

van der Hoek, A., & Wolf, A. (2003). Software release management for component-based software. *Software: Practice and Experience, 33*(1), 77–98.

Variant, H. (2000). Buying, sharing and renting information goods. *The Journal of Industrial Economics, 48*(4), 473–488.

Wallin, C., Ekdahl, F., et al. (2002). Integrating business and software development models. *IEEE Software, 19*(6), 28–33.

Ward, J. L., & Peppard, J. (2002). *Strategic planning for information systems*. Chichester: Wiley.

Wernerfelt, B. (1984). A resource-based view of the firm. *Strategic Management Journal, 5*(2), 171–180.

Wohlin, C., & Aurum, A. (2005). What is important when deciding to include a software requirement into a project or release. *International symposium on empirical software engineering (ISESE 2005)*, Noosa Heads, Australia.

Wu, F. F., Moslehi, K., et al. (2005). Power system control centers: Past, present, and future. *Proceedings of the IEEE, 93*(11), 1890–1908.

Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information Systems, 16*(5), 531–541.

Yourseff, L., Butrico, M., et al. (2008). Toward a unified ontology of cloud computing. *Grid computing environments workshop (GCE'08)*, Santa Barbara.