8-7-2011

The Software Value Chain as an Analytical Framework for the Software Industry and Its Exemplary Application for Vertical Integration Measurement

Anton Pussep
Technische Universität Darmstadt, Pussep@is.tu-darmstadt.de

Markus Schief
Technische Universität Darmstadt, Schief@is.tu-darmstadt.de

Thomas Widjaja
Technische Universität Darmstad, widjaja@is.tu-darmstadt.de

Peter Buxmann
Technische Universität Darmstadt, buxmann@is.tu-darmstadt.de

Christian M. Wolf
Software Economics Group Darmstadt, Cmw@christianmichaelwolf.de

The Software Value Chain as an Analytical Framework for the Software Industry and Its Exemplary Application for Vertical Integration Measurement

Anton Pussep
Technische Universität Darmstadt
Pussep@is.tu-darmstadt.de

Markus Schief
Technische Universität Darmstadt
Schief@is.tu-darmstadt.de

Thomas Widjaja
Technische Universität Darmstadt
Widjaja@is.tu-darmstadt.de

Peter Buxmann
Technische Universität Darmstadt
Buxmann@is.tu-darmstadt.de

Christian M. Wolf
Software Economics Group Darmstadt | Munich
Cmw@christianmichaelwolf.de

ABSTRACT

The value chain concept disaggregates a firm into the various activities it performs. Abstracting from the firm-level this concept has also been applied to industries as a whole. In this paper we conceptualize a software specific value chain and provide a first proof of concept. Our approach aggregates and unifies findings from a literature review on industry-level value chains, software value chains, and related concepts. The resulting unified software value chain comprises eleven activities: product research, component procurement, product development, user documentation, production and packaging, marketing, implementation, training and certification, maintenance and support, operations, and replacement. A first proof of concept is provided through expert interviews with software firms. Furthermore, we present an example that shows how the software value chain can be applied to measure the degree of vertical integration in the software industry.

Keywords

software industry, value chain, software value chain, vertical integration, degree of vertical integration, vertical integration measurement

INTRODUCTION

The concept of the value chain was initially introduced by Porter (1985) as a tool for developing and sustaining competitive advantage of a firm. By disaggregating a firm into its various activities, the value chain allows for a better understanding of costs behavior and sources of differentiation. Activities are defined as "the physically and technologically distinct activities a firm performs … by which a firm creates a product valuable to its buyers" (Porter, 1985). Thus, the focus is on value generating activities, whereas value can be defined as "the perceived worth in monetary units of the set of economic, technical, service and social benefits received by the customer firm in exchange for the price paid for a product offering, taking into consideration the available suppliers' offerings and prices" (Anderson, Jain and Chintagunta, 1993). Notably, value comprises the perceived worth, which might well differ from the actual economic value.

Even though the value chain was developed for single firms and business units, it can be applied to entire industries as well (Barnes, 2002; Feng and Whalley, 2002; Messerschmitt and Szyperski, 2003; Stanoevska-Slabeva, Talamanca, Thanos and Zsigri, 2007). An obvious way to do so is by applying the analysis to an abstract typical firm representing the industry. Such a value chain would contain typical activities performed within an industry. This broader view is important since competition has been shifting from the firm level to the network level (Parolini, 1999; Spekman, Kamauff and Salmond, 1994). Furthermore, looking beyond the firm value chain can be an important source of innovation (Pil and Holweg, 2006).

Several works have modified the original concept and applied it to industry analysis. Barnes (2002) develops the mobile commerce value chain and assigns major technologies and key players to the various activities. The result of his value chain analysis is a documentation of how value is created in mobile commerce and by whom. Feng and Whalley (2002) show that the telecommunication value chain is increasingly disaggregated. As a result, firms can choose among multiple possible

combinations of value activities, whereas the overall value chain becomes more complex. Stanoevska-Slabeva et al. (2007) perform a literature review of the grid industry. They also conduct 18 case studies and present an aggregated value network (synonyms are value grid, value web, or business web). The concept of the value network was proposed by Pil and Holweg (2006) as a way to break up linear thinking in value chains and identify business opportunities by looking at parallel value chains and their interrelations.

The key benefit of the value chain concept is its simplicity and high-level view on a firm or industry providing a simple model of the activities performed. However, there are various limitations, which should be taken into account when performing analyses based on value chains. Most of those limitations result from the fact that no clear-cut rules can be stated on such high-level concepts as industry or activity frontiers. Depending on the point of view the value chain might include or exclude another upstream or downstream activity. The granularity is up to individual judgment as well, since activities can be enriched or condensed by aggregation or disaggregation. The final decision should depend on the goal of the analysis and the target group (Porter, 1985). An important limitation is that the ordering of activities as shown by the value chain is no indication of the actual chronological order: "ordering of activities should broadly follow the process flow, but ordering is judgmental as well" (Porter, 1985). Thus activities shown might be executed in many ways, including parallel execution or exclusion of some activities. According to these limitations value chains of the same firm might be depicted in various forms. Analyses performed on different value chains might lead to different results and, thus, become incomparable.

This article focuses on the software industry which is defined as the sum of all organizational units whose core competence includes the creation of software products in its broadest sense from product idea to product replacement. Software itself is "a list of commands and instructions for data-processing" (Engelhardt, 2008). The explicit treatment of the software industry is justified by the size and growth of this industry. Gartner (2011) forecasts software revenues to surpass $ 254 billion in 2011, an increase of 7.5 percent from 2010. Carmel (2003) emphasizes the role of the software industry for developing countries as well as stimulating externalities stemming from spillovers. Moreover, software is different in comparison to other goods. Messerschmitt and Szyperski (2003) point out that software shares many concepts with economics of information and standardization. To name a few those include intangibility, network effects, lock-in, economies of scale, and low distribution costs. Network effects occur when the utility of a good depends on the size of the network of users (Katz and Shapiro, 1994). High switching costs result in a lock-in and therefore a high dependence on a once-implemented solution, making other solutions and providers inefficient (Shapiro and Varian, 1999). Whereas creation costs are very high, the replication costs are low resulting in high economies of scale and low distribution costs (Messerschmitt and Szyperski, 2003).

This article provides two contributions to the software industry research: the concept of the unified software specific value chain and a first empirical proof of concept. Based on a literature review on industry-level value chains, software value chains, and related concepts, we conceptualize the unified software value chain. Our first proof of concept is based on expert interviews with five software firms, which used the proposed value chain as a tool to depict which activities are performed through market or hierarchy by their firms. Finally, an exemplary application of the value chain is presented where we use the data from the expert interviews for the measurement of the degree of vertical integration in those software firms and calculate the average degree of vertical integration for the software industry.

This paper is organized as follows. Section 2 derives requirements on value chains based on a literature study. Section 3 evaluates software value chains and related concepts and shows shortcomings with regard to the derived requirements. As a result, a new software value chain is conceptualized in section 4. We provide a first empirical proof of concept through expert interviews and measure the degree of vertical integration by utilizing the unified software value chain in section 5. Finally, section 6 concludes the paper.

GENERIC REQUIREMENTS ON VALUE CHAINS

We start by explicitly outlining the requirements, which a value chain should fulfill in order to enable industry and firm analysis. There is little literature dealing with value chain requirements, therefore we suggest the adoption of principles from related fields such as software development processes. The proposed requirements can be separated in two groups. The first group is concerned with the range of the value chain, stating where the value chain starts and where it ends. The second group deals with the isolation and separation of activities within the value chain.

In order to identify the boundaries of a value chain we suggest focusing on two requirements. First, the value chain should contain all important activities of a generic firm competing in the corresponding industry. Second, the value chain should range from the very beginnings of the corresponding product up to its operation and final replacement.

The second group of requirements has been addressed by Porter (1985) who suggested that "the basic principle is that activities should be isolated and separated that (1) have different economics, (2) have a high potential impact of

differentiation, or (3) represent a significant or growing proportion of cost". Additional requirements can be identified when applying basic principles from software engineering. Software systems are often decomposed in components, with components themselves hierarchically composed of finer-grained components (Messerschmitt and Szyperski, 2003). Components should be separated such that coupling (interdependence between components) is low and cohesion (binding of the elements in a component) is high, resulting in higher software quality (Dhama, 1995). However, those generic principles do not provide a single level of granularity: "defining granularity is quite complex since it cannot draw on theoretical groundings ... granularity can hardly be measured in terms of absolute numbers, because of the subjectivity of the related concepts that may determine the granularity in question" (Haesen, Snoeck, Lemahieu and Poelmans, 2008). Therefore, there is no single definition of a component (Cusumano, 2004). We suggest that the view of hierarchical decomposition also applies to value chains, where coarse-grained activities comprise finer-grained activities. To our knowledge the application of software engineering principles has not been proposed before. In the context of value chains the principle of high cohesion states that comprising sub-activities within an activity should be highly related to each other. The principle of low coupling states that dependence between different activities should be low, such that they have as little impact on each other as possible. Notably, linkages between activities are inherited in hierarchical decomposition: aggregating highly linked sub-activities into distinct activities will result in high coupling between those activities as well. Finally, the appropriate granularity depends on the respective research purpose. While a single activity might envelop the entire value chain, activities can also be defined on a very fine-grained level, resulting in a huge number of activities.

EVALUATION OF SOFTWARE VALUE CHAINS AND RELATED CONCEPTS

The generic value chain as proposed by Porter (1985) consists of the five primary activities (1) inbound logistics, (2) operations, (3) outbound logistics, (4) marketing and sales, and (5) service. Primary activities are supported by firm-wide support activities. The visual size of the value chain is determined by the value generated. A margin is added to the visual representation. Evaluating the original concept, it appears that little is left of the generic value chain. All work reviewed did not take supporting activities or the margin into account and no mapping to the generic primary activities has been done (Barnes, 2002; Feng and Whalley, 2002; Messerschmitt and Szyperski, 2003; Stanoevska-Slabeva et al., 2007). Concerning the proposed requirements the generic value chain does not take into account software specific activities. Furthermore, activities such as outbound logistics are of little importance in context of software due to its intangibility and low distribution costs.

Messerschmitt and Szyperski (2003) propose two software specific value chains. They term them requirements and supply value chain. The requirements value chain encompasses the activities (1) analysis and design, (2) implementation, (3) provisioning, and (4) operation. The supply value chain encompasses the activities (1) implementation, (2) provisioning, (3) operation, and (4) use. Another distinction made by the authors is the discrimination of application and infrastructure which is important because the use activity does not apply to infrastructure software since there are no users involved. The inclusion of the use activity seems questionable, since software is usually applied in a different industry, such that the value created should be attributed to the industry where the usage takes place. Also the distinction between the two value chains seems questionable when looking at the industry as a whole. For instance, the value chains overlap in three activities. Finally, the authors themselves note that "other standard business functions like sales [are not] specifically discussed" (Messerschmitt and Szyperski, 2003).

Software engineering processes focus on the software development, "which refers to the range of activities ... surrounding the creation of working software programs" (Messerschmitt and Szyperski, 2003). A broader view is taken by lifecycle processes which "include all activities of a product or a system during its entire life, from the business idea for its development, through its usage and its completion of use" (Crnkovic, Larsson and Chaudron, 2005). Both works refer to the waterfall model. As initially proposed by Royce (1970), the process of the waterfall model consists of seven phases: (1) system requirements, (2) software requirements, (3) analysis, (4) program design, (5) coding, (6) testing, and (7) operations. Extended models for instance include phases such as conceptualization and upgrade (Messerschmitt and Szyperski, 2003). We propose that those activities can be viewed as activities in the software value chain. However, since the focus is rather technical, the waterfall model misses downstream activities such as marketing and operations. Furthermore, the detailed technical point of view blows up the number of upstream activities which should be reduced in the unified software value chain. Other related concepts are the software product development framework (Xu and Brinkkemper, 2007) and the software supply network (Slinger, Brinkkemper, and Finkelstein, 2007).

CONCEPTUALIZING THE UNIFIED SOFTWARE VALUE CHAIN

Works proposing unified value chains are usually based on literature reviews (Barnes, 2002; Feng and Whalley, 2002). Stanoevska-Slabeva et al. (2007) outline their research approach in three steps: broad literature review, 18 case studies, and aggregation to a generic value network. Porter (1985) just recommends: "starting with the generic chain, individual value activities are identified in the particular firm". It becomes apparent that there is little guidance on how to propose a unified value chain. We suggest the following recursive approach:

1. Definition of the industry and its frontiers.
2. Listing of value activities of proper granularity and naming entailed sub-activities.
3. Ordering of the value activities in a visual representation of the value chain.

As has been shown in the previous section, all concepts of the software value chain presented so far do not fulfill major requirements such as listing software specific activities or encompassing the entire industry. Therefore, we combine those approaches and propose a unified software value chain following the outlined approach.

Our definition of the software industry entails the sum of all organizational units whose core competencies include the creation of software products. We employ a broad understanding of the value creation process, such that the boundaries stretch from the business idea to the disposal and replacement of the product.

The second step is built on the literature review performed in the previous section. Whereas the initial waterfall model provides thorough insight into upstream activities, the extended waterfall model and lifecycle process model allow for a better understanding of downstream activities. Furthermore, we use Standard Industry Classification (SIC) code descriptions for additional software activity definitions. Past works scrutinizing the software industry used SIC codes 7371, 7372 and 7373 (Léger and Yang, 2005; Léger and Quach, 2009). SIC code 7379 focuses on software services and code 8243 on establishments training users in the use of software, thus we decided to include those as well. Based on this combination of sources the single activities are separated and defined as follows:

1. The product research activity stems from the conceptualization activity, which comprises the development of a product vision (Messerschmitt and Szyperski, 2003). We extend it by fundamental research and first feasibility checks. While fundamental research implies developing profound concepts and methods, the aim in later activities is to integrate those concepts into software products and to enable sophisticated software solutions.
2. The component procurement activity is a result of the generic value chain and the major role of components in software creation (Crnkovic et al., 2005). The procurement of components is similar to the inbound logistics activity from the generic value chain. We suggest that component procurement should be listed separately, since its economics are different from other activities, which are not related to procurement. It covers the sub-activities of selection, purchase, adaptation, verification, and validation of components in a broad context.
3. The product development activity entails most of the phases which can be found in software development processes and are concerned with the actual creation of the software product itself. Product development is the core activity of a software producing firm and consists of the sub-activities requirements engineering, software design, software development, code documentation, verification, and validation.
4. The user documentation activity involves sub-activities which document software functionalities and properties for end-users. Actuality, comprehensiveness, and usefulness for end-users need to be ensured via continuous reviews. The results of this activity are text files, which can be processed electronically or printed in user manuals. We propose regarding user documentation as a separate activity, because it is very different from the technical product development and does not have to be carried out by technical personnel.
5. The production and packaging activity comprises the sub-activities assembly, production, and packaging. Within assembly, software and respective documentation are bundled to one package and, if applicable, combined with other products. Production implies the built of physical media that contain the software product. Packaging, finally, bundles the artifacts in a physical package. The result is a product with all attributed artifacts, which is ready for shipment. The final product can have both, a physical or an intangible appearance. We suggest production and packaging to be a separate activity because it involves physical sub-activities, thus making it very different from other activities in the software creation process.
6. The marketing activity is contained in the generic value chain and entails sub-activities "associated with providing a means by which buyers can purchase the product and inducing them to do so, such as advertising, promotion, sales force, quoting, channel selection, channel relations, and pricing" (Porter, 1985).

7. The implementation activity entails the installation, configuration, and adaptation of the product. The installation comprises the transmission of the packaged binaries to the customer's information system. Moreover, it ensures that the binaries can be executed without runtime errors. Configuration allows the setting of software parameters and software modifications according to the customer's needs. Finally, adaptations can be performed that modify or enhance the functionality of the software product and employ business process changes. We identify the implementation activity from the provisioning stage in the Messerschmitt and Szyperski (2003) value chain.
8. The training and certification activity entails training of users and third party firms. In addition, certifications attest users and third party firms a certain degree of seniority in the handling of a software product. Whereas Messerschmitt and Szyperski (2003) name training as a part of the provisioning phase, we think of it as a separate activity as proposed by SIC code 8243 (OSHA, 2011).
9. The maintenance and support activity has been identified from the waterfall model (Crnkovic et al., 2005; Messerschmitt and Szyperski, 2003). In the scope of maintenance, the software product is updated by bug fixes and enhancements. Similar to product development, the sub-activities of maintenance involve requirements engineering, software design, software development, verification, and validation. The deployment of patches does not belong to maintenance. Tightly aligned with maintenance, support can be differentiated in customer service and technical support. While the first sub-activity deals with the support of users, the second activity comprises the collection of error messages and ideas for enhancements.
10. The operations activity ensures the execution of a product on an information system (Messerschmitt and Szyperski, 2003; Royce, 1970). By monitoring the system behavior can be analyzed and supervised. To minimize damages through data loss, regular data back-ups need to be planned, run, and administered. Finally, the information system needs to be upgraded to new releases during its lifecycle.
11. The replacement activity is similar to the disposal activity as described by Crnkovic et al. (2005). Replacement comprises the sub-activities migration and shut-down. Primarily, replacement deals with the decision if a legacy system shall be replaced by an alternative system. If the decision for an alternative is made, data needs to be migrated from the legacy to the new system. Subsequently, the legacy system is shut-down. A seamless transition to the new system is the main target at this stage. After the irrevocable data destruction of confidential information, the shut-down activity is completed.

Finally, we visualize the value chain as shown in Figure 1.

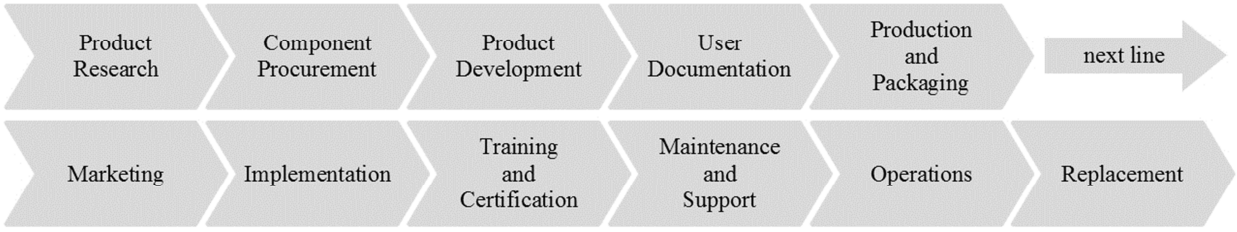

Figure 1: The unified software value chain consists of eleven activities. The line break between product and packaging and marketing was inserted due to place restrictions.

FIRST EMPIRICAL PROOF OF CONCEPT

As a first proof of concept for the value chain presented above we conducted expert interviews with five software firms from Germany. Our sample contained three big IT consulting firms producing individual software for their customers. Those were selected due to their great insight into the software industry, which extends well beyond the mere software development stages. The other two firms were one big and one small standard software producer, therefore reflecting different firm sizes in our sample. Each conducted semi-structured interview lasted about one hour. Three of them were performed via telephone and two face-to-face. The firms have been provided with a management summary of our main results.

We presented the software value chain as shown above plus sub-activities to the interviewees. We asked them (1) whether they agree with the presentation, (2) which activities they perform and (3) how those activities are performed: via market, firm hierarchy or a mixture of both (Williamson, 1991), thus revealing their make-or-buy strategies for every activity. All interviewees agreed with the presentation as shown in Figure 1, even though some of the activities had to be explained in more detail by the interviewer. Also all interviewees confirmed that their firms performed all of the activities. The results of

our third question are presented in Table 1. Thus, the proof of concept is not just based on the approval of the interviewees, but also on the actual application by the industry experts to their own firms. We regard this as a sufficient first proof of concept, even though a full validation should entail more interviews utilizing the software value chain.

Firm	Activities											Degree of vertical integration (%)
	1	2	3	4	5	6	7	8	9	10	11	
A	●	●	●	●	●	◖	◖	●	●		●	82
B	◖	◖	◖	◖	◖	◖	●	●	●	●	●	73
C	◖	◖	◖	◖	◖	●	●	●	●	●	●	77
D	●	●		●	●	●	●	●	●		●	82
E	●	◖	●	●	●	◖			●	●	◖	68

Table 1: Results of the expert interviews for each firm (A to E). Full circles denote activities carried out by firm hierarchy. Empty cells denote activities carried out by market and half circles denote a mixture of market and firm hierarchy.

The data presented so far can be used for instance for measuring the degree of vertical integration. Vertical integration occurs when two distinct activities are performed within the boundaries of a single firm (Porter, 1985). The degree of vertical integration is a key figure which is high when a firm performs many activities itself and low when a firms concentrates on just a few activities. There are many measures of vertical integration (Adelman, 1955). Rothaermel, Hitt and Jobe (2006) propose a simple measurement method based on value chain activities and apply it to the microcomputer industry. Given a value chain of five activities the degree of vertical integration is calculated as an index between 1 and 5, depending on how many activities are performed within the boundaries of a firm. We are not aware of comparable results for the software industry, thus we propose a calculation method which is based on the work of Rothaermel, Hitt and Jobe (2006):

$$degree\ of\ vertical\ integration = \frac{n_{hierarchy} + \frac{n_{mixed}}{2}}{n_{activities}} \quad \text{with} \quad n_{activities} = n_{hierarchy} + n_{mixed} + n_{market}$$

Thus, if company A performs 8 activities by the company hierarchy ($n_{hierarchy} = 8$), 1 activity by utilizing the market ($n_{market} = 1$) and 2 activities partly through hierarchy and partly through market ($n_{mixed} = 2$), the numerator sums up to 9.0 and the denominator to 11, resulting in a degree of vertical integration of 82 percent. There are strong limitations underlying this calculation formula. First, it is assumed that all activities are of the same "size". Second, a mixed activity is assumed to be carried out half-and-half between market and company hierarchy. Table 1 shows our calculations for the companies we conducted interviews with. As a result, the overall degree of vertical integration in our sample is 76 percent. The values range from 68 to 82 percent.

CONCLUSION

This article provides two contributions to the software industry research: the concept of the unified software value chain and a first empirical proof of concept. The unified software value chain can be applied to firm and industry analysis, as we demonstrate by measuring the degree of vertical integration in five software firms and deriving the overall degree of vertical integration in the software industry. We recommend the application of the unified software value chain as a consistent concept for further studies of software firms and the software industry.

The software industry is an important and fast growing industry with specific economic characteristics. However, as a result of a broad literature review it became apparent that existing software value chains and related concepts are not sufficient. They either blank out downstream activities or software specific activities are not included. In our concept the entire software creation process, from product idea to product replacement, is taken into account. In this article a unified software value chain was proposed by combining existing software value chains, the generic value chain, and software development processes. The resulting value chain comprises eleven activities: (1) product research, (2) component procurement, (3)

product development, (4) user documentation, (5) production and packaging, (6) marketing, (7) implementation, (8) training and certification, (9) maintenance and support, (10) operations, and (11) replacement.

As a first proof of concept five expert interviews with software companies were conducted. The interviewees confirmed the adequacy and completeness of the value chain. Furthermore, the interviewees confirmed the applicability by depicting their make-or-buy strategy for each activity based on the software value chain. Finally, we presented an exemplary application of the software value chain by measuring the degree of vertical integration of those five firms.

There are limitations to our research such as the small sample size and the challenge to define clear-cut requirements for the software value chain. Nonetheless, we think that it is important to disclose the requirements on which value chains are built and to provide a transparent process of conceptualization. The granularity of value activities is up for debate as well. Our future research will focus on providing more empirical evidence on validity and applicability of the proposed unified software value chain.

ACKNOWLEDGEMENTS

The project was funded by means of the German Federal Ministry of Education and Research under the promotional reference 01IC10S05M. The authors take the responsibility for the contents.

REFERENCES

1. Adelman, M. A. (1955) Concept and statistical measurement of vertical integration, *Business concentration and price policy*, 1, 281–322.
2. Anderson, J. C., Jain, D. C. and Chintagunta, P. K. (1993) Customer Value Assessment in Business Markets: A State-of-Practice Study, *Journal of Business-to-Business Marketing*, 1, 1, 3–29.
3. Barnes, S. J. (2002) The mobile commerce value chain: analysis and future developments, *International Journal of Information Management*, 22, 2, 91-108.
4. Carmel, E. (2003) The New Software Exporting Nations: Impacts on National Well Being Resulting From Their Software Exporting Industries, *The Electronic Journal of Information Systems in Developing Countries*, 13, 3, 1-6.
5. Crnkovic, I., Larsson, S. and Chaudron, M. (2005) Component-based Development Process and Component Lifecycle, *Journal of Computing and Information Technology*, 13, 4, 321-327.
6. Cusumano, M. A. (2004) The Business of software, Free Press, New York.
7. Dhama, H. (1995) Quantitative models of cohesion and coupling in software, *Journal of Systems and Software*, 29, 1, 65-74.
8. Engelhardt, S. (2008) The Economic Properties of Software, working paper 2008-045, Jena Economic Research Papers.
9. Feng, L. and Whalley, J. (2002) Deconstruction of the telecommunications industry: from value chains to value networks, working paper 2002/2, Strathclyde Business School.
10. Gartner, Inc. (2011) Gartner Highlights Five Long-Term, Overarching, and Interdependent Trends Affecting the Enterprise Software Industry, http://www.gartner.com/it/page.jsp?id=1535314, last access February 21 2011.
11. Haesen, R., Snoeck, M., Lemahieu, W. and Poelmans, S. (2008) On the Definition of Service Granularity and its Architectural Impact, *Lecture Notes in Computer Science*, 5074/2008, 375-389.
12. Jansen, S., Brinkkemper, S. and Finkelstein, A. (2007) Providing Transparency in the Business of Software: A Modeling Technique for Software Supply Networks, in L. Camarinha-Matos, H. Afsarmanesh, P. Novalis and C. Analide (Eds.) *IFIP International Federation for Information Processing*, Establishing the Foundation of Collaborative Networks, Springer, Boston, 677-686.
13. Katz, M. L. and Shapiro, C. (1994) Systems Competition and Network Effects, *The Journal of Economic Perspectives*, 8, 2, 93-115.
14. Léger, P.-M. and Yang, S. (2005) Network Effects and the Creation of Shareholders' Wealth in the Context of Software Firm Mergers and Acquisitions, in *Proceedings of the 13th European Conference on Information Systems, Information Systems in a Rapidly Changing Economy (ECIS 2005*) , May 26-28, Regensburg, Germany.
15. Léger, P.-M. and Quach, L. (2009) Post-merger performance in the software industry: The impact of characteristics of the software product portfolio, *Technovation*, 29, 10, 704-713.

16. Messerschmitt, D. G. and Szyperski, C. (2003) Software Ecosystem, The MIT Press, Cambridge.
17. OSHA: Occupational Safety and Health Administration (2011) 8243 Data Processing Schools, http://www.osha.gov/pls/imis/sic_manual.display?id=208&tab=description, last access February 27 2011.
18. Parolini, C. (1999) The Value Net, John Wiley & Sons, Chechester.
19. Pil, F. K. and Holweg, M. (2006) Evolving From Value Chain to Value Grid, *MIT Sloan Management Review*, 47, 4, 72-80.
20. Porter, M. E. (1985) Competitive Advantage, The Free Press, London.
21. Rothaermel, F. T., Hitt, M. A. and Jobe, L. A. (2006) Balancing Vertical Integration and Strategic Outsourcing: Effects on Product Portfolio, Product Success, and Firm Performance, *Strategic Management Journal*, 27, 1033-1056.
22. Royce, W. W. (1970) Managing the development of large software systems, *Proceedings of IEEE WESCON 26*, August 1970, Los Alamitos, USA, 1-9.
23. Shapiro, C. and Varian, H. R. (1999) Information Rules, Harvard Business School Press, Boston.
24. Spekman, R. E., Kamauff, J. W. and Salmond, D. J. (1994) At last purchasing is becoming strategic, *Long Range Planning*, 27, 2, 76-84.
25. Stanoevska-Slabeva, K., Talamanca, C. F., Thanos, G., and Zsigri, C. (2007) Development of a Generic Value Chain for the Grid Industry, *Lecture Notes in Computer Science*, 4685/2007, 44-57.
26. Williamson, O. E. (1991) Comparative Economic Organization, *Administrative Science Quarterly*, 36, 2, 269-296.
27. Xu, L. and Brinkkemper, S. (2007) Concepts of Product Software: Paving the Road for Urgently Needed Research, *European Journal of Information Systems*, 16, 5, 531-541.