5-15-2012

THE SOFTWARE VALUE CHAIN: METHODS FOR CONSTRUCTION AND THEIR APPLICATION

Anton Pussep
Technische Universität Darmstadt

Markus Schief
Technische Universität Darmstadt

Thomas Widjaja
Technische Universität Darmstadt

THE SOFTWARE VALUE CHAIN: METHODS FOR CONSTRUCTION AND THEIR APPLICATION

Pussep, Anton, Technische Universität Darmstadt, Chair of Information Systems, Hochschulstraße 1, 64289 Darmstadt, Germany, pussep@is.tu-darmstadt.de

Schief, Markus, Technische Universität Darmstadt, Chair of Information Systems, Hochschulstraße 1, 64289 Darmstadt, Germany, schief@is.tu-darmstadt.de

Widjaja, Thomas, Technische Universität Darmstadt, Chair of Information Systems, Hochschulstraße 1, 64289 Darmstadt, Germany, widjaja@is.tu-darmstadt.de

Abstract

The value chain is a widely used framework for industry and firm analysis. To our knowledge, the conceptualisation of value chains is so far guided by "soft" criteria like intuition of experts rather than clearly stated methods with regard to the value chain boundaries and the granularity as well as the separation of activities. Therefore, we propose a combination of well-known methods – such as the Delphi study approach and clustering algorithms – to (1) ensure a holistic view of the industry at hand by covering all underlying economic concepts, (2) ensure the uniqueness of activities, and (3) provide a hierarchy of activities that allows deriving value chains at different levels of granularity. Since software is a good with specific economic properties, practitioners and researchers require a value chain framework reflecting the industry specifics. This paper contributes by proposing methods for value chain construction and applying these methods to the software industry. The resulting universal and hierarchical software value chain can serve as a sound foundation for further studies of the software industry. Furthermore, practitioners can tailor the proposed methods to their needs and apply the software value chain to their firms.

Keywords: value chain, granularity, software industry.

1 Introduction

The value chain is a widely used framework introduced by Porter (1985) as a tool for developing and sustaining competitive advantage of a firm. Value chain analysis decomposes a firm into the activities it performs to create value. This allows a better understanding of cost behaviour and sources of differentiation (Porter, 1985).

The value chain concept is widely used in research and praxis (Stabell and Fjeldstad, 1998). Its applications range from performance measurement of a firm's positioning within an industry and vertical integration (Rothaermel et al., 2006) to structured industry analysis (Barnes, 2002) and cost analysis (McCormick, 2010). A firm's value chain is also considered a major component of business models (Morris et al., 2005). However, the generic value chain as proposed by Porter (1985) is not applicable to all industries (Stabell and Fjeldstad, 1998). Since software is driven by specific economic properties (Messerschmitt and Szyperski, 2003; Shapiro and Varian, 1999), analyses of the software industry require a value chain framework reflecting the industry specifics.

Methodical shortcomings hamper the construction of value chains. For instance, Porter (1985) suggests that "activities should be isolated and separated that (1) have different economics, (2) have a high potential impact on differentiation, or (3) represent a significant or growing proportion of cost". However, we are not aware of any applications of methods which ensure the separation of activities based on these criteria. Other unsolved issues involve the coverage of the entire value creation process at hand, as well as the appropriate granularity level of activities.

In response to the methodical shortcomings and the missing software specific value chain frameworks, this paper addresses the following research questions:

1. How to derive a holistic value chain for the software industry?
2. How to ensure the economic uniqueness of activities within the software value chain?
3. How to provide the appropriate level of activity granularity within the software value chain?

The main contribution to Information Systems research is the methodically sound deduction of a software value chain. The resulting value chain is on the one hand specific to the software industry, but on the other hand universal with regard to software firm specifics (e.g. firm size and standard / individual software providers). The software value chain can be used by practitioners and researchers to depict a firm's strategic choices with regard to vertical integration and positioning within an industry. Further, structured analyses of strategic choices and business models building upon our framework allow for performance measurement and comparisons of software firms. Finally, this paper contributes on a methodical level by proposing methods for value chain construction. These can be applied to other industries and enable similar analyses there as well.

The remainder of this paper is organized as follows: Section 2 provides an overview of the structure of this study. In section 3 we introduce a literature-based concept of the software value chain. Section 4 introduces attributes to describe activities in a structured way based on economic principles of the software industry. The collection of the data using expert judgements through the Delphi method is described in section 5. Section 6 provides a detailed description of our method to ensure the uniqueness of the value chain activities. In section 7 we describe the clustering algorithm that is used to build a hierarchy of activities. Finally, section 8 concludes the paper.

2 Structure of the study

For the development of the value chain methods, we follow some fundamental requirements. Firstly, we must build upon related work in the area of value chain concepts. Further, this perspective needs to be enhanced by software industry specific research that provides specific means and hence allows domain experts to analyse economic characteristics of software value activities. Finally, from a method point of view we shall refer to well-known and proofed methods. By compelling logic and reasoning, these methods need to be combined and arranged into a study setting that allows to be used as a method to distinct value chain activities. This section provides an overview of the methods applied in this study. Detailed methodical descriptions can be found in the respective chapters. The overall structure of this study can be decomposed in three phases as follows:

1. Definition of activities and their attributes: We refer to existing literature to identify a set of activities of the software industry. To analyse the economics of the value activities, we refer to economic principles of the software industry. The principles are transformed into economic attributes which can be used to evaluate the defined activities.
2. Data collection: We classify value chain activities based on the previously introduced economic attributes. For that, we refer to the Delphi study method as expert knowledge is required to evaluate economic characteristics of value chain activities.
3. Data analysis: We process the collected data in two main directions. Firstly, we analyse if the activities are unique according to their classification. By pairwise comparison, we identify a minimal set of attributes that allows a perfect distinction of the activities. The second analysis builds a value chain hierarchy with clustering algorithms. Based on the similarity of activity classifications we combine them to clusters. We derive various value chains that comprise ten to three activities. This approach ensures that activities within each cluster are more similar from an economic point of view than activities being assigned to different clusters.

3 Initial set of activities

Before activities can be classified and combined into a hierarchy, they need to be defined. Our understanding of activities and the value chain concept is built on the original definition by Porter (1985). A firm's value chain is a collection of activities, which can be separated in primary and secondary activities. Activities within the value chain 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).

Since our objective is to derive a framework applicable to the entire software industry, the entity of the investigation is a generic software firm independent of the product type (e.g. standard and individual software). Furthermore, we focus on primary activities as most other work in this area (Barnes, 2002; Li and Whalley, 2002; Messerschmitt and Szyperski, 2003; Stanoevska-Slabeva et al., 2007).

As mentioned previously, Porter (1985) suggests that activities should be separated if they are (1) economically distinct, (2) have a high impact on differentiation, or (3) represent a large or growing proportion of costs. Whereas the first criterion is rather governed by product specifics, criteria two and three are more dependent on firm specifics. In this paper, we focus on the first criterion, which can be derived theoretically from the economic principles of software.

Our initial set of software activities is derived from Pussep et al. (2011), who provide a generic software value chain based on a broad literature review and a first empirical proof-of-concept with practitioners from five software firms. They propose a software value chain comprising the 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. Following changes are made for reasons of granularity: component procurement is removed; user documentation is now part of the product development; maintenance and support are divided into separate activities. Activity names have been slightly adjusted and activity descriptions have been sharpened after the pre-test (see section 5). This results in ten activities with detailed descriptions shown in Table 1.

	Activity	Sub-activities	Detailed description
1	**Research**	Development of a product vision; fundamental research of algorithms; **decision upon major technologies and sub-systems; proof of concept**	This activity comprises fundamental product research. A product vision is developed and fundamental algorithms are researched. Major technologies and subsystems are selected. A first proof of concept is provided through a prototype or analysis of algorithms, technologies and subsystems. The result is a product idea, algorithm or proof of concept. Unlike in the following activities, no code is created here which becomes part of the actual product.
2	**Develop-ment**	**Requirements engineering; software design; coding;** subsystem testing; **subsystem integration; system testing;** user documentation; provisioning	This activity comprises the actual software development process. Based on requirements, a software design is created. The entire system is decomposed into subsystems. Subsystems are programmed and tested separately, before they are integrated and tested as a combined system. The user documentation is created and the product is compiled to an executable and versioned product. The result is an executable version of the product.
3	**Mainte-nance**	Same as in development	Same as development, but the focus is on bug fixing and enhancing an existing product, whereas the activity development aims at the creation of a new product. Within maintenance, disruptive changes are not allowed. Instead, incremental changes are made by the producer to an existing product in the marketplace.
4	**Production**	Assembly; **printing;** packaging	Within assembly, software and respective documentation are bundled to one package. The assembled software package is printed to a physical medium and the documentation is printed on paper. In packaging the physical product artefacts are packaged in a physical package. The result is a product with all attributed artefacts, which is ready for shipment.
5	**Marketing**	Launch; price; **place; promotion;** bundling; brand management	Providing a means by which buyers can purchase the product and inducing them to do so, such as sales and promotion. The result is the readily marketed product in the marketplace, such that potential customers are aware of the product and the product is available for purchase.
6	**Replace-ment**	Alternatives; **migration**; shut-down	First, replacement deals with the decision if the product (once it becomes outdated and reaches the end of its lifecycle) 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.
7	**Implemen-tation**	Installation; **configuration;** adjustment; **business process reengineering**	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.
8	**Education**	**Training;** certification	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.
9	**Support**	**Primary support; development support**	Support can be differentiated in primary and development support. While the first sub-activity deals with the support of users, the second activity relies on deep technical knowledge and implies code reviews.
10	**Operations**	**Hosting;** monitoring; backup; **upgrade**	The operations activity ensures the execution and management of a product on an information system during actual usage by customers. By monitoring, the system behaviour can be analysed 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.

Table 1. Descriptions of activities in the software value chain. Sub-activities defined as being more important than others are written in bold.

4 Derivation of activity attributes

In general, before activities can be classified, we have to define attributes and their value ranges. Activities can be described by a vector, where size equals the number of attributes and vector elements correspond to attribute values of the respective activity. Using this representation of activities, quantitative analysis can be performed.

The definition of attributes is crucial in order to obtain meaningful results. We identified following requirements on attributes:

1. Relevance: Attributes must be relevant for the industry at hand in order to be able to describe it. Furthermore, attributes should be relevant for all types of created product (e. g. in case of software individual and custom software).
2. Completeness: Attributes should capture all facets of the activities with regard to some concept. The concept might be a theory providing a set of properties relevant to the industry at hand. If those properties are chosen as attributes, then all of them should be used, such that the selection of attributes can be claimed to be complete with regard to the underlying theory.
3. Determinability: Attributes should have a clearly defined range of values and the attributes should be determinable by an expert.

Software is inherently different than other products and is governed by own economic principles. Engelhardt (2008), Messerschmitt and Szyperski (2003), and Stelzer (2004) provide comprehensive overviews. Table 2 summarizes all economic principles identified from these sources. A software value chain should reflect industry specifics. This can be achieved by evaluating activities with regard to economic principles of the software industry. However, it was not possible to use the economic principles as attributes directly. This was due to the large number of principles. Furthermore, it was often not possible to evaluate how an activity relates to a single principle. A major reason is the undefined range of values per principle.

Since the economic principles of the software industry turned out to be insufficient activity attributes, we introduced an interlayer of attributes as shown in Table 3. The value range of each attribute is binary, such that each activity can be represented by a binary vector. Each attribute can be mapped to a number of economic principles, thus representing at least one economic principle. The only exception is attribute I which was introduced in order to reflect chronological order within the value chain (Porter, 1985). Table 2 shows the mapping from economic principles to attributes. For example, attribute C (asking whether an activity is performed once or multiple times per customer) is mapped to principle 7 (ease of replication), because an activity which is performed multiple times per customer is likely easier to replicate than an activity which needs to be adjusted on every execution.

The introduced interlayer of attributes fulfils the previously stated requirements, because: (1) the attributes reflect the economic principles of the software industry and are therefore relevant; (2) the attributes cover all underlying economic principles and are therefore complete; (3) all value ranges are binary, this improves the determinability of attributes because in many cases it is easier to evaluate based on two extremes rather than on a scale of gradually different values.

Economic principle		Attribute										
		A	B	C	D	E	F	G	H	I	J	K
1	System dependency										x	
2	Intangibility	x										
3	Integration of external factor						x					
4	Increasing computing power (Moore's Law)										x	
5	Secondary role of performance										x	
6	Cheap storage of increasing data										x	
7	Ease of replication			x				x				
8	Portability by information systems	x										
9	Development with information systems				x	x						
10	Ease of modification				x	x						
11	High complexity				x	x						
12	High need for good product and system architecture				x	x						
13	High fix costs								x			
14	High requirements for technology und innovation management				x	x						
15	Customer-oriented design of goods and services						x					
16	Customer involvement during product development						x					
17	Iterative development						x					
18	Support of users during information processing										x	
19	Tradeoff between availability and capactiy utilization		x									
20	High economies of scope			x				x				
21	Opportunities for differentiation		x									
22	High importance of broad user basis											x
23	Software as an experience good											x
24	Utility-dependent value											x
25	New pricing models		x									
26	Special requirements for security and authenticity										x	
27	High change barriers for customers											x
28	Possibility of standardization of software											x

Table 2. Mapping from economic principles of the software industry to attributes of software value chain activities. Attribute descriptions can be found in Table 3. All economic principles have been identified from Engelhardt (2008), Messerschmitt and Szyperski (2003), and Stelzer (2004).

	Attribute	Value 0	Value 1
A	The activity result is rather:	Tangible / "can be touched"	Intangible / immaterial
B	The decisions involved during activity performance are rather:	Strategic	Technical / operational
C	Per customer (thus, for one product instance) the activity is rather performed:	Multiple times	One time
D	Does the activity execution require knowledge of the product source code?	Yes	No
E	Does the activity execution require deep IT understanding?	Yes	No
F	How closely are the end-users involved in the activity execution?	Intensively	Loosely
G	Can the producer perform the activity once and reuse the result multiple times (for multiple customers)?	Yes	No
H	On first activity execution, which costs prevail?	Personnel	Non-personnel (e.g. hardware)
I	In relation to the point of productive usage on customer side (go live), when is the activity performed?	Before	After
J	Is the result rather a change in:	Human knowledge	Information systems
K	To what extent can the activity result contribute to a software's compatibility?	High	Low

Table 3. Activity attributes and values.

5 Classification of activities

Given the attributes to describe activities, activities can be classified by assigning values to attributes. In general, this can be done by a single researcher who is familiar with the economic theories and the software industry. However, we decided to go for a Delphi study with twelve participants in order to build upon a broad understanding. Furthermore, we abstract from the authors' views by excluding them from the set of participants.

The Delphi study, being an iterative feedback technique among an expert panel, was developed at Rand Corporation in the 1950s (Dalkey and Helmer, 1963; Landeta, 2006). The main objective usually is to obtain a reliable consensus among a group of experts on a complex issue (Okoli and Pawlowski, 2004). Furthermore, the Delphi study allows follow-up interviews leading to a deeper understanding. Finally, it comprises a virtual panel of experts that can be contacted asynchronously, thus allowing including experts from different locations. The structure of our Delphi study is derived from Okoli and Pawlowski (2004) and contains the following phases:

1. Questionnaire design: The initial questionnaire is derived from activity attributes in section 4. For each activity/attribute combination each participant is asked to choose between the two possible values and leave a comment justifying the judgement. The questionnaire includes a detailed description of all activities and their sub-activities as shown in Table 1.

2. Pre-test: We conducted a pre-test with one participant to assure that activity descriptions and attribute values are understood correctly. This participant has similar level of domain knowledge to other experts. He was excluded from further phases and his judgments are not included in the final results.

3. Participant selection: In general, the number of participants involved should be in the range from 10-18. We selected twelve experts having a similar background and level of knowledge.

4. Delphi round: The survey is rolled out to all participants in the format of a questionnaire and is returned within a given timeframe.

5. Result analysis: Participants' answers and comments are analysed by moderators. For each activity/attribute combination a satisfactory level of agreement is reached if at least 80% of all participants gave the same judgement. All comments are evaluated in order to identify misunderstanding of activities or attributes.
6. Reiteration: Steps 4 and 5 are reiterated until the judgements reach a satisfactory degree of consensus and misunderstandings are resolved.

The resulting consensus values are shown in Table 4. It contains values where a minimum 80% level of agreement could be reached after last round. All elements below this threshold are empty, indicating not available (NA) values. Since further analyses work on available values only, they are based on element values where the high 80% level of consensus among experts could be reached.

	Activity	**A**	**B**	**C**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
1	R&D	* 1	* 0	* 1			* 0	* 0	* 0	* 0	* 0
2	Development	1	* 1		* 0		* 0	* 0	* 0	* 1	* 0
3	Maintenance	* 1	* 1	* 0	* 0		* 0	* 0	* 1	* 1	
4	Production	* 0	* 1	* 1	* 1	* 1		* 1	* 0	* 1	* 1
5	Marketing		* 0	* 0	* 1		* 0	* 0	* 0	* 0	* 1
6	Replacement	* 1	* 1	* 1	* 0		* 1	* 0	1	* 1	
7	Implementation	1	* 1	* 1	* 0	0	* 1	* 0	* 0	* 1	
8	Education	* 1	* 1	* 0	* 1	* 0		* 0	* 0	* 0	* 1
9	Support	* 1	* 1	* 0		* 0		* 0	* 1	* 0	* 1
10	Operation	* 1	* 1	* 0	* 0	* 1		* 1	* 1	* 1	1

Table 4. The consensus matrix after the final Delphi round. Empty cells contain NA values and indicate elements where the consensus level among experts is below 80%. Elements marked with an asterisk reach a consensus level of more than 90%. The values 0 and 1 correspond to attribute value ranges defined in Table 3.

The objective of our Delphi study is to find as many activity/attribute combinations with a satisfactory level of agreement as possible. In total, three Delphi rounds are carried out. During this process, consensus can be reached on 84% of all combinations. For others, participants' comments lead to the conclusion that no consensus can be reached. In most cases, this is due to the dependence of the judgement on product type or other assumptions which cannot be pre-defined because of the requirement on the generic nature of the software value chain. For instance, for activity 5 (marketing) and attribute A it is not possible to decide if the result is tangible or intangible. Whereas product placement can be tangible in case of a physically touchable product in a shop, it can be intangible as well if the product is marketed through the Internet (e.g. Software as a Service). Table 5 summarizes not decidable combinations and provides further descriptive details for each round.

In the second and third Delphi round, we use Kendall's coefficient of concordance (W) to provide participants with a qualitative assessment of consensus ranging from "very weak" to "very strong". This statistical method is often used within Delphi studies, particularly in the area of ranking-type Delphi studies (Schmidt, 1997). Further important changes include the deletion of attribute D as it was too redundant with attribute E. Finally, attribute K was changed after first round, because participants' comments indicated too broad scope of network effects.

	Initial	Round 1	Round 2	Round 3
Attributes	11	10	10	10
Elements on round start	110	110	100	100
Elements deemed as not decidable	None	All D and: 5A, 2C, 1E, 9E, 1F, 2F, 3F, 8G, 9G, 10G	16 (in addition to round 2: 5F, 6F, 4G, 3K, 6K, 7K)	Equal to round 2
Remaining decidable elements		90	84	84
Participants		12	12	4
Questions asked		110*12=1,320	376	6
Kendall's W		0.5441	0.7646	0.7741
# of all NA values in consolidated matrix		35	18	16
# of decidable NA values in matrix		25	2	0
Updated judgements		100	152	4

Table 5. Descriptive information after each Delphi round. Elements are identified through shortcuts such as "5A", where 5 denotes the activity marketing and A the respective attribute.

6 Uniqueness of activities

In order to prove the uniqueness of the activities each of them must be shown to have own specifics when compared to all other activities. We define an activity as unique if there is no other activity which has exactly the same values in all attributes. Attributes, where at least one of the two activities has an NA value are not compared. This treatment of NA values is reasonable because an NA value is not necessary different from other values and is not necessarily equal to another NA value (Witten and Frank, 2005). This treatment of NA values makes the appearance of duplicates more likely when the share of NA values is high. It can be regulated by adjusting the threshold which is used to transform average votes to values in the consensus matrix. A low threshold results in fewer duplicates, but undermines the confidence in the values due to the low consensus levels. As a consequence, a high threshold should be chosen in order to ensure a sufficiently reliable proof of uniqueness within the value chain.

Based on this definition of uniqueness, each activity illustrated in the consensus matrix in Table 4 is unique. Furthermore, it appears that far less attributes are necessary in order to ensure the economic uniqueness of activities. Trying out all possible combinations of attributes we derived two minimal combinations: (B, G, H, I, J, K) and (B, C, G, H, I, J). Thus, there is no set of less than six activities which would lead to a unique software value chain. Also, attributes B, G, H, I and J appear to be more important than others, since they are contained in both sets, whereas attributes C and K are substitutes to each other.

7 Hierarchy of activities

The appropriate level of granularity in a value chain may vary depending on current needs. For strategic questions a high-level view of few activities might be more suitable, whereas looking at processes requires a more detailed view. Starting with ten activities as defined in section 3, more coarse-grained levels of granularity can be provided by combining existing activities. Within this process two questions arise:

1. Which activities should be combined?
2. How to evaluate and choose between different levels of granularity?

It appears reasonable that the second questions cannot be answered for all cases, but some criteria can be provided to indicate which levels should not be used. With regard to the first question, the most similar activities should be combined, whereas the similarity between remaining activities should be low.

The combination of similar objects is called clustering. A generic bottom-up approach that starts with a single object per cluster and successively combines those to clusters is hierarchical agglomerative clustering (HAC) (Chakrabarti, 2003). We added two modifications to the generic HAC algorithm:

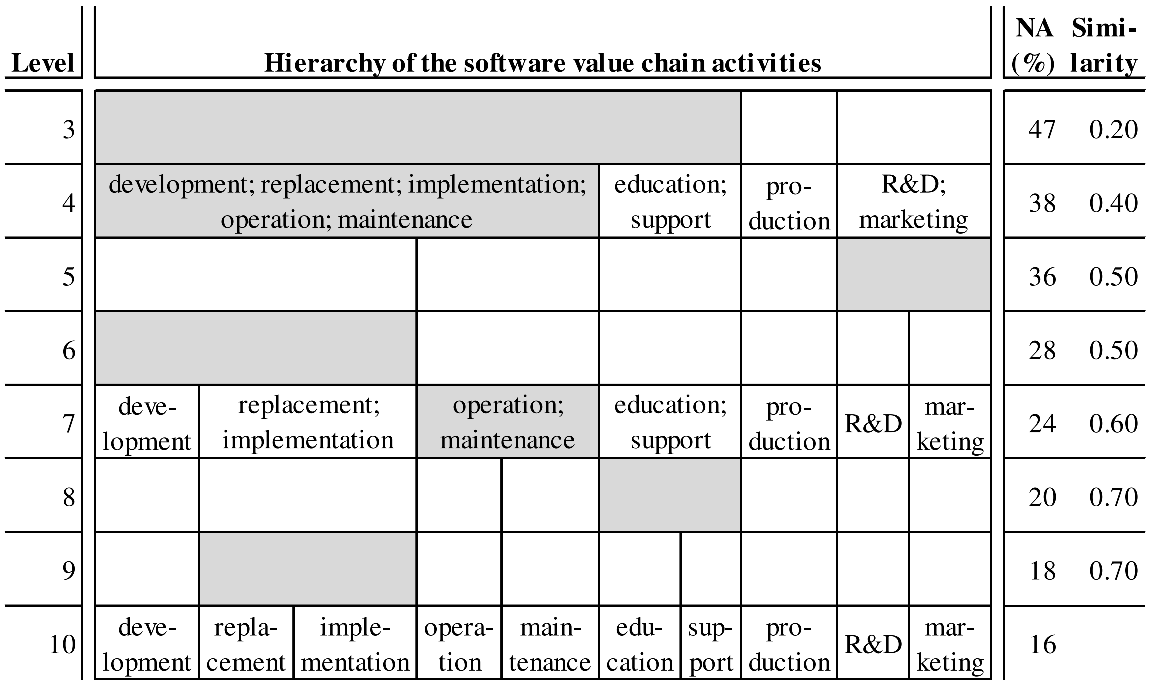
1. It can deal with NA values.
2. Selected activities are actually merged to a new activity, resulting in a new set of activities.

The similarity measure between two activities is calculated as the number of all equal values, divided by all attributes including NA values (Finch, 2005). Due to the low value range of the similarity measure, we introduced an additional goodness criterion which is used when there are multiple candidates for combination. For instance, this is the case on level 9 of the resulting hierarchy as shown in Figure 1. There the similarity value is 0.7 for the activities pairs (replacement, implementation) and (education, support). Therefore, the goodness criterion is used. It is defined as the average of absolute differences between all attribute values, whereas the attribute values are not 0 or 1, but the average consensus values obtained from the Delphi study. Thus, the goodness criterion differs from the similarity value in the usage of all attributes, as well as the attribute values.

The resulting hierarchy and describing statistics are shown in Figure 1. Starting from the bottom at level 10 with 10 activities, in each round two activities are successively combined. As a result, it can be seen that logically similar activities are combined. For instance, the implementation of a new system replaces an old system, such that we can assume similarities between these activities. Education and support are both targeted at increasing user's capabilities with the software, thus there is compelling logic in the combination of those activities as well. The combination of operation and maintenance is reasonable, since software companies providing operation will usually take care of maintenance as well. The combination of R&D and marketing is likely to be caused by the fact that both activities differ from other activities by nature. The combination seems hard to justify from a logical point of view.

With regard to the appropriate number of activities, multiple criteria can be used. Their nature can be quantitative and qualitative. We look at four criteria: (1) similarity value of combined activities, (2) relative number of NA values in resulting consensus matrix, (3) uniqueness of activities, and (4) logical reasoning between combined activities. When a criterion significantly worsens between two levels, it is an indicator that a reasonable level of granularity has been reached before the combination between those two levels takes place. This "stepsize" rule was proposed by Sokal and Sneath (1963). A discussion of different rules can be found in Milligan and Cooper (1985).

The similarity measure decreases from level 4 to 3 by 0.2, whereas it only decreases by 0.1 in all previous steps. Therefore, it suggests that level 4 is preferable. The second criterion suggests level 6 or 4. However, level 5 and 6 contain duplicate activities, such that the third criterion discourages their usage. Finally, applying the fourth criterion it appears that level 4 is unfortunate due to the combination of R&D and marketing. Therefore, we would recommend the usage of level 7, resulting in a value chain of the activities (1) development, (2) replacement & implementation, (3) maintenance & operation, (4) education & support, (5) production, (6) marketing, (7) R&D. For more coarse-grained granularity level 4 should be used.



Level	Hierarchy of the software value chain activities										NA (%)	Similarity
3											47	0.20
4	development; replacement; implementation; operation; maintenance					education; support		pro-duction	R&D; marketing		38	0.40
5											36	0.50
6											28	0.50
7	deve-lopment	replacement; implementation		operation; maintenance		education; support		pro-duction	R&D	mar-keting	24	0.60
8											20	0.70
9											18	0.70
10	deve-lopment	repla-cement	imple-mentation	opera-tion	main-tenance	edu-cation	sup-port	pro-duction	R&D	mar-keting	16	

Figure 1. *Resulting hierarchy of activities after each round. Starting with ten activities (at the bottom), in each round two activities are successively merged to a new activity. The resulting combined activity is highlighted in grey. Activity labels are not shown for non-desirable hierarchy levels.*

8 Conclusion and avenues for further research

In this paper, we introduce methods to derive value chains which (1) ensure a holistic view of the industry at hand by covering an underlying theoretical concept, (2) ensure the uniqueness of activities by evaluating activity specifics, and (3) provide an appropriate level of granularity within the value chain by building a hierarchy of activities. By applying these methods to the software industry we deduct a software value chain of ten economically unique activities. Further, we present a hierarchy of those activities and provide recommendations on the usage of appropriate levels of aggregation in applications.

Our contribution to research is the combination of well-known methods such as clustering algorithms and the Delphi study to value chain construction. The methodical approach is generic and can be applied to other industries as well. The software value chain can be used by practitioners and researchers to depict a firm's strategic choices with regard to vertical integration and positioning within an industry. Further, structured analyses of strategic choices and business models building upon our framework allow for performance measurement and comparisons of software firms. The software value chain is independent of the particular software type or firm size, but can also be tailored to specific needs, for instance by incorporating firm-specific attributes.

As a limitation to the proposed methods, the selection of appropriate attributes is crucial. Those must be understood by all participants and cover the underlying concept well. The applicability of the value chain construction methods should be evaluated further by applying them to other industries and comparing the resulting value chains. Our further research will focus on a larger survey for activity classification and software value chain validation.

Acknowledgments

This work was supported by the Software-Cluster and partially funded within the Leading-Edge Cluster competition by the German Federal Ministry of Education and Research (BMBF) under grant “01|C10S05”. The authors take the responsibility for the contents.

References

Barnes, S.J. (2002) The mobile commerce value chain: analysis and future developments, International Journal of Information Management, 22 (2), 91-108.

Chakrabarti, S. (2003) Mining the Web: Discovering Knowledge from Hypertext Data, Morgan Kaufmann Publishers, San Francisco.

Dalkey, N. and Helmer, O. (1963) An Experimental Application of the Delphi Method to the Use of Experts, Management Science, 9 (3), 458-467.

Engelhardt, S.V. (2008) The Economic Properties of Software (Working Paper), Jena Economic Research Papers 2008-045, Friedrich-Schiller-University Jena, Max-Planck-Institute of Economics.

Finch, H. (2005) Comparison of Distance Measures in Cluster Analysis with Dichotomous Data, Journal of Data Science, 3, 85-100.

Landeta, J. (2006) Current validity of the Delphi method in social sciences, Technological Forecasting and Social Change 73, 73 (5), 467-482

Li, F. and Whalley, J. (2002) Deconstruction of the telecommunications industry: from value chains to value networks, Telecommunications Policy, 26 (9-10), 451–472.

McCormick, T. (2010) Understanding Casts Using the Value Chain A Ryanair Example, Accountancy Ireland, 42 (5), 28-30.

Messerschmitt, D.G. and Szyperski, C. (2003) Software Ecosystem, The MIT Press, Cambridge.

Milligan, G.W. and Cooper, M.C. (1985) An Examination of Procedures for Determining the Number of Clusters in a Data Set, Psychometrika, 50 (2), 159–179.

Morris, M., Schindehutte, M. and Allen, J. (2005) The entrepreneur's business model: toward a unified perspective, Journal of Business Research, 58 (6), 726-735.

Okoli, C. and Pawlowski, S.D. (2004) The Delphi method as a research tool: an example, design considerations and applications, Information & Management, 42 (1), 15-29.

Porter, M.E. (1985) Competitive Advantage, Free Press, London.

Pussep, A., Schief, M., Widjaja, T., Buxmann, P. and Wolf, C.M. (2011) The Software Value Chain as an Analytical Framework for the Software Industry and Its Exemplary Application for Vertical Integration Measurement, AMCIS 2011 Proceedings pp. 1-8, Detroit.

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.

Schmidt, R.C. (1997) Managing Delphi Surveys Using Nonparametric Statistical Techniques, Decision Sciences, 28 (3), 763-774.

Shapiro, C. and Varian, H.R. (1999) Information Rules, Harvard Business School Press, Boston.

Sokal, R.R. and Sneath, P.H. (1963) Principles of Numerical Taxonomy, Freeman, San Francisco.

Stabell, C.B. and Fjeldstad, Ø.D. (1998) Configuring value for competitive advantage: on chains, shops, and networks, Strategic Management Journal, 19 (5), 413-437.

Stanoevska-Slabeva, K., Talamanca, C.F., Thanos, G. and Zsigri, C. (2007) Development of a Generic Value Chain for the Grid Industry, In (Eds, Veit, D. and Altmann, J.): Grid Economics and Business Models, Lecture Notes in Computer Science, Springer, Berlin, 44-57.

Stelzer, D. (2004) Produktion digitaler Güter, In (Eds, Braßler, A. and Corsten, H.): Entwicklungen im Produktionsprozessmanagement, Vahlen, München, 233-250.

Witten, I.H. and Frank, E. (2005) Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers, San Francisco.