Concepts of Product Software: Paving the Road for Urgently Needed Research

Lai Xu and Sjaak Brinkkemper

Institute of Information and Computing Sciences,
Utrecht University, 3584 CH Utrecht, The Netherlands
emails:{L.Xu,S.Brinkkemper}@cs.uu.nl

Abstract. Software products are an everyday phenomenon. Yet, there are very few scientific studies reported on the engineering of software as a product in particular. This paper stipulates the urgent need for more research on product software. The various terms used for software products are reviewed and some categorizations of software products are presented. Moreover, we provide a software product development framework to position the key business domains in a product software company. From the perspective of the software product development framework, we evaluate the related work and provide directions for future research.

1 Introduction

In the early days of computing, all software that manufacturers did not provide as part of computer had to be custom-built. This started an era in which information systems were designed and developed according to the specific wishes of the customer. The first software product came about as a result of an agreement reached between IBM and the United States Department of Justice in the later 1960s to have IBM unbundled software from hardware [1]. In the 1980s, a new class of independent software vendors started to pre-build integrated software designed to fulfill a whole range of business functions, and these offerings become known as packaged software. This resulted in the creation of thousands of product software companies, of which Microsoft, SAP, Borland, and Oracle, are examples of the ones that made it into large multi-nationals with billions of revenues.

Product software accounts for substantial economic activity all over the world [2], [3]. In 2001 (1999) the total market of the product software industry was estimated to be 196 (154.9) billion USD, which is just 9% of the overall ICT spending of 2.1 trillion USD worldwide. "The product software sector is among the most rapidly growing sectors in OECD countries, with strong increases in added value, employment and R&D investments." [3]. Although this percentage of product software usage differs from country to country, the trend in many organizations is that the make-or-buy decision falls more and more in favors of purchasing of a standard software product.

Despite the economic importance of product software, there is still very limited research activity on the development of software as a product. In order to

build a strong and lasting industry that serves society with high quality products, we need to find out the true differences between the development of a software product and the development of other types of software like embedded software or tailor made information systems. Generalizing experiences from product development a body of knowledge needs to be established with theories, methods, and tools. According to our observations, we are still at the beginning of a long journey.

This paper is organized as follows. First, we present the various terms and categorizations of software products and provide our definition of software products in Section 2. Next, we discuss specifics of the software business and the differences between developing product software and tailor-made software in Section 3. In Section 4, a software product development framework is presented. Related work is discussed in Section 5. We conclude with some remarks on future work in Section 6.

2 What are software products?

When we talk about software product, terms like shrink-wrapped software, commercial off-the-shelf (COTS) software, packaged software and commercial software immediately come to mind. Other concepts like open source software and application service providers (ASP) should also be considered. In this context, we will distinguish those concepts and provide our definition of software products in Section 2.1. The categories of software products are categorized from two different perspectives in Section 2.2.

2.1 Terminology

In common literature [4], [5], [6], [7], the boundaries distinguishing the concepts of shrink-wrapped, COTS, packaged and commercial software are blurred. To understand the concept of software products, the meaning of those terms should be reviewed. Moreover, open source software and ASP's services are also put into the perspective of product software. Relationships between those software are presented in Fig. 1. In this section, we discuss many different software and software-based service. "make one, sell many" is one of the common characters that kind of software has.

Shrink-wrapped software is software on mediums that are boxed, shrink-wrapped and sold in stores. Shrink-wrapped software also implies a widely supported standard platform.

COTS software is developed for a whole market instead of individual customers. COTS software is either used as is, or moderately personalized within the bounds of the application's ability to be readily altered without changing its original functionality (e.g., modifying its appearance). A COTS product, such as an application or a component, is sold, leased, or licensed to the general public; offered by a vendor trying to profit from it; supported and evolved by the vendor, who retains the intellectual property rights; available in multiple, identical copies and used without source code modification [8].

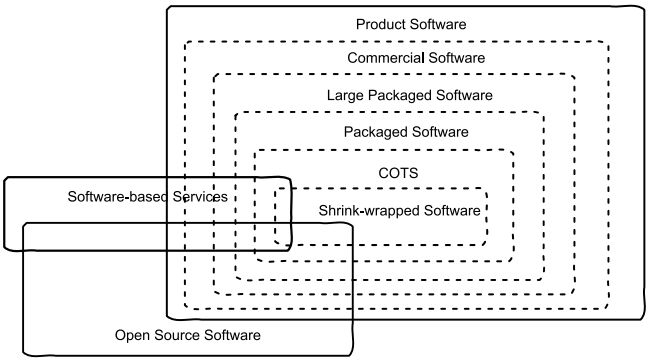

Fig. 1. Categories of product software

Packaged software describes ready-made software products that can be readily obtained from software vendors, and which generally require little modification or customization. The term today typically refers to upscale enterprise software suites, such as ERP (Enterprise Resource Planning) or CRM (Customer Relationship Management) systems. These examples of packaged software, although ready-made, rarely comes ready-to-run. Large packaged software typically requires weeks or months of deployment and implementation work to set it up for the specific needs of each individual business. Besides, there are also differences between packaged software and a separate software package [5]. We can further distinguish a *software packages* which consist of software and accompanying instructional materials. The software also possibly contains materials such as utility programs or tutorial programs, recorded on a medium suitable for delivery to the user, and from which the user can transfer the software to a data-processing device. At the same time, instructional materials may include items such as handbooks and manuals, update information, and possibly support services information.

Commercial software is software which is purchased through the retail market. Commercial software must be bought or licensed before using. Making copies of this software without the express permission of the author or controlling party is usually prohibited.

Standard software are those that, by certain consent, are routinely installed by the vendor and/or IT staff on most of computers within certain organizations. Standard software includes applications and operating systems.

Moreover, there are other concepts, such as complex COTS and customized information systems [9], that need to be addressed. A *complex COTS* is usually developed in separated products and in versions to be able to bring out new features fast and to react flexibly to a changing market [10].

Customized information system typically integrates several COTS products with other components, commercial or developed by internal-to-the-organization information technology departments or consulting/service firms [5],[6],[11],[12].

Open source software is software for which the underlying "source" code is readily available for inspection and modification by any interested person. This contrasts with most commercial software, for which the source code is a closely guarded trade secret. Advocates of open source software believe that the more eyes that can see and hands that can change the source code, the quicker "bugs" are found and fixed, and new features added and tested by the open source community. Most open source software has some type of license agreement for its use that may cover rights to modify, redistribute, use for commercial purposes, and so on. Open source software is not necessarily free in price, redistribution is generally allowed though [13]. Open source software and commercial software both can make profit. In this sense, they are overlapping. The difference is formed by freely changeable and distributable source.

ASPs also offer application software that runs behind the web servers at hosting services. Some of them are free to use; most of them are not.

In our research, we define a software product as follows.

> *A* **software product** *is defined as a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market.*

In this definition, we emphasize four concepts: "packaged components", "software-based services", "auxiliary materials", and "release and trading". "Packaged components" refer to all software discussed above which implies code, executables and web pages. "Software-based services" cover concepts like ASPs sold commercial software services. "Auxiliary materials" consists of software documentation, standards (e.g. HTML, XML etc.) which are implemented by those software, user manuals, training material, brochures and the like. Finally, the concepts of "release and trading" identifies their commercial value. Release and training activities do not include only the release of the software product into the market, but also the implementation of the customer system, training users, and related commercial activities.

2.2 Categories of software product

Software products range from small personal computer applications to large distributed systems; from common accounting systems to strategic operational systems; from commercial software to open source software. There are several classifications of software products. Robert L. Glass and Iris Vessey in [14] classified commercial products by application domain. In [15], [16], [17], [18], [19] and [20] categories of COTS software are identified by defining a set of attributes.

We present the software product classification from two different aspects: standard/semi-standard, and OECD (Organization for Economic Co-operation and Development).

Software products can also be classified according to architectural standards, language standards, and other standards /semi-standards. First, from the architectural standards view, possible architectural patterns are centralized, client-server, 2-tier, 3-tier, peer-to-peer, pipe and filter and blackboard etc [21]. In Fig.

2, the 3-tier architectural pattern is shown. An example of product software

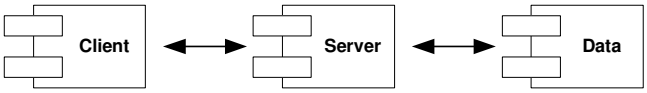

Fig. 2. 3-tier architectural pattern

with client engines is Microsoft Internet Explorer. An example of product software with server engines Microsoft Biztalk server 2000. An example of software product with architectural "data" level is Microsoft Access.

Second, from language standard aspect, there also exists a big amount of software products. For example, Sun and IBM have their Java compilers J2SE and Jikes respectively.

Finally, there are also software-based services, such as bookkeeping, project management, project hosting, version control, bug and issue tracking backups and archives, communication and collaboration, etc. For instance iTeamwork.com [22].

According to the OECD, software products can be categorized as system infrastructure software, software development tools, or application software [3]. "System infrastructure software" includes system level operating system and other software, middleware, system management, and security software. "Software development tools" contains database management systems, development environments, development life-cycle management, and internet tools. "Application software" covers ERP systems, cross-industry business software, CAD /CAM/CAE and other vertical industry business software.

3 What is specific to software business?

After explaining the terms used for software products, providing our definition of software products and presenting the categorizations of software products, we show specifics of software businesses in this section. Differences between developing product software and tailor-made software are identified in Section 3.1.

The software business is a special business where making one copy or one million copies of software product costs about the same [23]. Software investments can result in substantial productivity gain and strategic advantage [24], like in the movie, music and medicine production industries. It is also a business with up to 99% gross profit margins for its product sales.

In the software business, productivity of the best employee and the worst one has frequently up to ten- or twenty fold difference. About 75% to 80% of the product-development projects are also late and over budget [23].

Customers of software business are easier "locked in" to a particular vendor because of product decisions some one made a decade or two ago that can

not easily be reversed [23]. Those characters of software business cause some differences between producing traditional products and software products.

The traditional product flow pipeline is shown in Fig. 3. Most traditional business has one decoupling point (Point A in Fig. 3). The decoupling point is that point in the primary business process where supply and demand meet and the primary stock is located. The operations in a traditional product flow pipeline contain "suppliers", "component assembly", "system assembly", "local sales" and "buyer". The stocks are comprised of "materials", "component", "system", and "local stock".

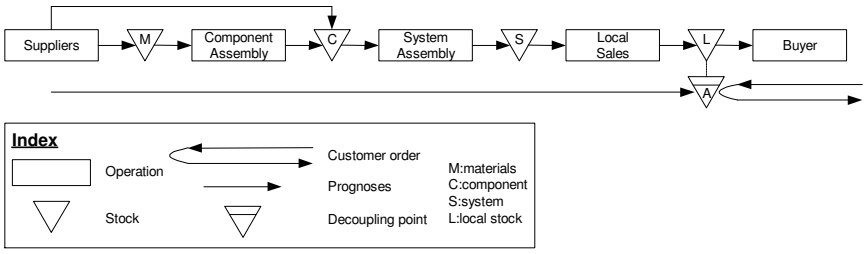

Fig. 3. Traditional product flow pipeline and its decoupling point

In traditional business, the costs of keeping stock are formed by capital costs and storage costs. While the storage costs for software are very low, capital costs for keeping stock longer than necessary are still a real issue.

The software product flow pipeline and its decoupling points are variable. Software firms produce or buy many different components for building a software product. The components need to run and be tested as a system. After all components work reasonably well together, they are packaged as products. Those packages will be sold in different countries with different functional choices which are for customized systems. During the production process, all components, prototype system, stocks of software products and local stock need places to be stored after each operation. Therefore, the operations in a software product flow pipeline are "component", "system", "product", "customized system", and "customer". The stocks in a software product flow pipeline are "library of component", "prototype system", "stock of software product" and "local stock".

As an example, a software product such as Microsoft Office or Symantec AntiVirus can be purchased in local stores as it is easy enough for most people to install by themselves. Therefore, its decoupling point (at Point A in Fig. 4) has a similar position as most traditional business. Enterprise software products like the SAP R3 ERP or Siebel CRM are not simple to purchase and deploy. After a lengthy selection and contracting phase a useable installation has to be implemented by specialists. These thus have an earlier decoupling point in the chain of activities (at Point B in Fig. 4).

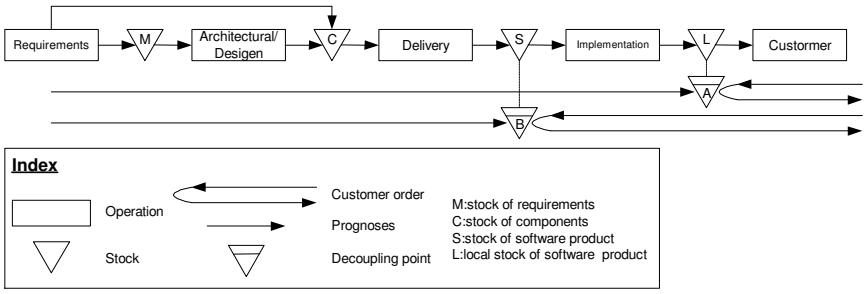

Fig. 4. Software product flow pipeline and decoupling points

3.1 Difference between software product and tailor-made software

The differences in customers, markets, and goals of software products are so diverse that comparisons are often difficult [5], [7]. Looking from a software engineering point of view, a number of basic differences between product software and tailor-made software can be observed. First market introduction requires precise synchronization of dependable software engineering activities, i.e. market oriented instead of working for one customer. Second, software products require installation and usage in different organizations, with different hardware and software platforms. Third, vendor normally stays owner of the software and auxiliary materials, while the usage is licensed to the customers. Besides, in the software product development stage, requirements engineering, design, coding, documentation and training can concur. Extensive testing and defecting handle are also key for developing software products [25].

4 Software product development framework

Software product development can be viewed from organization and development perspectives as shown in Fig. 5. From the organization perspective, a product software firm will develop its corporate strategy, product strategy and service strategy according to its target market. Process and quality control provides an overview to the production process and guarantee certain level of quality. The product and service strategies are in service of the corporate strategy. The corporate strategy mainly focuses on organization internationalization, product investment, process quality control, and resource management strategies. The product strategy concerns product lifecycle management and portfolio analysis. The service strategy includes service portfolio, market strategy, and localization and customizations.

When software product firms create their corporate strategy, internationalization can be seen as the key prerequisite for continued growth in the product software business [26]. In the software business, product investment typically

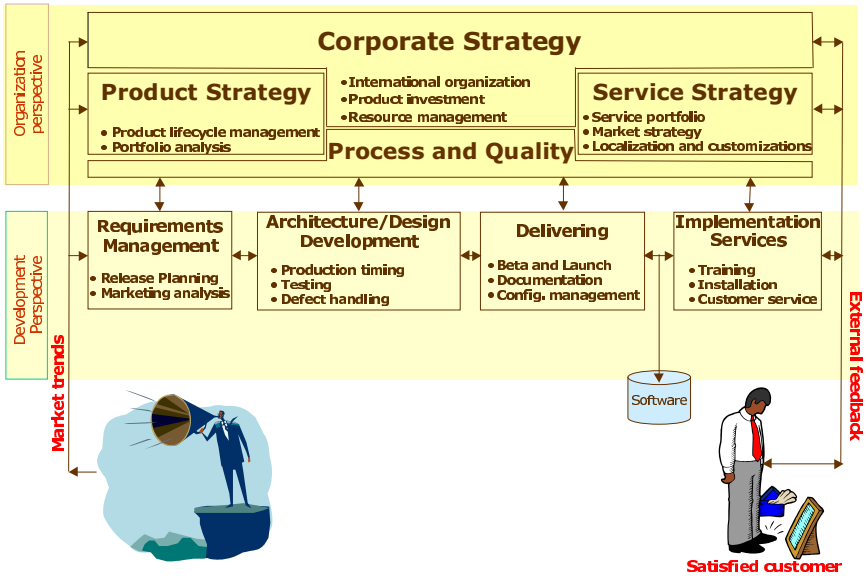

Fig. 5. Software product development framework

includes a choice of standardized products for mass markets. The internationalization and product investment is especially relevant for the resource management where technology development has traditionally been the central area of interest among managers [27], [28]. Furthermore, the ever-growing market offers opportunities for the development and growth of product software firms.

Product strategy considers the product lifecycle management and portfolio analysis. Mapping market trends to software product design is becoming the central issue in new software product development [29], [30]. It is also important to use customer feedback during the product life cycle to enhance customer satisfaction. Besides, product software firms also have a major pressure on time-to-market. For the product strategy, it is essential to prioritize the customer requirements [31] as the external pressure makes that not all requirements can be implemented initially [32]. In short, software products are often offered to a market through releases with significant increases in functionality. This requires careful release planning and requirements prioritization [33].

The service strategy is aimed at improving service quality. Researchers in the information system area have argued that customer satisfactions is critical to adoption and use of information systems [34]. For large size software products such as ERP software, product software firms can choose between building customized software systems by themselves or to outsource the customization project to vendors that perform the complete implementation activities [23], [35]. Those decisions eventually effect the service strategy of product software

firms. Process and quality control improves effective engineering work and brings substantial quality.

From the development perspective, the product software includes requirements engineering, architecture/design development, delivery and implementation services stages. Product software firms are driven by market trends. The requirements engineering stage includes planning, and market analysis. The architecture/design development includes production time control, testing, and defect handling. The delivery stage includes documentation and configuration management. Finally, the implementation service includes training, service and customer service.

Large complex software products aimed for a broad market involves many different factors. Requirements management includes capturing market trend, analyzing requirements and releasing the software product at right time. It is a key for the software business [31], [36], [37]. It is the first step to develop a software product.

Software architecture is fundamental for the development of software products. Work in software architecture can be seen as attempting to codify the structural commonality among a series of software products so that the high-level design decisions inherent in each product need not be re-invented, revalidated and re-described [21]. Under a flexible, substantial architecture, the prototypes software products can be developed and tested. Quality of the development process and the prototypes can be improved by inspection and testing defects under the common architecture.

Delivery is an important step at which software products are launched on and offered to the market. Typically delivery consist of configuration management and documentation. Software configuration management is the control of the evolution of complex systems [38]. The goal of software configuration management is to keep evolving software products under control and help satisfy delay and quality constraints. Software configuration management does not only focus on versioning [39], rebuilding [40], composition [41], and synchronized team coordination [42], but also on web supported distributed and remote development [43]. Software documentation describes the requirements of software products which need to be satisfied, the design, implementation, capabilities and limitations of the software product to make the product easier to use, maintain and reuse [44].

After more than 30 years of software development, most software applications, such as large-scale information systems, and ERP systems, are constructed by adapting existing software products. While software product firms aim to extend their solutions into as many different settings as possible, there are two choices for implementing software product into customer systems. Software product firms can do implementation (such as training, installation and custom services) by themselves or outsource this part of business to consulting firms. Both implementation choices exist at the current time [23].

All development activities such as requirements management, architecture/design, delivery and implementation services will receive feedback and adjust themselves according to information from each other and customers.

5 Related work

Looking at earlier work, some research has been focused on software processes in small-size firms [45] which includes some research in portfolio management [46], [47] and software design quality improvement [48] for small software firms. The main purposes of their research has been to report on a preliminary investigation of in small firms which have different business strategies, organization issues and product development processes from other kinds of firms. According to our software product development framework (see Section 4), the 4CC framework of software development only considers the relationship between release management and software development in a small-size software product firm. Many other business strategies such as sales and marking strategies are not considered in this research.

Research in the COTS software area has been carried out on definition and classification [15], [20], selection [17], and assessment [19]. Following our software product development framework, we can see that this research focuses only on the implementation part of the development perspective.

The center for software engineering of the University of Southern California visualize software and the process of developing software from four different models: process models, product models, property models and success models. A set of guidelines is provided to describe software engineering techniques for the creation and integration of development models for a software project [49]. The process models describe elements such as lifecycle and risk models [50]. The product models items include such as object oriented analysis and design models and traditional requirements models [51]. The property models have elements such as cost and schedule [52]. Finally, the success models involve items such as business-case analysis and stakeholder win-win. The research addresses some important issues in software product development. The four provided models however are separated with each other. We are interested in the interaction between those models.

In short, the works discussed above have mainly concentrated on small or medium-size industrial software development practices. Many different models for the improvement of software product development have been discovered. Our software product development framework is intended for all sizes of software product firms. We look from both organization and development perspectives and their relations.

6 Conclusions

In this paper, we have placed some first stones on the road for research in the area of product software. We have reviewed software product related concepts

and provided our definition of a software product. A brief overview of some of the categorizations for software products has been provided. Moreover, we presented the specific characters of software product and identified differences between software product and tailor-made software. The software product development framework provides a positioning of the research focus for both the organizational and the development perspective of a software product company.

In our future research, we will focus on software product development of all types of software product firms and consider relationships between all business strategies and software development stages.

References

1. Carmel, E.: American hegemony in packaged software trade and the culture of software. The Information Society **12** (1997) 125–142
2. for Economic Co-operation, O., Development): The software sector: Growth, structure and policy issues. OECD Report DSTI/ICCP/IE(2000)8/REV2 (2001)
3. for Economic Co-operation, O., Development): Highlights of the oecd information technology outlook 2002. OECD Report (2002)
4. Sawyer, S.: Packaged software: Implications of the differences from custom approaches to software development. European Journal of Information System **9** (2000) 47–58
5. Carmel, E., S.Sawyer: Packaged software development teams: What makes them different? Information Technology & People **11** (1998) 7–19
6. Sawyer, S.: A market-based perspective on information systems development. Communications of the ACM **44** (2001) 97–102
7. Sawyer, S., Guinan, P.J.: Software development: Processes and performance. IBM system Journal **37** (1998) 552–569
8. Brownsword, L., Oberndorf, T., C.Sledge: Developing new processes for cots-based system. IEEE Software **17** (2000) 83–86
9. Carney, D.: Assembling large systems from cots components: Opportunities, cautions, and complexities. SEI Monographs on Use of Commercial Software in Government Systems, Software Engineering Institute, Pittsburgh, USA (1997)
10. Deifel, B.: A model for version planning of ccots. In: Proceedings of the Workshop on Software Change and Evolution. (1999)
11. Carmel, E., Becker, S.: A process model for packaged software development. IEEE Transactions on Engineering Management **41** (1995) 50–61
12. Cusumano, M.A., Smith, S.A.: Beyond the waterfall : Software development at microsoft. Technical report, Massachusetts Institute of Technology (MIT), Sloan School of Management (2003)
13. Center, C.T.: (Glossary of visualization terms) http://www.tc.cornell.edu/services/edu/topics/OpenDX/glossary.html.
14. Glass, R.L., Vessey, I.: Contemporary application-domain taxonomies. IEEE Softw. **12** (1995) 63–76
15. Morisio, M., Torchiano, M.: Definition and classification of cots: a proposal. In: Proc. 1st International Conference on COTS Based Software Systems (ICCBBS'2002). (2002) 165–175
16. Carney, D., Long, F.: What do you mean by cots? finally, a useful answer. IEEE Software **17** (2000) 83–86

17. Kontio, J.: A case study in applying a systematic method for cots selection. In: ICSE '96: Proceedings of the 18th international conference on Software engineering, IEEE Computer Society Press (1996) 201–209
18. Morisio, M., Tsoukiás, A.: Iusware: a methodology for the evaluation and selection of software products. IEE Proceedings - Software **144** (1997) 162–174
19. Ochs, M., Pfahl, D., Chrobok-Diening, G., Nothhelfer-Kolb, B.: A method for efficient measurement-based cots assessment and selection -method description and evaluation results. In: IEEE METRICS. (2001) 285–
20. Jaccheri, M.L., Torchiano, M.: Classifying cots products. In: ECSQ '02: Proceedings of the 7th International Conference on Software Quality, Springer-Verlag (2002) 246–255
21. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison Wesley Longman (1998)
22. iTeamwork home page available at http://www.iteamwork.com/.
23. Cusumano, M.A.: The Business of Software. Free Press (2004)
24. Brynjolfsson, E.: The productivity paradox of information technology. Communication of the ACM **36** (1993) 66–77
25. Brinkkemper, S.: Overture propsoal. Utrecht University (2002)
26. Alajoutsijarvi, K., Mannermaa, K., Tikkanen, H.: Customer relationships and the small software firm a framework for understanding challenges faced in marketing. Information and Management **37** (2000) 153–159
27. Hilburn, T.: Software engineering education: a modest proposal. IEEE Software **14** (1997) 44–48
28. Rao, P., Klein, J.A.: Growing importance of marketing strategies for the software industry. Industrial Marketing Management **21** (1994) 29–37
29. Hauser, J., Clausing, D.: The house of quality. Harvard Business Review **66** (1988) 63–73
30. Garvin, D.A.: Managing Quality: the Strategic and Competitive Edge. Free Press (1998)
31. Natt och Dag, J., Gervasi, V., Brinkkemper, S., Regnell, B.: Speeding up requirements management in a product software company: Linking customer wishes to product requirements through linguistic engineering. In: Proceedings of the 12th International Requirements Engineering Conference, IEEE Press (2004) 283–294
32. Natt och Dag, J., Gervasi, V., Brinkkemper, S.: A linguistic-engineering approach to large-scale requirements management. IEEE Software **22** (2005) 32–39
33. Regnell, B., Höst, M., Natt och Dag, J.: An industrial case study on distributed prioritization in market-driven requirements engineering for packaged software. Requirements Engineering **6** (2001) 51–62
34. Leong, L.: Theoretical models in IS research and the technology acceptance model (TAM). Idea Group Publishing (2003)
35. Nelson, P., Richmond, W., Seidmann, A.: Two dimensions of software acquisition. Communication ACM **39** (1996) 29–35
36. Sawyer, P., I., S., Kotonya, G.: Improving market-driven re processes. In: Proceedings of International Conference on Product Focused Software Process Improvement (PROFES99). (1999)
37. Carlshamre, P., Regnell, B.: Requirements lifecycle management and release planning in market-driven requirements engineering processes. In: DEXA Workshop. (2000) 961–965
38. Estublier, J.: Software configuration management: a roadmap. In: ICSE '00: Proceedings of the Conference on The Future of Software Engineering, ACM Press (2000) 279–289

39. Wingerd, L., Seiwald, C.: Constructing a large product with jam. In: ICSE '97: Proceedings of the SCM-7 Workshop on System Configuration Management, Springer-Verlag (1997) 36–48
40. Feldman, S.I.: Make-a program for maintaining computer programs. Software - Practice and Experience **9** (1979) 255–65
41. Tryggeseth, E., Gulla, B., Conradi, R.: Modeling systems with variability using the proteus configuration language. In: In Jacky Estublier (Ed.): Software Configuration Management - ICSE SCM-4 and SCM5-5 Workshops, Selected Papers, Springer Verlag (1995) 216–240
42. Buffenbarger, J.: Syntactic software merging. In: Selected papers from the ICSE SCM-4 and SCM-5 Workshops, Software Configuration Management (1995) 153–172
43. WebDav: Http extensions for distributed authoring (1999) RFC 2518http://andrew2.andrew.cmu.edu/rfc/rfc2518.htm.
44. Forward, A.: Software documentation – building and maintaining artifacts of communication. (2002)
45. Rautiainen, K., Lassenius, C., Sulonen, R.: 4cc: A framework for managing software product development. Engineering Management Journal **14** (2002) 27–32
46. Vähäniitty, J., K., R.: Towards an approach for portfolio management in small product-oriented software companies. In: Proceedings of Hawaii International Conference on System Sciences (HICSS-38), IEEE Computer Society (2005) 1–10
47. Vähäniitty, J.: Product portfolio management in small software product businesses - a tentative research agenda. In: Proceedings of the 6th International Workshop on Economic-Driven Software Engineering Research (EDSER-6) at ICSE 2004. (2004)
48. Mäntylä, M.V.: Developing new approaches for software design quality improvement based on subjective evaluations. In: Proceedings of the International Conference on Software Engineering (doctoral symposium). (2004) 48–50
49. Boehm, B., Port, D., Al-Said, M.: Avoiding the model clash spiderweb. IEEE Computer **33** (2000) 120–122
50. Boehm, B., Brown, W., Basili, V., Turner, R.: Spiral acquisition of software-intensive systems of systems. CrossTalk (2004) 4–9
51. Kitapci, H., Boehm, B.W., Grunbacher, P., Halling, M., Biffl, S.: Formalizing informal stakeholder requirements inputs. In: Proceedings INCOSE 2003 conference. (2003)
52. Huang, L., Boehm, B.: Reasoning about the value of software dependability: the idave model. In: Proceedings 14th. IEEE International Symposium on Software Reliability Engineering (ISSRE). (2003)