Journal of Software: Evolution and Process

JOURNAL OF SOFTWARE: EVOLUTION AND PROCESS
J. Softw. Evol. and Proc. 2013; **25**:711–741
Published online 27 June 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/smr.1560

The software value map — an exhaustive collection of value aspects for the development of software intensive products

Mahvish Khurum[1,*,†], Tony Gorschek[1] and Magnus Wilson[2]

[1]*Blekinge Institute of Technology, SE-371 79, Karlskrona, Sweden*
[2]*Ericsson AB Box 518, SE-371 23, Karlskrona, Sweden*

SUMMARY

In software intensive products such as cars or telecom systems, software has traditionally been associated with cost, and there has been no real perception of its value in relation to the entire product offering. However, because software is becoming a larger part of the main competitive advantage, driving innovation and product differentiation, hardware is becoming more standardized, thus the valuation of software is becoming critical. In existing literature, several value constructs and corresponding valuation/measurement solutions needed for making decisions about software product development are presented. However, the contributions are often isolated with respect to a certain perspective such as focusing on product's internal or external quality aspects only. Consequently, a complete view of value constructs relevant from different perspectives required for making decisions about software product development is missing. This paper presents a consolidated view of the software value concept utilizing the major perspectives and introduces a software value map. The created value map was evaluated through an industry case study through the development of impact evaluation patterns, which were subsequently used by professionals in industry, and experiences gathered. During industry evaluation, practitioners found substantial benefits of having a consolidated, vastly improved, and extended value aspect's view of software. Copyright © 2012 John Wiley & Sons, Ltd.

Received 9 August 2011; Revised 28 March 2012; Accepted 16 May 2012

KEY WORDS: value-based software engineering; decision-support; software value, software value analysis; software value map; requirements engineering; technology and software product management; product customization; software engineering management; value taxonomy

1. INTRODUCTION

It is very hard to estimate and calculate software value. This is especially true for embedded or hardware intensive products. In many cases, the traditional view has been that the software part of a product is the 'poor cousin' that 'had to be there', bundled with the hardware, but without any real value in itself [1, 2]. Examples of this can be seen in many embedded fields like the automotive or the automation domain. Thus, at best software was handled as a cost and there was no real perception of the software value beyond immediate sales of the product [3, 4].

In today's competitive world, however, software has become the main competitive advantage, enabling faster and cheaper innovation and product differentiation, and at the same time hardware is becoming standardized [5]. Simultaneously, the size and complexity of software in products are increasing, and so is the impact of software development decisions on the overall product offering. That is, any decision taken regarding software (e.g., what features to realize, what quality to offer, or what technology to choose) will impact the entire product's life cycle and value, not to mention that it limits future possibilities and direction of the product and business [2, 6, 7]. This situation

*Correspondence to: Mahvish Khurum, Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden.
†E-mail: Mahvish.Khurum@bth.se

gives rise to many decision-making challenges for industry practitioners, for example, what is the actual value of software? How does the realization of one feature or quality aspect influence the overall value of the product offering, where short-term potential sales and revenues are only small parts of the complete picture? Value-based software engineering (VBSE) emphasizes that every decision and/or feature of a product does not have an equal value like in a value-neutral setting [2]. This requires making decisions that are better for overall value creation [8, 9], and balancing short-term and long-term value creation. To enable this, different perspectives and corresponding value constructs need to be considered together while making decisions for sustaining growth and maintaining the competitive advantage.

The research presented in this paper offers four main contributions. First, a consolidated view (called the software value map, SVM) detailing software value as a concept is presented utilizing four major perspectives: (i) financial, (ii) customer, (iii) internal business process, and (iv) innovation and learning. SVM offers a unified view of value where value constructs are categorized as value aspects, subaspects, and value components, which can be used by professionals to develop a common understanding of value, and acting as decision support to assure no value perspective is unintentionally overlooked when taking product management and development decisions. The second contribution is the mapping of links between different value constructs — making interrelationships explicit. As a part of this categorization, we also collected measures, measurement methods, techniques, and models (called measurement solution from here on in) used to measure a specific value component. As a third contribution, the gaps are mapped. That is, missing value constructs, and connections between different value constructs are made explicit to lift the importance of future research to fill the gaps. Fourth, but not least, SVM was used to construct a number of impact evaluation patterns at Ericsson, in essence a tailoring SVM to an industry case achieving an initial industrial evaluation.

The research presented here is a result of collaborative research conducted with Ericsson AB Karlskrona. Ericsson is involved in the development of embedded software and, realizing the importance of value-based software product development, is currently evolving its advanced conceptualization of software as a central value component in its embedded products.

The paper is structured as follows. Section 2 gives an overview of value from different perspectives relevant for software product development organization based on software engineering, business, management, and economics literature. Section 3 details the research methodology used to identify the need for the software value map, its creation, and evaluation in industry along with associated validity threats. The results of the literature review are presented in Section 4. Taxonomy, structure, and definitions of the value map are presented in Section 5. In Section 6, the industry evaluation of the software value map is presented together with impact evaluation patterns created for the selected strategic decision-making activity at Ericsson. Lessons learned during this case study are presented in Section 6.4, and Section 7 concludes the paper.

2. BACKGROUND AND RELATED WORK

Research focusing on VBSE recommends that the economic value perspectives be integrated into the software engineering processes to extend the traditional scope of software engineering from the technical issues to business relevant decision challenges [2]. Section 2.1 expands on the absence of a consolidated view of value and elaborates on associated repercussions. Section 2.2 provides a summary of the measurement and valuation solutions presented in literature for measuring different value constructs.

2.1. The absence of a consolidated view on value

Several researchers have presented value constructs and corresponding valuation/measurement solutions needed for making decisions about software product development [9–14]. However, the contributions are often isolated and have a limited perspective, focusing on, for example, only cost, or only product characteristics like usability. Consequently, a complete picture of value constructs,

DOI: 10.1002/smr

relevant from different perspectives required for taking software product management and development decisions, is missing. For example, from the product perspective, if software value is measured by measuring external and internal quality attributes [15], say usability or reliability, it is only one isolated measure. The usability value is often shown as a number (for example, understandability = A / B, where A = number of functions (or types of functions) understood, and B = total number of functions (or types of functions)). The question remains however, how does this number relate to the software product value relevant from different perspectives, like the overall customer perceived value, or the impact on internal business processes?

Similarly, cost–benefit analysis methods like the Cost Benefit Analysis Method (CBAM) and LiVASAE [16, 17], based on subjective evaluation of architectural strategies with respect to quality attributes of the architecture, fall short on connecting cost–benefit discussions to a broader context (e.g., customer perceived value, innovation value, and/or the impact on internal business processes).

Business research investigates value constructs from a number of other perspectives, that is, production value, differentiation value, intellectual capital value and shareholders' value [10, 18]. However; in software engineering literature, these have little or no explicit connection to the software product planning (requirements triage, requirements selection, and product management), and/or development (design, coding and testing) activities [2, 19].

Furthermore, the value of software is influenced by various 'nontechnical' factors (for example, business and marketing) [20], and realized only after the product is marketed [14]. Moreover, many businesses, marketing and technical factors are interrelated [21]. For example, the perceived value of software is dependent on many factors like the product characteristics and marketing efforts. The product characteristics in turn, are determined by the development process characteristics like elicitation quality, verification, and validation standards, but also by the system properties, like usability, performance, and reliability [19, 21]. When considering business, marketing, and technical factors independently, there is also a risk of overlap or redundancy, which can affect the accuracy of valuation.

On the basis of an extensive literature study and an industry case study (see Section 3), a list of challenges was compiled. These challenges summarized in Table I stand as motivation for the creation of the SVM.

Each of these challenges is addressed directly through the creation and use of the software value map as can be seen in the remainder of the paper, and explicitly discussed in Section 6.4.

3. RESEARCH METHODOLOGY

The general research approach used for the creation of the software value map and its validation in industry is shown in Figure 1. The need for the creation of software value map was identified through exploratory case studies at Ericsson (Step 1).

On the basis of the problem statement, an extensive literature review was undertaken to identify state-of-the-art (Step 2, see Section 3.1 for details). Through this literature review, the software value map was created as a candidate solution for the problems identified (Step 3, see Section 5 for details). Because the software value map was created based on needs identified in industry, a natural progression was an evaluation in industry. Through collaboration with Ericsson AB, static evaluation of the value map concept was performed through creation of impact evaluation patterns and use of the created patterns for decision support (Step 4, see Section 6).

3.1. Step 2: Literature review methodology

Extensive literature studies were used to identify all the value constructs relevant for making decisions in the development of software intensive products. Ideally a systematic mapping of all the relevant literature in economics, management, business, and software engineering should have been performed. However, because it was not practical to cover all literature from so many subject areas, as a first step a systematic mapping was performed to collect all value-based aspects from software engineering literature. Then as a next step, snow-ball sampling (for details see Section 3.1.2) was

DOI: 10.1002/smr

Table I. Challenges in decision-making.

C1: Multiple perspectives/views and lack of common vocabulary

It was found during exploratory studies at Ericsson that practitioners in the product management and development have different definitions and understanding of value constructs and they do not consider all the value constructs while making decisions. Decision-making, on the other hand, often involves deliberations in different perspectives. Distinct perspectives or views support knowledge acquisition and representation suitable for different types or stages of inference in the same discourse. Involvement of distinct perspectives does offer an opportunity to look at a decision from different aspects; however, this gives rise to misinterpretations and misunderstandings about value constructs to be considered.

C2: Long-term versus short-term gains and losses

In a market-driven environment the development organization takes all the risk, because development is not contractually bound; rather customers are whole markets and there is a large number of potential customers [28]. In addition, the requirements coming in are from a wide variety of sources, both external such as market surveys and key customers, and internal sources like developers, sales, marketing, support, competitor analysis, and management [29]. This presents several challenges to the product management organization, which has to be handled as a part of the market-driven requirements engineering process:

1. First, large quantities of requirements, sometimes numbering in the thousands or even tens of thousands, risk to overload the development organization [30]; thus initial triage of requirements is necessary [31, 32].
2. Second, the analysis and trade-off between requirements dictates long-term versus short-term product development, and the ability to balance functional requirements with nonfunctional aspects such as architectural longevity and maintainability.
3. Third, once analyzed and weighed, the ultimate selection of what requirements to realize, and which to postpone and dismiss, are central to both short-term and long-term success of a product [33].

In this environment, long-term costs and benefits in terms of sustainability and innovation are often ignored or overlooked. Practitioners decide what to include in a product by looking only at the short-term costs, short-term customer satisfaction, and short-term sales. However, sustainable software (sustainability of the software: a term used interchangeably with software maintenance [34]), and an innovation infrastructure are fundamental inputs to continuously maintain and evolve software products such that they remain competitive throughout their planned lifecycle.

C3: Release planning quality

Commonly, software-intensive product developing organizations provide successive releases of the product and release planning is a critical activity [35]. A major challenge in market-driven requirements engineering is to prioritize and select the right set of requirements to be implemented in the next release [36], while avoiding obstruction in the selection process [37]. This decision-making is very challenging because it is based on uncertain predictions of the future, while crucial for the product's success on the market [35].

C4: Value-based impact and consequence analysis

Money is the least common denominator that can be compared across all enterprise functions. This is because it is much easier to estimate the costs and revenues in monetary terms. However, financial/economic view of a software product life cycle is missing today. This limits the possibility to compare the actual value of adding and maintaining a feature in software for detailed analysis with respect to total cost of ownership, packaging, and procurement. Additionally, impact and consequences on sustainability and innovation potential are ignored.

used to cover all the areas other than software engineering to identify value constructs that could be relevant. The design of each study is described below.

3.1.1. Step 2.1: Systematic mapping. The systematic mapping methodology outlined in [38, 39] was used. The aim was to find all the theoretical value constructs, which are or should be considered while making decisions related to the development and management of software intensive products.

3.1.1.1. Search strategy and screening. Search strings were formulated through the following steps.

- All possible value keywords were identified by scanning research papers' title, abstract, and index terms (research papers referenced in Section 2 were used as a starting point)
- Main terms were identified by determining the population and the intervention
- Synonyms of the keywords were identified by using a thesaurus
- Last, search strings were formulated using Boolean operators such as AND, OR, PRE, ONEAR, etc.

In the search strings all the phases of software development were included to identify all value constructs relevant and/or used in different phases of software development. Exact keywords used and the search strings are available online [40]. Table A.I in Appendix A gives an overview of the

DOI: 10.1002/smr

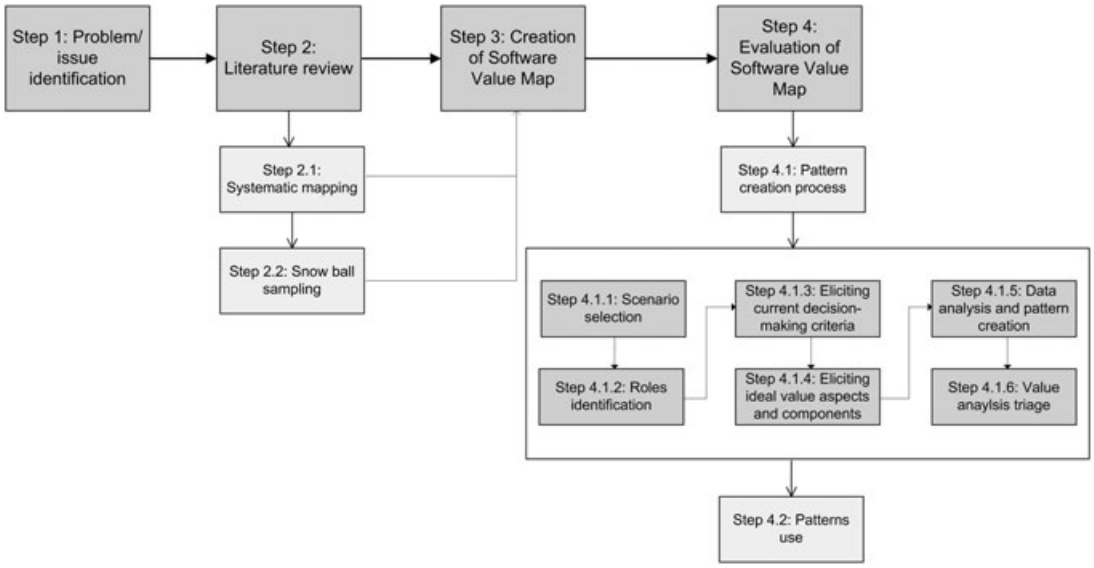

Figure 1. Overview of research approach.

data items relevant for the systematic mapping. Appendix A also contains details about the inclusion/exclusion of the research papers at different stages of selection.

3.1.1.2. Establishing a classification scheme. Because the purpose of the systematic mapping was to identify which value constructs were considered while taking decisions, the value keywords [40] from the papers were used to classify the papers. Papers stating similar value constructs with different names, for example 'product value' and 'exchange value' were classified under one value construct (in this case 'product value' was used in the classification). The definitions/descriptions stated by the authors in their papers were used to judge similarity and was used for the classifications.

3.1.2. Step 2.2: Snowball sampling. Snowball sampling is a nonprobability sampling technique [42]. It involves starting with the review of one or two seminal research papers and/or books and pursuing references of references and electronic citation tracking [43]. This gives rise to an external validity threat with respect to the reliability of literature review [44]. Threats to external validity are conditions that limit our ability to claim that our literature study is as complete and thorough as possible [44]. The reliability has been addressed as far as possible by starting the snowball sampling by going through the reference lists of all the papers selected for systematic mapping for example. As a result, 13 additional papers were identified. The value constructs identified in both the systematic mapping and the snowball sampling are detailed below. Complete set of papers and value constructs are available online [40].

3.2. Threats to validity

3.2.1. Publication bias. Publication bias refers to the general problem that positive research outcomes are more likely to be published than negative ones [45]. This is considered as a minor threat, because the research questions in this mapping are not geared towards the performance of a specific solution for the purpose of a comparison. The same reasoning applies to the threat of sponsoring in which certain methods are promoted by influential organizations [46], and negative research outcomes regarding this method are not published. The sources of information were not restricted to a certain publisher, journal or conference such that it can be assumed that the breadth of the field is covered sufficiently. However, there is a trade-off between considering as much literature as possible and, at the same time, accumulating reliable information. Therefore, grey literature (technical reports, work in progress, unpublished or not peer-reviewed publications) was excluded [46].

DOI: 10.1002/smr

3.2.2. Threats to the identification of primary studies. Only two databases were searched (Scopus and EngineeringVillage), because of the fact that Scopus has relatively lesser weaknesses compared with databases such as IEEEXplore, ACM Digital Library, SpringerLink, ISI Web of Knowledge and ScienceDirect [47]. Additionally, EngineeringVillage was also searched to retrieve as many papers as possible.

Another potential threat related to the identification of primary studies is that among the selected studies some have been written by the authors' colleagues. However, the selected studies were the ones identified through the developed search strategy (which was reviewed by an external person) thus this threat is not huge for the selection.

3.2.3. Threats to selection and data extraction consistency. Because of the scope of the systematic mapping, there was a need to develop efficient (in terms of execution time) and effective (in terms of selection and data extraction consistency) strategies for papers selection and data extraction. One of the main aims of defining a mapping protocol is to reduce researcher bias [46] by defining explicit inclusion/exclusion criteria and a data extraction strategy. A well defined protocol increases the consistency in selection of primary studies and in the following data extraction if the mapping is done by multiple researchers. One approach to further increase the validity of the mapping results is to conduct selection and data extraction in parallel by several researchers and then crosscheck the outcome after each phase. In the case of disagreements they should be discussed until a final decision is achieved. Because of the large amount of initially identified studies (558) it was impossible to implement this strategy. Therefore, as proposed by Brereton *et al*. [48], paper selection and data extraction were piloted and the consensus was improved iteratively. Two issues were addressed through piloting: first, the selection criteria and the data extraction form were tested for appropriateness, and second, the agreement between the researchers could be assessed and discrepancies streamlined. Although it can be argued that this strategy is weaker in terms of consistency than the previously mentioned crosschecking approach, it was a necessary trade-off to fulfill the schedule and the targeted breadth of the systematic mapping.

4. LITERATURE REVIEW RESULTS

This section presents results of literature review (Step 2, see Figure 1). Table II gives an overview of the value constructs identified and a short description of each aspect and the corresponding sources (the sources, e.g., P1, P2..P*n*) can be found online [40].

Several contributions have been made integrating economic and value perspectives. Aurum and Wohlin [49] emphasized that adding value is an economic activity that has to be taken into account from a business perspective. Value is created when a company makes a profit by succeeding in the market. Thus, the critical success factor for software companies is their capability to develop and deliver a product that satisfies customer requirements while offering high value that provides increased support for market success [2]. Thus, value is created for the customer [P3, P10, P30, P33, P35, P36, P42, P43, P49, P50, and P60]. In this context, it appears logical to consider customer value as a strategic merit to assess the value of a product and also to assess the overall value of the business [21].

From the customer's perspective, Rönkkö *et al*. [P44] proposed a value decomposition matrix based on the concept of value and utility in software business research and value of software as technology. They presented three value constructs, namely, intrinsic value, externalities [P68], and complementary value [P68]. Intrinsic value has been defined as the value that is embedded in the software as functionality and attributes such as security and reliability [P44]. Complementary value has been defined as a value that is created by combining a piece of software with another good or service [P44]. If the software is used for communication, it is subject to network externalities and its value is dependent on the amount of other users of the software, relevant to the focal user. Perrey *et al*. suggested delivery process value as one of the value constructs to be considered when measuring customer perceived value [P56]. Several others discussed that user experience is another fundamental value construct, among others, which contributes to the success or failure of a software

DOI: 10.1002/smr

Table II. Summary of results of systematic mapping and snowball sampling.

Value construct	Description	References
Customers perspective		
Perceived value (PV) – use value	Benefits derived from the product/feature. It is the trade-off between perceived benefits and the cost of ownership.	P21, P27, P39
Intrinsic value	It is embedded into the software as functionality and attributes, for example, usability, security.	P28, P31, P34,P55
Functionality	The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.	P55
Reliability	The capability of the software product to maintain a specified level of performance when used under specified conditions.	P55
Usability	The capability of the software product to be understood learned, used and attractive to the user, when used under specified conditions.	P55
Maintainability	The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.	P55
Portability	Measures such attributes as the behavior of the operator or system during the porting activity	P55
Delivery process value	Quality of process in installing/upgrading/receiving the product	P56
Network externalities	The amount of other users of the software product that are relevant to the focal user	P18, P44, P53, P68
Complementary value	A complement can be defined as a product/service, which increases the value of another product/service so complementary value refers to value, which is created by combining a piece of software with another good/service	P44
User experience value	A factor in addition to Usability, which may contribute to the success or failure of the product in the competitive market	P57-P59
Customer lifetime value	Represents a profound supplier-oriented understanding of customer value which is measured as profit streams of a customer across the entire customer life cycle	P60
Retention rate	Is a factor which is typically defined with regard to the individual customer that he/she remains loyal to a particular supplier and keeps yielding expected revenues and costs within a fixed period of time	P60
Revenue	Profit, money made by the company's customers	P60
Costs	Costs incurred by the company's customers	P60
Internal business perspective		
Production value	Represents an aggregated value of software production process, which may involve its physical value (PV) and its market purpose and values	P63
Market requirements value	Represents the production value with respect to a given market requirement	P63
Physical (project) value	Represents the integration of both product's features and software process deployment for better organizing critical complexities within the production system. Used to estimate the real added value of production processes and final product, and its influence on markets	P5, P23, P25, P41, P47, P48, P63
Physical value related to Time (PVt)	Represents the PV wrt Time. A product being evaluated, adjusted and put into market within a shorter time production cycle will have higher PVt	P63
Physical value related to Quality (PVq)	A company that has more flexible organizational structure to deal with virtual development processes will have higher PVq - organization	P63
Physical value related to Quality of Process (PVq-process)	A company that uses 'industry best practices' for process improvement and quality assessment will have a higher PVq-process	P63

DOI: 10.1002/smr

Table II. *Continued*

Value construct	Description	References
Customers perspective		
Physical value related to Quality of product (PVq-product)	A product that exposes functionalities with higher quality benefit according to industry standards will have higher PVQ-product	P63
Architectural value	Related to the architecture and design aspects	P11, P51, P55
Physical value related to Cost (PVc)	A product being developed and marketed with lower production cost will have higher PVc	P63
Differentiation value	The differences of a product or offering from others, to make it more attractive to a particular target market. This involves differentiating it from competitors' products and one's own product offerings	P64
Financial perspective		
Shareholder's value	Is determined by estimating the economic value of an investment by discounting forecasted cash flows by the cost of the capital	P61, P6
Innovation and learning perspective		
Value of technology	The potential value that is embedded in the subject technology itself	P8, P67
Value of market	Practical value of subject technology that is materialized in market or in business process	P67
Intellectual capital value	Software product and process knowledge	P50, P65, P66

product in a competitive market [P57–P59]. Exploring the customer perceived value definition further; one can see that the customer lifetime value concept represents a profound supplier-oriented understanding of customer value [P60]. Customer lifetime value is measured as profit streams of a customer across the entire customer life cycle. It is an application of contemporary finance to the evaluation of customer retention [P60]. However, in the existing software engineering literature this has not been considered when determining the value of a software product.

While considering the internal process perspective, Lei *et al.* [P63] presented a value model tree to address the common challenges faced by small-to-medium sized companies when entering the international market. The proposed value model tree specifies general relationship decomposition from final value, to market inputs and critical elements of the production processes, and human values related to risk attitude and time preferences. The production value represents an aggregated value of the software production process, which may involve its physical value and its market purpose and values [P63]. The proposed value model helps to visualize the internal production value with respect to the market requirements, production processes (project), and marketing activities. In addition, several other authors have discussed project value while taking decisions, for example, [P5, P23, P25, P41, P47, P48, P63].

From the financial perspective, shareholder value is a fundamental aspect for business valuation, in addition to customer perceived value and internal processes value, and needs to be considered while decomposing software value from the company perspective. In financial and business literature, increasing shareholder value was the central theme for the last two decades [P61, P62]. As a result, many organizations aimed at maximizing shareholder value and were constantly looking for new opportunities to maximize economic value added, or EVA, which led to the pursuit of attaining short-term objectives and targets [P61]. For example, prime focus had been on continuous increase of quarterly profits by delaying needed investments. Considerably less attention was given to long-term objectives, like investing in effective and efficient development processes, innovation, the quality of employees and the workplace. However, this approach makes sustainable growth hard to maintain and leaves almost no room to increase customer satisfaction with new innovative products or services [P61]. As a result, this approach affects the customer perceived value of the software product and needs to be considered when drawing a complete picture of software value. However, this aspect has not been discussed explicitly in VBSE literature.

DOI: 10.1002/smr

In the current emerging competitive environment, product and service innovations differentiate a company from its competitors; technology valuation is gaining popularity [P67]. Thus, there has been a realization that the value of a corporation and business cannot be gauged without knowing the value of the technological assets [P67]. In the absence of knowledge about maturity, life cycle, contribution ratio, scope of application and standardization of the technology, the measurement/valuation of innovation with respect to technology would be deceiving and irrelevant. Consequently, any product development based on a specific technology would affect the perceived value of the product. Others have discussed the value of intellectual capital (IC) as a fundamental value construct to be considered for achieving long term prosperity and sustainability(see, e.g., [P50, P65, P66]).

All of the value constructs identified during the literature reviews were used as input to creating the software value map presented in Section 5. Although the focus of systematic mapping was to extract value constructs from the selected papers; however, Table III shows the top five venues where most of primary studies have been published.

4.1. Measurement and valuation methods

While reviewing economics, business, management, and VBSE literature, we collected measurement solutions used to measure/value a specific value construct. This section gives a brief summary of measurement and valuation concepts. This will help readers understand the measurement solutions listed for the value components in the software value map.

Measurement can be defined as 'the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules' [22, 23]. A central goal of measurement is to capture the characteristics of entities and manipulate them in some formal way. Examples of software entities are products, processes, or resources of different types. Attributes are defined as the properties that an entity possesses. A measure maps an empirical attribute to the formal, mathematical world. A measurement unit determines how an attribute is measured. When measurement units are considered, the measurement scale type implied by the unit needs to be understood. The most common scale types are: nominal, ordinal, interval, and ratio. It is fundamental to be aware of the unit's scale types because it determines the admissible transformations that can be applied to particular units [24].

There are two broad types of measurement: direct and indirect. Direct measurement of an attribute is a measurement that does not rely on measurement of any other attribute. Whereas, indirect measurement of an attribute is measurement that entails the measurement of one or more attributes [25].

The valuation of software product/projects/processes depend on a detailed analysis of underlying costs and benefits. However, the valuation of software products and related activities is so far nearly an unexplored area in the software engineering context [3]. In economics, various valuation methods ranging from intuitive judgment to complex options models [14] have been developed and utilized. Although individual methods may differ from one another in terms of the criterion and procedures used, the 'value' of software is articulated from different perspectives using score, index, or monetary value. Among others, scoring method has been widely used. For example, while performing customer value analysis, a number of value constructs (performance, usability, usefulness) can be given to the evaluators (in this case the customers' representative) and then evaluators can subjectively rate the score for each factor for the product and other competing products. Similarly, this can also be conducted for innovative technology valuation. Scoring models

Table III. Top five publication venues.

Publication venue	Number of papers
IEEE Software	5
International Workshop on the Economics of Software and Computation	3
EUROMICRO Conference on Software Engineering and Advanced Applications	3
International Workshop on Software Process Simulation and Modeling	2
Communications of the ACM	2

DOI: 10.1002/smr

are popular because they are simple and robust [14]. However, the score itself never conveys the real meaning of value; rather, it indicates the relative preference among alternatives.

Alternatively, index models can be used to develop a functional form of valuation. This type of model is more flexible, because it can accommodate more diverse measures such as ratio and percent. However, it is still a ratio or percentage, and fails to convey the actual value. On the contrary, the monetary value model attempts to measure the monetary worth by using capital budgeting methods to estimate the discounted cash flow to calculate net present value. For example, the traditional shareholder value model uses net present value calculations to calculate shareholder value [26].

Monetary value models can be further categorized into three basic approaches: cost approach, market approach, and income approach [26]. The cost approach is based on the economic principle of substitution that states a practical buyer would pay no more, and a willing seller can ask no more, for a product than the cost to create the intellectual asset of equal quality and utility. This approach has been applied mostly to calculate the cost of product development; however, it only calculates one factor for determining value.

The market approach is a simple and direct method that argues the value of a product is equivalent to what others in the market place have judged it to be. Customers are potential judges and methods like customer value analysis helps to calculate the monetary value of the software product; however, as discussed earlier, this approach is quite subjective. Additionally, in relation to embedded products the software is difficult to separate from the overall offering. Thus, a central problem for valuation when using external sources for the value estimation is that they see the entire product and not just the software. This is not completely handled by the proposed software value map; however, by combining internal value estimations, the software part of the value can be gauged [2].

The income approach is based on the rationale that value is determined by the income-producing capability of the product. The income approach is considered to be best suited for the valuation of intellectual property such as software, patents, trademarks, and copy rights [14]. It might be desirable but not always easy to calculate the monetary worth of every value construct, for example, while determining user experience value, defined as ‘a consequence of a user’s internal state (predispositions, expectations, needs, motivation, mood, etc.), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.), and the context (or the environment) within which the interaction occurs (e.g. organizational/social setting, meaningfulness of the activity, voluntariness of use, etc.)’ [27]; we have to resort to subjective scoring. Similarly, for estimating the value of usability or reliability, we can use the measurements and metrics proposed in ISO 9126 [15].

The central challenge lies in the fact that different valuation models are used for valuing/measuring different value constructs, and from different perspectives, which results in incompatibility between the measured values.

5. STEP 3: SOFTWARE VALUE MAP

The results from the literature studies were used to create the software value map (see Figure 1, Step 3). The SVM shows a consolidated view of the value concept, presenting ‘true value’ as a complex interaction of perspectives and views. This section presents the SVM, and the underlying concepts, structure, and taxonomy, to categorize and describe value constructs.

5.1. Taxonomy

The taxonomy used to categorize the value constructs was inspired by the balanced scorecard (BSC) [50, 51]. BSC can be defined as a set of measures that gives managers a fast but comprehensive view of the business using four main perspectives, namely, the financial, customer, internal process, and innovation and learning, each described below [50, 51].

The financial perspective contains aspects that address the company’s implementation and execution of its strategy, which are contributing to the bottom-line improvement of the company. It represents the long-term strategic objectives of the organization and thus it incorporates the tangible outcomes of the strategy in traditional financial terms [50, 52]. Some of the most common financial measures that are

DOI: 10.1002/smr

incorporated in the financial perspective are earned value analysis, revenue growth, costs, profit margins, cash flow, net operating income, and customer value analysis.

The customer perspective defines the value proposition that the company will apply to satisfy customers and thus generate more sales to the most desired (i.e., the most profitable) customer groups [52]. Measures that are selected for the customer perspective should measure both the value that is delivered to the customer (value proposition) with respect to the perceived value, which may involve time, quality, performance and service, and cost, and the outcomes that come as a result of this value proposition (e.g., customer satisfaction and market share).

The internal process perspective is concerned with the processes that create and deliver the customer value proposition. It focuses on all the activities and key processes required in order for the company to excel at providing the value expected by the customers both productively and efficiently. These can include both short-term and long-term objectives and incorporating innovative process development to stimulate improvement. Quality, cycle time, productivity, and cost are some aspects where performance value can be measured [50].

The innovation and learning perspective is the basis of any strategy and focuses on the intangible assets of an organization, mainly on the internal skills and capabilities that are required to support the value-creating internal processes. The innovation and learning perspective is concerned with the intellectual capital categorized as human capital, structural capital, and the organization capital of a company [50, 53].

The balanced scorecard combines the financial perspectives with operational perspectives to provide a consolidated view of the business performance and was thus considered as a good base for the categorization in the creation of the software value map.

5.2. *Structure and definitions*

Several different value constructs were identified; each needs to be specified and placed. Thus, the central challenge while creating SVM was to categorize and present all the value constructs in a clear and structured manner. A nested structure with each value perspective (VP) being the parent node was chosen (based on concept-centric approach [54]). Each VP has nested clusters of nodes called value aspects (VAs), and in turn a VA can have subvalue aspects (SVAs). The end nodes are called value components (VCs). Each concept is detailed below along with attributes specified for each. Table IV shows the corresponding attributes specified for every VP/VA/SVA/VC.

The description of attributes is given below:

1. *ID*: Every VP has an ID prefixed with VP X where $X = 1, 2, 3, \ldots . n$. Every VA has an ID prefixed with VA $X.Y$ where $X = 1, 2, 3, \ldots . n$ of the parent VP, $Y = 1, 2, 3, \ldots . m$. Every SVA has an ID prefixed with SVA$X.Y.Z$ where X and Y are the same as above $Z = 1, 2, 3, \ldots . p$. Every VC has an ID prefixed with VC $X.Y.Z$ where $X = 1, 2, 3, \ldots . n$ of the parent VP, $Y = 1, 2, 3, \ldots . m$ of the parent VA (or SVA) and $Z = 1, 2, 3, \ldots .$ p
2. *Name*: The name should reflect the perspective/aspect/subaspect from which the value is categorized. For example, *Value for Customer* is one VP that categorizes value constructs relevant from the customers' perspective. The name should not be longer than five words.

Table IV. Attributes of VP/VA/SVA/VC.

Attributes	Value perspective (VP)	Value aspect (VA)	Subvalue aspect (SVA)	Value component (VC)
ID	X	X	X	X
Name	X	X	X	X
Description	X	X	X	X
Motivation/rationale	X	X	X	X
References	X	X	X	X
Additional notes	X	X	X	X
Composed of	X	X	X	
Measured through				X

DOI: 10.1002/smr

3. *Description*: The VP, VA, SVA or VC description should not be longer than five sentences, and should describe the central essence of the aspect in broad strokes.
4. *Motivation/rationale*: This attribute should specify the motivation in the context of the perspective being stated. If the perspective is of *customer* it should illustrate the motivation/rationale from the customers' perspective.
5. *References*: This attribute specifies the references in literature that have introduced/addressed the VP/VA/SVA/VC.
6. *Additional notes*: This attribute contains any additional information about the VP/VA/SVA/VC.
7. *Composed of*: A mapping of what VAs are included in the VP, what SVAs are included in the VA and so on.
8. *Measured through*: Contains a list of measurement solutions, presented in literature or otherwise, as to how to measure a VC. It contains a short description of the measurement solution and references to the literature where the measurement solution has been presented. However, this attribute does not contain an exhaustive list of measures that have been proposed in literature to measure/evaluate the corresponding VC.

5.2.0.1. Value perspectives, value aspects, and subvalue aspects. Figure 2 illustrates the structure where each VP holds a cluster of related VA. The VA can recursively have a cluster of SVAs. A VC is the end node, that is, the leaves in the model tree structure (see Figures 2 and 3). It represents a discrete value factor that is measurable. Every child node (whether it be a VA, SVA or VC) inherits context from the parent node. The complete SVM, and user manual with a step-by-step instructions for pattern creation can be found online [40].

Looking at Figure 2; 'Customer perspective' is one of four VPs (with seven attributes of its own). It contains two VAs: 'VA1.1: Perceived value' and 'VA1.2: Customer lifetime value'. Every VA also

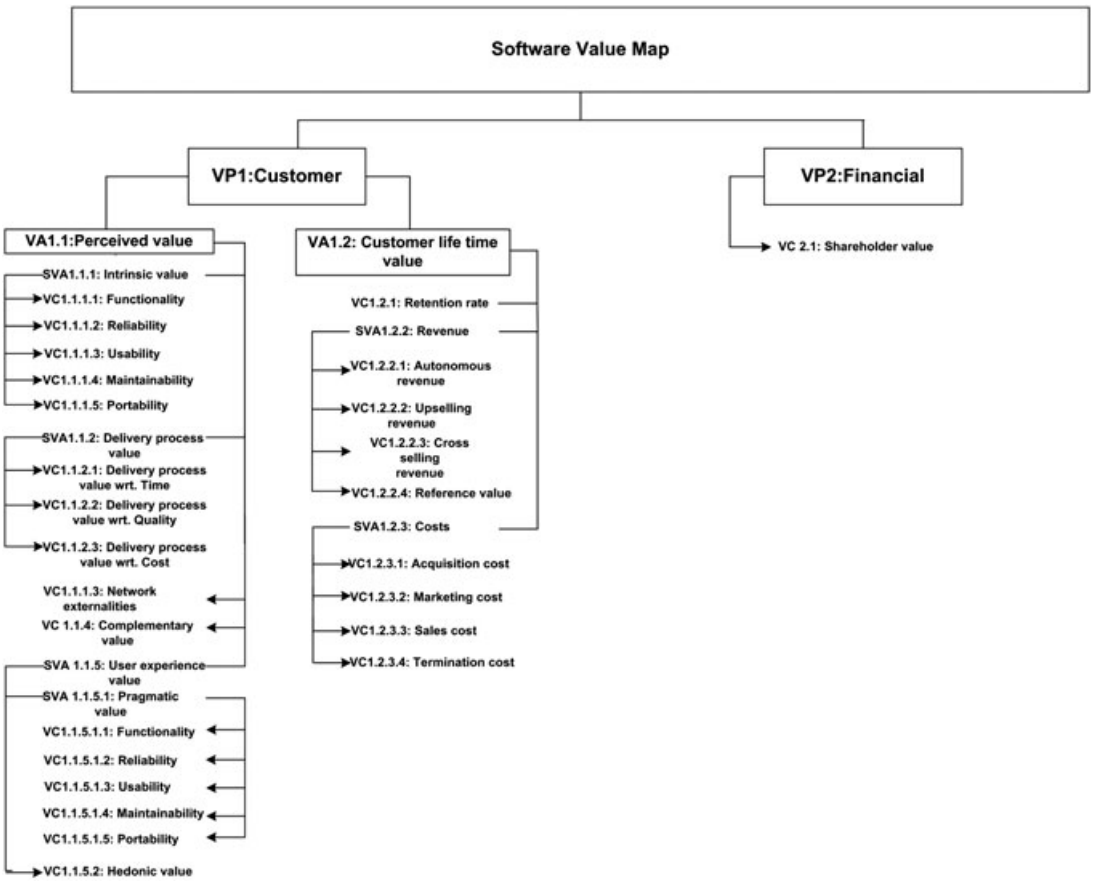

Figure 2. The first two value perspectives and corresponding VAs, SVAs, and VCs.

DOI: 10.1002/smr

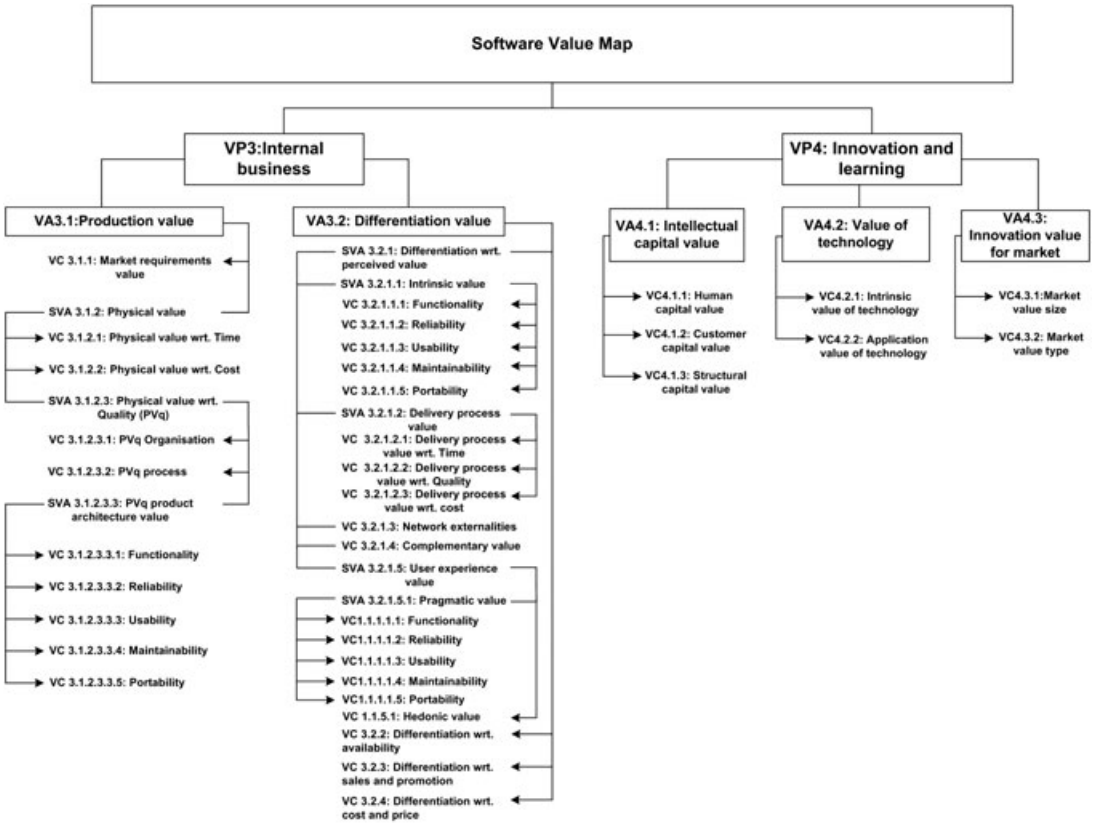

Figure 3. Last two value perspectives and corresponding VAs, SVAs, and VCs.

has seven attributes. Each VA further contains one or more SVAs or VCs. For example, VA1.1 contains three SVAs and two VCs namely: 'SVA1.1.1: Intrinsic value', 'SVA1.1.2: Delivery process value', 'VC1.1.3: Network externalities', 'VC1.1.4: Complimentary value' and 'SVA1.1.5: User experience value'. Looking further down the structure into 'SVA1.1.1: Intrinsic value', it contains five measurable VCs: 'VC1.1.1.1: Functionality', 'VC1.1.1.2: Reliability', 'VC1.1.1.3: Usability', 'VC1.1.1.4: Maintainability', and 'VC1.1.1.5: Portability'. The criteria used for measuring each of the VCs are given in the 'Measured through' attribute of the corresponding VC. The entire SVM together with detailed and complete attributes can be downloaded [40].

5.3. Interrelationships

Interrelationships in the software value map are possible and common among value aspects. We do not claim to map all possible relationships, the rationale being SVM is an evolving entity and so are the interrelationships. The interrelationships detailed thus far in SVM were identified using three sources: (1) interrelationships that were stated in the selected papers, (2) interrelationships that were obvious during categorization and placement of VAs/SVAs and VCs, and (3) interrelationships that were identified during the creation of patterns in the industry case study (see Table V). For example, 'SVA: PVq-product architecture value', from the internal business perspective is related to the external quality attributes of a product's perceived importance from the customer perspective 'SVA: intrinsic value' [P44]. This is due to that if the software architecture is not designed with flexibility, reliability, usability, maintainability, and portability in mind, it will not be possible to provide users/customers with software that is in turn flexible, reliable, usable, maintainable, and portable. Moreover, when determining differentiation with respect to perceived value for customer 'SVA: Differentiation wrt. perceived value', it is related to the subvalue aspects of perceived value from customer's perspective 'VA: perceived value'. This is particularly important when companies decide

DOI: 10.1002/smr

Table V. Interrelationships within software value map.

VA/SVA	VA/SVA
VP3: Internal business perspective	VP1: Customer perspective
SVA3.1.2.3.3: PVq-product architecture value	SVA1.1.1: intrinsic value
VP3: Internal business perspective	VP1: Customer perspective
SVA3.2.1: Differentiation wrt. perceived value	VA1.1: perceived value
VP4: Innovation perspective	VP4: Innovation perspective
VA4.2: value of technology	VA4.2: innovation value for market
VP1: Customer perspective	VP1: Customer perspective
SVA1.1.5.1: Pragmatic value	SVA1.1.1: intrinsic value

differential advantages of their software products as they would strategically plan to differentiate on value aspects that are valued more by their customers.

Looking into the innovation and learning perspective, 'VA: value of market' is related to 'VA: value of technology'. For example, duration of income in 'VA: value of market' is dependent on the life of subject technology 'VA: value of technology' and amount of income is influenced by proprietary position or contribution ratio of subject technology [P67].

Thus, if interrelated value aspects are considered independently, a number of VCs may be overlapping or redundant and the accuracy of valuation may deteriorate. One of the contributions of the software value map is to make these interrelations explicit to avoid blind addition of the overlapping and redundant value components. However, this has not been addressed during the industrial evaluation of SVM, and is an area of future research during evaluation and evolution of the SVM.

6. STEP 4: INDUSTRIAL EVALUATION THROUGH APPLICATION OF SVM

The software value map in itself offers a large overview of the concept of value from multiple perspectives, although useful for many tasks, practitioners in a company generally work with one task at a time. For example, a project manager may struggle with the decision whether or not to put extra resources into the development of a certain component to make it easily reusable, and a product manager may struggle with the decision relating to requests for product customizations. In the case of the project manager, he needs to evaluate the potential value of making, for example, a feature or function reusable. In the case of the product manager, there has to be an evaluation of the potential value (or lack thereof) of the customization and the ability to weigh customization requests against each other.

In these and other specific cases using the complete value map is impractical, because traversing the entire map would imply going through all the value components. For this reason, the concept of 'impact evaluation pattern' was introduced, consisting of a subselection of value components. Each pattern is adapted for a specific task, directly supporting the practitioner. These impact evaluation patterns were created using the data collected in the evaluation study (Step 4, see Figure 1) conducted at Ericsson, our main industry partner for this project.

Ericsson is a world leading telecommunication system provider, offering a wide range of products and solutions. Products are developed and sold as generic solutions offered to an open market, although customized versions of the products are also developed. Within Ericsson, like other software product development companies, they are faced with the challenge of making decisions incorporating value aspects and value components relevant from different perspectives rather than taking product development decisions based only on short term costs and immediate sales.

The idea behind impact evaluation patterns is described in Section 6.1, and Section 6.2 details the steps for pattern creation. In total, four patterns were created at Ericsson, for four different roles involved in a particular decision-making scenario (fully described in detail online [40]). Section 6.3 describes in detail how the created patterns for decision-support can be and were used. Section 6.4

DOI: 10.1002/smr

concludes the evaluation section by presenting lessons learned from the use of the software value map and the creation of patterns.

6.1. The concept of impact evaluation patterns

The concept of impact evaluation patterns was inspired by software design patterns. In software engineering, a design pattern is a general reusable solution to a commonly occurring problem in software design [55]. The same philosophy was used to identify impact evaluation patterns in different decision-making scenarios. An impact evaluation pattern can be described as a generally reusable solution for a commonly occurring decision-making challenge in a particular scenario. For example, a product manager can use an impact evaluation pattern for initial screening to decide if a set of new requirements should be selected for implementation in the product or not. For large companies, like Ericsson, who have multiple products, and a large number of people working in the same role, common evaluation criteria for a particular decision is fundamental for homogenous decision-making, which is achieved using a pattern.

Each pattern is documented/specified through a number of attributes, as can be seen in Table VI. Here we can see that attribute 5: 'Value aspects' would contain VCs that are used in the pattern, for example 'VC1.1.1.1: Functionality', 'VC1.1.1.2: Reliability', and 'VC.1.1.2.1: Deliver process value wrt. time' can be included in a requirement selection decision-making pattern. Attribute 6: 'Impact evaluation criteria' would contain rubrics-based criteria to evaluate the impact on value aspects given in attribute 5.

6.2. Step 4.1: Pattern creation process

A total of four patterns were created for four different roles as a part of the industrial evaluation case study at Ericsson. This section goes through the steps involved in the creation of impact evaluation patterns (Step 4.1, see Figure 1).

6.2.1. Step 4.1.1: Scenario selection. The first step in evaluation was to identify a scenario. This could be any scenario that has different decision-making activities related to the evaluation of software creation, updating, customization, and/or modification. A scenario can have one or more discrete decision-making activities. As a result of a joint discussion with the industry practitioners one major scenario was chosen, namely product customization (PC). The rationale being that

Table VI. Impact evaluation pattern attributes.

1. Pattern Name and Classification: A descriptive and unique name that helps in identifying and referring to the pattern.
2. Intent and Motivation: A description of the goal (decision-making challenge that is faced) behind the pattern and the reason for using it.
3. Also Known As: Other names for the pattern.
4. Applicability: Situations in which this pattern is usable; the context for the pattern.
5. Value aspects: A hierarchical branching of the VPs, VAs, SVAs and VCs used in the pattern.
6. Impact evaluation criteria: A rubrics-based criteria to evaluate the impact on value aspects given in the pattern
7. Relationships: If SVAs or VCs have interrelationships a description of what the interrelationships are and how they affect the overall pattern.
8. Consequences: A description of the results (valuation), side effects, and tradeoffs caused by using the pattern.
9. Related Patterns: Other patterns that have some relationship with the pattern; discussion of the differences between the pattern and similar patterns.
10. Involved stakeholders:
• → A list of stakeholders involved in creating the pattern.
• → A list of stakeholders involved in verification of the pattern.
• → A list of stakeholders actually using the patterns.
• → A list of stakeholders maintaining the pattern.
• → A list of stakeholders providing measurements/assessments for the value components included in the patterns.
Because an impact evaluation pattern would be a living entity it is necessary to keep a record of all the stakeholders involved to ensure that pattern creation, verification, usage ,and maintenance responsibilities are explicitly stated and in case of modifications required in the pattern, relevant stakeholders can be contacted.

DOI: 10.1002/smr

Ericsson has many PCs in their day-to-day work, and the potential optimization and introduction of a rich view in relation to evaluation of PCs was recognized as potentially very beneficial.

A product customization can be defined as a modification of the source code of an Ericsson product, initiated by a customer request. A product customization can consist of new functionality in a product that is not planned for in the product roadmap, or new functionality in a product that is planned for in the product roadmap, but not in the timeframe as requested by the customer. In case of PCs the actual development costs will have to be paid by the customer, and fast delivery is central. However, the potential value impact of a PC on the present product implied by a customization is not always analyzed, because the estimation of 'value' was often limited to cost and estimated revenue. To achieve a deeper value impact analysis, an impact evaluation pattern creation was initiated.

Within product customization there can be different decision-making activities at four different stages, as identified in relation to how Ericsson optimally wanted to work:

1. Initial screening: This involves a quick value analysis, should the PC in question be realized or dismissed.
2. Pre-analysis: This requires a relatively detailed analysis from business and technical perspectives.
3. Enable reuse: This activity requires performing value analysis to identify which PC as such can be made reusable or what parts of the PC should be made reusable to enable faster handling of future PCs and faster delivery of product releases. The main decision being should one spend extra resources to develop the PC in a reusable fashion or should it be carried out with time-to-market as a focus.
4. Post delivery analysis: This involves performing value analysis to measure if the value predicted for a PC in the 'Initial screening' stage is gained or not. This analysis is conducted after the PC has been implemented and delivered to the customer/market.

Within the PC scenario, one of the listed decision-making activities was selected to be studied, and included in the pattern, namely pre-analysis. The pattern created for pre-analysis used by a system manager for impact evaluation is used as an example in this paper.

6.2.2. Step 4.1.2: Role identification. The next step, after the selection of a scenario and corresponding decision-making activity, was to identify the roles involved in that decision-making activity. These practitioners would be the main contributors in the creation of the pattern. Members of the strategic product management organization at Ericsson supported in the identification of the roles and participants for the study.

Table VII contains the roles directly involved in pre-analysis decision-making activity. At least two practitioners in each role were chosen to participate in the evaluation study.

After the participants were selected, a 3-h work session was held with each participant. A work session started with a 15–20 min introduction followed by Step 4.1.3 and Step 4.1.4, described below. The work sessions were performed in an interactive manner, discussing and elaborating on what aspects were presently considered critical for the decision-making activity, and what additional analysis/value components could be beneficial and should be added to the pattern.

Table VII. Roles and participants in the study.

Role	Description	Number of participants
Software product manager (SPM)	A software product managers deals with product release, product market and general functionality.	2
System manager (SM)	A system manager is responsible for the technical aspects of a software product.	2
Solutions architects (SA)	A solution architect is responsible to design a solution given a set of requirements from the customer. The solution proposed may be then evaluated by system managers. SA represent the customer perspective.	4
Customer unit representatives (CU)	A CU representative is responsible for the business case of a set of requirements. He/she is also responsible for the pricing of the solutions. CU represent the business and customer perspectives	2

DOI: 10.1002/smr

6.2.3. Step 4.1.3: Eliciting current decision-making criteria. To create the impact evaluation pattern, it was important to first elicit what value aspects were considered at present by the participants when deciding if a PC should be realized or not (prior to the introduction of the value map and impact evaluation pattern). This activity was important to 'benchmark' the current decision criteria. The software value map was not shown to the participants prior to this step, because we did not want to influence their answers. In total the following information was elicited from the participants:

1. Value aspects: Value considerations that the decision maker has in a particular decision-making scenario.
2. Description: In order not to misinterpret a value aspect it was necessary to record the description of every value aspect as stated by the participant.
3. Measured today: Against each value aspect, the participant was asked how the stated value aspect is measured/evaluated.
4. Should be measured: To identify if measurement/evaluation of the stated value aspect needs to be more refined or coarse, the participants were asked how it should ideally be evaluated/measured.

Table VIII gives an example of the type of answers collected from the participants.

As can be seen in Table VIII, system managers described cost, impact on different nodes, and architectural value among the criteria used for taking decision about a PC in the pre-analysis phase. However, for certain value aspects, like architectural value, they could not state which explicit architectural value aspects were evaluated. They were not satisfied with the subjective evaluations; however, it was difficult for them to suggest ideal measures for the value aspects they stated. The fundamental difficulty was to identify a common evaluation scale for all the value aspects.

6.2.4. Step 4.1.3: Eliciting ideal value aspects and components. After benchmarking the current decision criteria, all relevant value aspects and components were elicited from the participants, in essence trying to catch the ideal aspects. In this step, the value map was shown to the participants, and all the VCs in the value map were presented to the participants in a form. An extract of this form can be seen in Table IX (the complete form is available in the online manual [40].

For each value component, 'perspective and description' field contained one of the four perspectives: customer, internal business, financial, or innovation and learning. The description was copied from the 'Description' attribute of the VC in the SVM. The participants were asked to indicate the applicability of a particular VC in the specific decision-making activity. The choices ranged from fully applicable to not applicable. Furthermore, because one of the objectives of the case study was to elicit which value components participants wanted to consider ideally, it was required to elicit how coarse or refined they wanted a particular value component to be measured/evaluated. This information was elicited to make the impact evaluation patterns usable and useful.

6.2.5. Step 4.1.5: Data analysis and creation of impact evaluation pattern. Conducting Step 4.1.3 and Step 4.1.4, the work sessions indicated that currently decision-making was cost and immediate revenue centric. Although system managers implicitly consider effects on the architectural value of a product, they do not have any explicit measurements/evaluations of the architectural value when

Table VIII. Example of eliciting current value aspects.

What are the major decision scenarios that you are involved with?	Value aspects and value components			
Decision scenarios	Value aspects	Description	Measured today	Should be measured
Pre-analysis	Cost	How much it costs	Subjective evaluation	Rubrics- based
	Impact on different nodes	How the development of different nodes are impacted	Subjective evaluation	Some concrete measures
	Architectural value	How a PC affects the architecture of the product?	Subjective evaluation	No answer

DOI: 10.1002/smr

Table IX. Form for eliciting 'ideal' value components and their measures — example.

Value components	Perspective and description	Applicability	Measures
	Customer perspective	Fully applicable = 3, Applicable = 2, Kind of applicable = 1, Not sure = 0, Not applicable = −1	
Functionality	The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.		
Reliability	The capability of the software product to maintain a specified level of performance when used underspecified conditions.		
Usability	The capability of the software product to be understood, learned, used, and attractive to the user, when used under specified conditions.		
...	...		

deciding if a PC should be conducted or not. Furthermore, system managers mentioned that because they do not have architecture value evaluations as a specific value aspect in their work at present, it was almost impossible to show higher management that the PC in question, although promising higher short-term revenues, would have a devastating effect on the product's value in the long run.

To create an impact evaluation pattern for a role, system manager (SM) in this illustrated case, we set up rules for identifying what value components should be included in the pattern (based on the questionnaire filled by participants in Step 4.14). These rules are given below:

- Rule 1: For all the participants in one role (for example System management) take the average of the values assigned to every VC. If the average is greater than or equal to 2, it is included in the pattern
- Rule 2: If the difference between the values assigned by the participants (difference of opinion) is greater than or equal to 2, more analysis is required.

Figure 4 shows the value components selected by system managers that should be considered for analyzing a PC in question. Starting from the left, system managers believed that during pre-analysis they should evaluate functionality value from the customer perceived value perspective.

From the internal business value perspective, system managers were of the opinion that while analyzing a PC, competitive market requirements value from the resell perspective ('VC3.1.1: Market requirements value') should be evaluated. This is certainly relevant from the internal business value perspective as if the PC in question gives competitive advantage, and/or can be resold, it would add to the value for the business. System managers also selected some architectural VCs ('VC 3.1.2.3.3.1: Functionality', 'VC 3.1.2.3.3.1.3: Usability', 'VC3.1.2.3.3.1.4: Maintainability' and 'VC 3.1.2.3.3.1.5: Portability') to be evaluated for explicit analysis of how architecture of the product is affected by implementing the PC in question. It could be either improving architecture value or degrading it. Last but not the least, project value with respect to time and cost should be evaluated.

The attributes of the created impact evaluation pattern are given in Table X.

'VC 3.1.2.3.3.1: Functionality' in Figure 4 is bold because it is in the critical path of SM's impact evaluation pattern (for details see Sectoin 6.2.6).

For some VCs it was easy to put measurement/metrics parts (for example VC3.1.2: Physical value wrt. cost), where monetary measures can be used. For others it was hard to estimate impact in, for example, monetary terms (for example, VC1.1.1.1: Functionality value); therefore, it was agreed to use rubrics-based expert opinion criteria for impact evaluation. The rubrics designed for impact evaluation of VCs given in the system managers' pattern are given in Table A.II (see Appendix B).

From Figure 4 , it can be seen that an impact evaluation pattern for pre-analysis for SM resulted in having seven VCs to be evaluated for each decision. For software product managers (SPMs), the VCs

DOI: 10.1002/smr

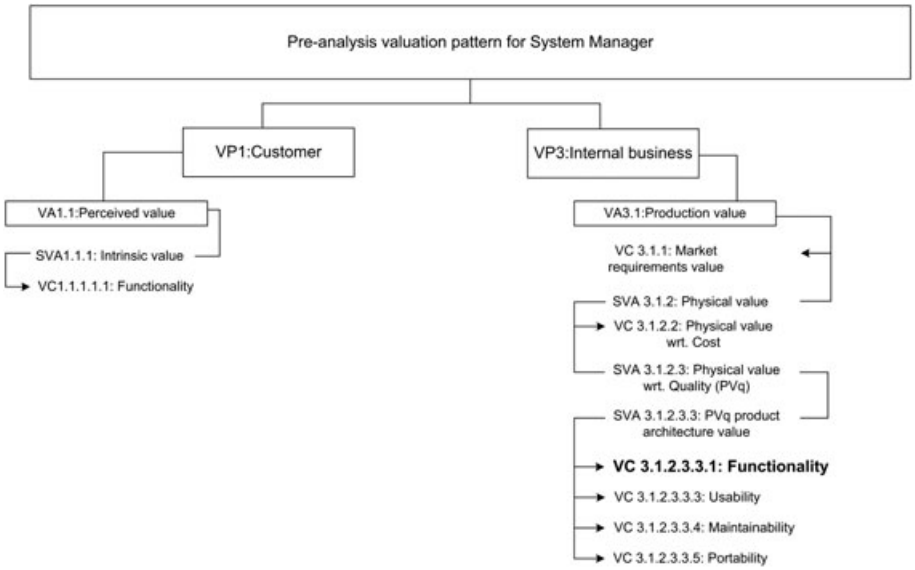

Figure 4. Pre-analysis impact evaluation pattern for system managers with critical path.

Table X. Attributes of pre-analysis impact evaluation pattern.

1. Pattern Name and Classification: Pre-analysis impact evaluation pattern for SM
2. Intent and Motivation: To perform a detailed value analysis from the business and technical perspectives This was stated as the main motivation by Ericsson practitioners to have pre-analysis impact evaluation pattern.
3. Also Known As:
4. Applicability: to analyze and decide if a PC in question should be carried out or not
5. Value aspects: see Figure 4
6. Impact evaluation criteria: see Table A.II in Appendix B
7. Relationships: Business value contains **Value for Customer** and **Product Value**
8. Consequences: //to be added when the pattern is put into actual use
9. Related Patterns:
10. Involved stakeholders: SMs, SPMs and marketing department

in the pattern went up to 13. During the work sessions, although the participants realized that a lot of VCs are relevant and should be considered in their decision-making, they raised concern as to the practicality of using patterns that were large (many VCs). This was a correct observation, and motivated the creation of the concept of critical path and value analysis triage (Step 4.1.6 in Figure 1), detailed below.

6.2.6. Step 4.1.6: Value analysis triage and critical path. In market-driven software product management and development, optimization of resources expenditure for decision-making is critical [7, 30, 31, 56–58]. This demands scalable and efficient decision support methods/models/tools to be developed.

The software value map (large) was tailored and a subset of VCs was selected to create a pattern (smaller), fitting a specific decision-making scenario and role. As the next step to even further optimize the potential use of the value map, critical path was introduced as a concept. The critical path is a 'deal breaker' part of a pattern that is one or several VCs that can be evaluated first when using a pattern. This path contains VC(s) that are most critical for decision-making. If these do not produce satisfactory results, the rest of the pattern can be ignored, because the decision is 'No' in any case. This triage of value aspects was inspired by Simmons [59] and Davis [60] and their application of the concept of requirements selection. We do the same using critical paths, but apply it on what VCs to look at first when using a pattern, to, for example take a decision whether or not to accept a product customization for a product.

DOI: 10.1002/smr

To identify the VCs included in the critical path, another rule was introduced (as elicited in Table IX).

- Rule 3: For all the participants in one role (for example system management) take the average values assigned to every VC. If the average is greater than or equal to 2.5, the corresponding VC is a critical path candidate.

In the example used here, the system managers considered 'VC3.1.2.3.3.1: Functionality' to be in their critical path (see Figure 4, the VC on critical path is made bold and has larger font size). System managers belong to the development unit. Their central responsibility is to sustain the existing functional design of the architecture, thus functionality value from the product's architecture value perspective ('VC3.1.2.3.3.1: Functionality') constitutes critical path in their impact evaluation pattern.

There can be more than one VC in a critical path as can be seen in the patterns for SPMs, SAs, and customer unit (CU) representatives [40]. During the creation of impact evaluation patterns using SVM, identifying critical paths for each impact evaluation pattern emerged as a fundamental activity.

6.3. *Step 4.2: Using the patterns*

Once the patterns and critical paths were created, an evaluation of their usability and usefulness was deemed a natural next step. A simple Excel-based (Magnus Wilson, Karlskrona, Sweden) prototype tool called the software value analysis tool (see Figure 5) was created to aid the practitioners when using their patterns. As an example Figure 5 shows the system managers pre-analysis impact evaluation pattern. For the examples here, PC requests are used; however, the use of patterns can be on any artifact or phenomenon (e.g., requirements). While evaluating a PC request against VCs in the pattern, the system manager simply has to choose one out of five options given in relation to each VC. For example, for the VC 'Market requirements value', the options are:

+2: More than *n* additional operators might buy the product with this functionality
+1: More than *m* additional operators might buy the product with this functionality
0: No additional customers will buy this functionality
−1: Operators will be hesitant to buy this functionality
−2: Operators will not buy this functionality

A system manager has the possibility to either evaluate the PC against the entire pattern or use the critical path VCs first for quick and dirty analysis. Once a PC has been evaluated against the patterns, several interesting visual aids can be employed, aiding discussions and decision meetings. Examples of such graphics are detailed below.

Figure 5. Snap shot of software value analysis tool.

DOI: 10.1002/smr

6.3.1. Theme-based analysis. In addition to patterns, themes were developed. Themes are effectively a weighting of value aspects depending on the current focus of the development organization. A theme can be anything that is mandated by the product strategies. For example, for one year/period a theme might be to focus on ARCHITECTURE or maybe on INNOVATION. In case of the innovation theme the value aspects such as 'differentiation value' and 'Innovation value of technology' (nested under that area in the SVM) are prioritized over other value aspects such as 'Architectural usability value' (and the other way around if using the architecture theme). By changing themes (one click in the tool) one can compare a specific PCs value in relation to a theme (which also can be seen as a benchmark to accept or reject a PC) not only based on the estimation of the value aspects, but also depending on timing and strategies.

Figures 6, 7, and 8 illustrate the use of patterns and the evaluation of a specific product customization. On the horizontal axis all VCs in the pre-analysis impact evaluation pattern for system managers and VCs in the current theme are listed. On the vertical axis whole numbers showing value of VCs are listed. If the current theme is the 'architectural theme', then according to a system manager's evaluation, the product customization CIN-191 is perfectly aligned with the existing focus (see Figure 6) and can be accepted for implementation without any issues. However, if the theme is changed from 'Architectural theme' to 'Innovation theme', Figure 7 clearly shows that CIN-191 is not the best choice for implementation, because it does not add positively to the value components prioritized with the 'Innovation theme'.

This one-click theme-based analysis was very appreciated by the practitioners during the evaluation of the patterns and tool. Similarly, such an analysis can be done simultaneously for multiple PCs to select the ones more aligned with the current theme. For example, Figure 8 shows CIN-191 and a new product customization MOBI-056. Both are well aligned with the current focus and positively impact the architectural value of the product, however, CIN-186 could, if implemented, decrease the maintainability value of the architecture. Through this tool practitioners can quickly benchmark PCs against each other, and against the current theme.

6.3.2. Internal versus external value analysis. Another interesting analysis possibility provided through the use of patterns is the placement of, for example, PCs in relation to their impact on internal and external value. All the VCs in SVM are classified as contributors to either external or internal value.

Figure 9 shows four quadrants in which PCs can be placed. The top-right quadrant is the one where PCs positively impacting both external value (for the customer), and internal value (for the development company) are placed. The top right quadrant can be seen as a win–win situation both for the customers and Ericsson in this case. Whereas, the top-left quadrant would contain PCs that positively impact/add to the internal value, but negatively affect external value. For example, a PC might improve the architecture's maintainability while simultaneously negatively affecting the efficiency of use of the product. Also, if the cumulative internal value (after summing values for all VCs contributing to internal value) is positive and cumulative external value is negative (after

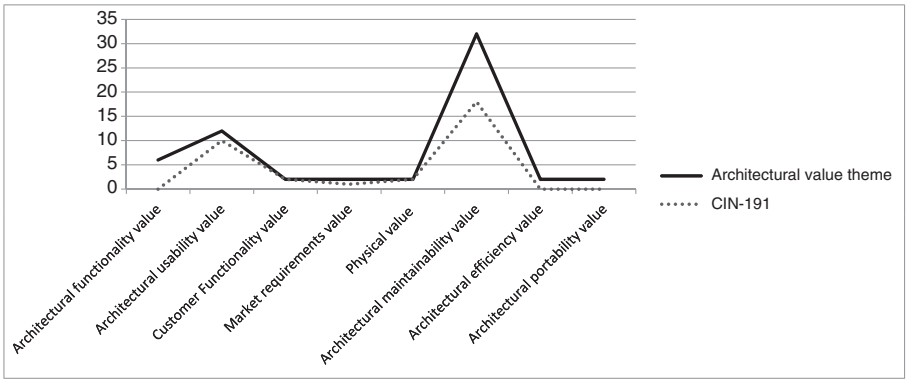

Figure 6. Theme-based analysis (architectural theme).

DOI: 10.1002/smr

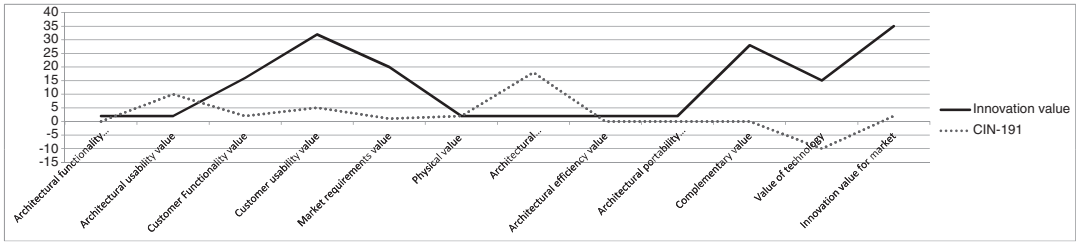

Figure 7. Theme-based analysis (innovation theme).

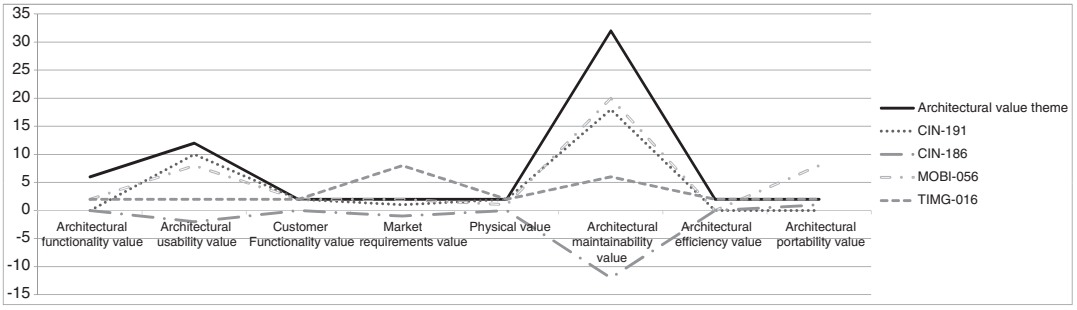

Figure 8. Theme-based analysis (architectural value theme) for multiple PCs.

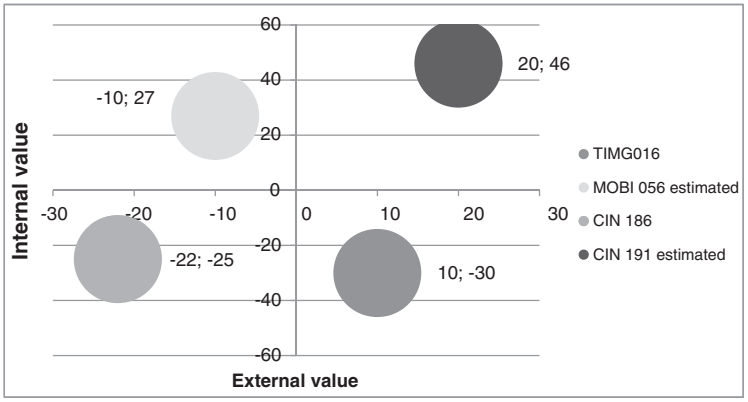

Figure 9. Analysis of PCs in relation to external versus internal value.

summing values for all VCs contributing to external value), the PC gets placed in this quadrant. For PCs in this quadrant, Ericsson needs to carefully analyze the gains in internal value with respect to losses in external value before accepting the PCs. The bottom right quadrant contains PCs that have higher external value, but negative internal value. In such cases, costs for avoiding/minimizing the negative impact on internal value need to be calculated and added to the cost of the PC. Such PCs can still be accepted if the customer is willing to pay the additional cost of the negative internal impact. The PCs lying in the bottom left quadrant are candidates for rejection because they neither add to external nor to internal value.

During industry evaluation of the patterns and tool, practitioners pointed out that the placement of PCs in relation to external value and internal value can be used in retrospect also, for analyzing the quality of decisions post-fact. The decisions made using pre-analysis impact evaluation patterns can be recorded and once every six months a sample of PCs can be picked and the pre-analysis impact evaluation pattern for each one of them is filled out again, based on the post-delivery facts (how were the internal and external values actually impacted), and compared with the estimates. For example, Figure 10 shows that for CIN-191 estimated external value and internal value impacts (20,

DOI: 10.1002/smr

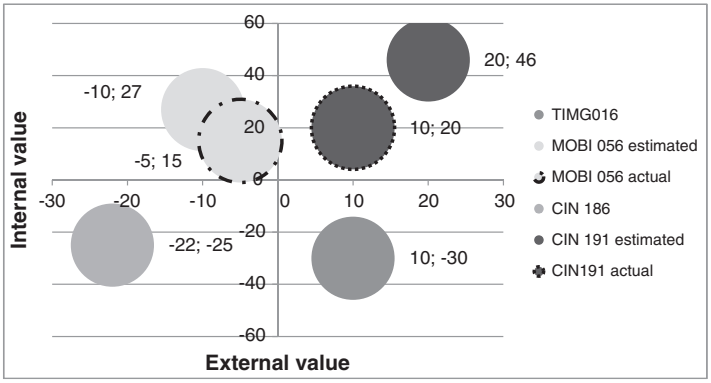

Figure 10. Post-analysis of PCs in relation to external versus internal value (estimated and actual).

46) were more than actually achieved (10, 20). Similarly, estimated internal value projection (27) for MOBI-056 was not actually achieved (15). On the other hand, the positive aspect is that actual negative impact on external value (−5) was less than the estimated one (−10). Keeping a record of decisions made based on impact evaluation patterns can help practitioners learn from their estimations and decisions and consequently improve the quality of both.

6.3.3. Role-based analysis. In addition to looking at comparing PCs, and PCs and themes, a comparison between roles can be performed. The central aim of this is to explicitly compare different perspectives and to be able to lift a PC's different impact on the product. In Figure 11, on the horizontal axis one can see all VCs in the pre-analysis impact evaluation pattern for system managers, and the VCs in the current theme are listed. On the vertical axis whole numbers showing value of VCs are listed. The bars represent the ideal values according to the selected theme, impact evaluation conducted by software product managers, and system managers according to their corresponding rubrics. From Figure 11 it is interesting to note that a product manager (PC_Prod mgmt_Initial_RM) deems the architectural usability impacted as 'positively', while the system manager (PC_Syst mgmt_Initial_RM) deems the same value aspect as being 'negatively' impacted. By catching these types of 'problems' and inconsistencies, explicit discussions can result in well-founded decisions without escalations because of implicit knowledge and judgment. Similar analysis can be carried out involving all the stakeholders involved in decision-making.

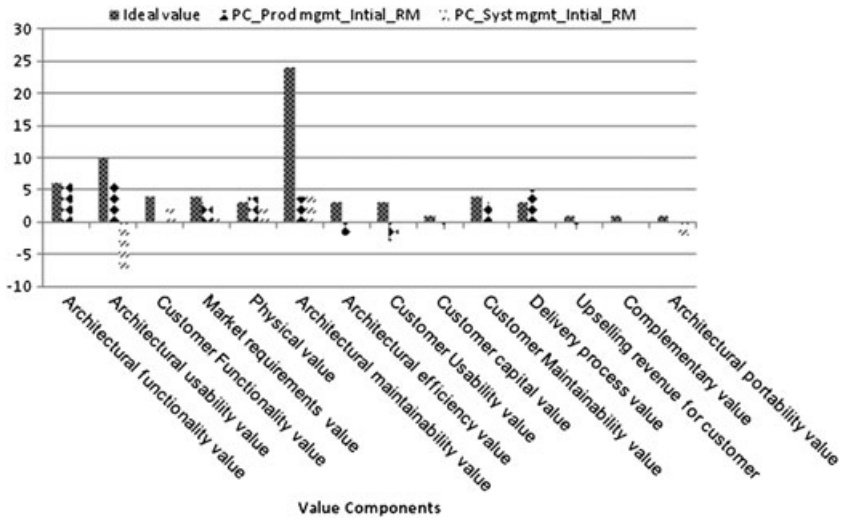

Figure 11. Role-based analysis of PC with respect to architectural value theme.

DOI: 10.1002/smr

6.4. Results and lessons learned

6.4.1. Challenges addressed by software value map and impact evaluation patterns. This section describes the lessons learned in relation to the use of SVM and patterns, and relates it to the challenges described in Table I (see Section 2).

6.4.1.1. Communication enabler. One of the major contributions of SVM is that it offers a unified view of value, which can be used by professionals to develop a common understanding and vocabulary pertaining to value, and acting as decision support to assure no value perspective is unintentionally overlooked when taking product management and development decisions (addressing Challenge C1, see Table I). This made communication between, for example, a solution architect, a product manager, system manager, and a customer unit representative not only possible (addressing Challenge C1), but explicit because SVM was used as a base for the communication and the impact of the decision could be weighed (addressing Challenge C4). Prior to SVM, communication was often based on an assumed definition of value, which varied between stakeholders. These variations were often not caught; rather the consequences of them were seen in failures and problems down the line.

6.4.1.2. Value-based requirements selection. To address *C2: Long-term versus short-term gains and losses* and *C4: Value-based impact and consequence analysis* (see Table I), SVM can be used as a consolidated view of value including value components required to be evaluated during requirements selection; balancing short-term and long-term gains and sacrifices keeping in view the entire product life cycle. For example, during requirements selection, in addition to short-term increases in customer value and company revenue, a company's and product's long-term sustainability view can also be considered. For example evaluating the effects of a requirement on the maintainability value of the product's architecture (VC 3.2.1.1.1.4, see Figure 3), human capital value (VC 4.1.1, see Figure 3) of the company and innovation value for the market (VA 4.3, see Figure 3) would achieve a more comprehensive (long-term) impact analysis of a certain decision. The overall trade-off between positive and negative impact on the present product offering can be estimated. This is central from many perspectives. For example, from a business perspective, the selection and realization of a feature might be good idea, but simultaneously the long-term effects pertaining to, for example, system architecture, might be very negative. Another less obvious example might be that the functionality value for one customer, and thus revenue, might be high when implementing Feature X, but the effect for other customers might be negative in relation to the inclusion of said feature, for example, decreasing interoperability value (part of VC1.1.1.1: Functionality, see [40]).

6.4.1.3. Reduced time-to-decision. Software value map was used as a basis to derive different impact evaluation patterns that can reduce time-to-decisions in relation to a wide range of software product development situations, like reuse and customer tailoring (for details see Section 6). An impact evaluation pattern can be described as a general reusable solution for a commonly occurring decision-making challenge in a particular scenario. The concept of 'critical path' was also introduced and evaluated to further reduce the time-to-decision by identifying the value components considered to be 'deal-breakers' within each pattern (for details see Section 6.2.6).

6.4.1.4. Retrospective analysis. If impact evaluation patterns are used for decision-making, a common value-based set of criterion is used by all the practitioners. By recording the decisions made using the patterns, for example the requirements selection pattern (the decision is documented according to the pattern's value components); a retrospective examination of release planning decision-making can be made, at a time when the consequences of requirements selection decisions are visible (thus addressing challenge C3: release planning quality). Also, by analyzing the decision outcome in retrospect, organizations can gain valuable knowledge of how to improve the requirements selection process and increase the chances of market success.

In summary, in addition to the stated points above, the industry practitioners at Ericsson participating in the creation and evaluation of patterns gave the following feedback in relation to the SVM and the use of impact evaluation patterns:

a. Software value map was viewed as a dictionary that offers a modular view, and common definitions of software value aspects, subvalue aspects, and value components

DOI: 10.1002/smr

b. It was possible to identify value components that might be important to consider in a certain scenario and for a particular pattern. Even if this was not an exact science, and many times too many value components were selected (as they could be relevant), the possibility of a more complete view beyond the obvious was welcomed. The challenge was to iteratively refine the patterns to make them complete enough but at the same time not too large to be usable.
c. The use of critical paths in impact evaluation patterns was a good way in which fast prescreening could be performed prior to filling in the complete pattern. The implicit challenge here was that if a practitioner only fills in the (few) value components in the critical path, and others fill in the entire pattern, comparisons (as shown in Figures 8–11) would be harder because information was missing.
d. The Excel tool for software value analysis was highly appreciated with respect to usability and usefulness [61].

As SVM was introduced the complexity of the value concept also increased. This can for example be seen in the fact that some VCs can be an aggregate of two or more value components, for example, 'VC1.1.4: Complementary value' could be further categorized as 'VC: Complementary value for a product' and 'VC: Complementary value for a service'. This example is relevant when taking a decision regarding a product customization, given if the PC in question increases complementary value for another product or another service. In addition, as noticed by practitioners at Ericsson, the use of SVM and the patterns can be seen as increasing the overall effort put on the analysis of an artifact or phenomenon (e.g., a PC or requirement). Using a pattern might imply filling in 5–10 VCs. However, the explicit nature of the patterns, and the possibility to be explicit and use a shared vocabulary inherent in the use of the patterns, analysis time was saved overall.

7. CONCLUSION

The software value map presented in this paper contributes as a first step towards a complete categorization of value aspects, and the division into subvalue aspects and measurable value components. This was done utilizing knowledge from state-of-the-art in software engineering, business, management, and economics, gathered through extensive literature reviews, but also working with professionals in industry.

Moving away from a cost-based perspective, to a value-based perspective, utilizing both well-known aspects, such as customer satisfaction, but at the same time introducing more complex and complete aspects such as the customers' perception of value, is central. In themselves, most constituents of the value map are not new, but collected from different disciplines. The contribution of the software value map is the collection, categorization, and qualification of aspects adhering to the original intent, but also adding the concepts of interdependencies, attributes and measurement (where available), and the angling of value perspectives towards impact evaluation. In addition, the industrial evaluation of the value map was preceded by the identification of scenarios and subsequent creation of impact evaluation patterns. Impact evaluation patterns are purpose built incarnations of the complete value map, usable and useful for specific tasks. In addition to this, to further enhance usability and reduce time-to-decision, the concept of critical paths was introduced to enable triage of decisions. Thus, transforming a complete value map to a usable and useful tool was also of paramount importance.

A fundamental impact within Ericsson, attributed to the creation and use of the software value map, is the shift from cost-based discussions and reasoning, to value-based decision support. Even without the creation of patterns the value map gave a common vocabulary to the professionals, enabling them to be precise in their discussions of value, giving them the possibility to confirm exactly what they all mean by 'value'. The industry practitioners did not only verify the benefits of having a consolidated view of value components, relevant for a particular impact evaluation pattern, for decision-making; they also found the software value map a step towards common definitions and understanding of value components enabling effective communication. Additionally, they viewed the value map as a basis for customer value analysis, improving their understanding of what constitutes value for their

DOI: 10.1002/smr

customers, and for themselves. On the basis of such an analysis, explicit links between customer value perceptions to specific software attributes can be made — a very interesting avenue for future research. Last, but not the least, by recording the decisions taken by evaluating value components in the corresponding impact evaluation patterns, a retrospective analysis of the decisions for improving requirements selection quality was also seen as a major contribution by the practitioners.

From a research perspective, researchers can use the software value map as a collection of different value aspects and value components that need to be evaluated and compared for taking decisions about software management and development. This would enable devising measurement/evaluation solutions that do not just evaluate one aspect of value but rather evaluate all the different aspects spanning from customer perceived value, to internal business value and architectural impact and reusability. The idea is for researchers to use SVM as a starting point, where additions and refinements can be made, and any number of patterns can be created and evaluated in industry. The key is that if a pattern is incomplete, and the addition needed cannot be found in the SVM, then an addition can and should be made in the SVM.

'Completeness' was never the goal of the SVM, rather to create an as complete view on value as possible. The SVM should be used and extended, complementing value aspects and perspectives as needed, but maybe more importantly, we need to improve upon how these aspects should be measured in industry — a central avenue for future research in value-based software engineering.

APPENDIX A: Table A.I. Systematic map search strategy.

Data items	Values
Databases searched	Inspec and Compendex (through Engineering Village) and Scopus
Population	Software development process areas AND Decision-making
Intervention	Value related keywords
Outcomes	The outcome of systematic mapping study was the systematic classification and mapping of the research papers with reference to value constructs.
Search queries formulation	To make the search exhaustive, electronic databases were searched using the following strategy • → Boolean OR was used between the interventions. • → Boolean AND was used between the population and intervention. • → To restrict the search to research papers that contain only value constructs and the term 'software' Boolean AND was also used between them. Detailed queries are available on the weblink [40]
Out of scope	Fields such as enterprise resource planning, computer-aided software engineering, web services, etc., were out of the scope of this study. The rationale being the focus was on the value constructs used for taking decisions related to the software intensive product.
Reference management system	Endnote.
Year	The research papers are selected from the year 1969 to 2010.
Papers targeted	Research papers published in peer-reviewed journals, conferences, and workshops. Editorials, prefaces, summaries, news, reviews, correspondence, discussions, comments, reader's letters and summaries of tutorials, workshops, panels, and poster sessions were excluded.

Using the search strings [40] in Stage 1 (see Figure 12), a total of 670 papers were retrieved. In Stage 2 the duplicates were removed leaving 558 unduplicated papers. For each of the 558 papers, the source, the retrieval decision, retrieval status, and eligibility decision were recorded.

In Stage 3, the titles of all papers were screened to judge their relevance to the systematic mapping being performed. The papers whose titles were clearly not related to software engineering and value were excluded. In Stage 3, 417 papers were excluded leaving 141 papers in total. In Stage 4, 87 papers out of 141 were excluded primarily after reading the abstracts leaving 54 papers. The reason for excluding the 87 papers was absence of any value considerations for decision-making. However, it was found that abstracts were of variable quality; some abstracts were missing, poor, and/or misleading, and several gave little indication of what was in the full article. In particular, it was not always obvious whether a paper included any value construct. If it was unclear from the title, abstract,

DOI: 10.1002/smr

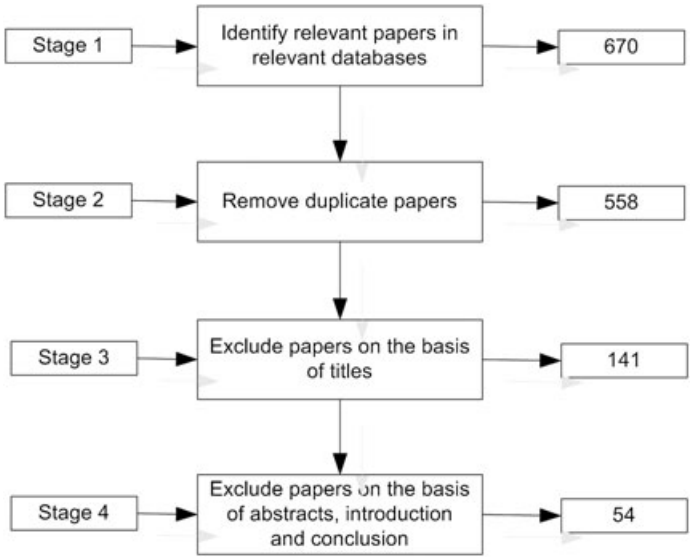

Figure 12. Stages of the selection strategy for the systematic mapping of SE literature.

conclusion, and keywords whether a research paper conformed to the screening criteria, it was included in the mapping, giving it the benefit of the doubt.

The inclusion and exclusion criteria were piloted by the authors on a random sample of 20 papers. The Fleiss' Kappa [41] value showed a reasonable agreement (0.67) among the authors.

APPENDIX B: Table A.II. VCs with corresponding rubrics for impact evaluation.

VC1.1.1.1.1: Functionality value

Impact		***Side effects***	
+2	critical	+2	Supports standard features
+1	Important	+1	Supports optional feature
0	No effect	0	No effect
-1	Wrongly specified	-1	Blocks/removes optional feature
-2	Completely misunderstood	-2	Blocks/removes standard feature

VC3.1.1: Market requirements value

Resell/standard product		***Effort required***	
+2	More than n customers would want it	+2	Minimal effort required to make it resell-able or add to standard product
+1	More than m customers would want it	+1	Some effort required to make it resell-able or add to standard product
0	No other customers wants it	0	N/A
-1	Customers will be hesitant to buy product with this feature	-1	Substantial effort required to make it resell-able or add to standard product
-2	Customers will not buy product with this feature	-2	Huge effort required to make it resell-able or add to standard product

VC3.1.2.1: Physical value wrt. cost

Cost	
+2	Minimal cost to implement this feature
+1	Some cost to implement this feature
0	N/A
-1	Substantial cost to implement this feature
-2	Huge cost to implement this feature

DOI: 10.1002/smr

VC3.1.2.3.3.1: Architectural functionality

+2	***Greatly improves the existing functional design of the architecture***
+1	Improves the functional design of the architecture
0	No effect
-1	Degrades the existing functional design of the architecture
-2	Seriously degrades the functional design of the architecture

VC3.1.2.3.3.3: Architectural Usability

Understandability		***Learnability***		***Operability***	
+2	Greatly improves understandability of the architecture	+2	Greatly improves learnability of the architecture	+2	Greatly improves operability of the architecture
+1	Improves understandability of the architecture	+1	Improves learnability of the architecture	+1	Improves operability of the architecture
0	No effect	0	No effect	0	No effect
-1	Degrades understandability of the architecture	-1	Degrades learnability of the architecture	-1	Degrades operability of the architecture
-2	Seriously degrades understandability of the architecture	-2	Seriously degrades learnability of the architecture	-2	Seriously degrades operability of the architecture

VC3.1.2.3.3.4: Architectural maintainability

Analysability		***Changeability***		***Stability***		***Tesability***	
+2	Greatly improves analysability of the architecture	+2	Greatly improves changeability of the architecture	+2	Greatly improves stability of the architecture	+2	Greatly improves testability of the architecture
+1	Improves analysability of the architecture	+1	Improves changeability of the architecture	+1	Improves stability of the architecture	+1	Improves testability of the architecture
0	No effect	0	No effect	0	No effect	0	No effect
-1	Degrades analysability of the architecture	-1	Degrades changeability of the architecture	-1	Degrades stability of the architecture	-1	Degrades testability of the architecture
-2	Seriously degrades analysability of the architecture	-2	Seriously degrades changeability of the architecture	-2	Seriously degrades stability of the architecture	-2	Seriously degrades testability of the architecture

VC3.1.2.3.3.5: Portability

+2	*Greatly improves portability attribute of the architecture*
+1	Improves portability attribute of the architecture
0	No effect
-1	Degrades portability attribute of the architecture
-2	Seriously degrades portability attribute of the architecture

ACKNOWLEDGEMENTS

We would like to thank our Ericsson steering group members and all the participants of the study who provided valuable input during the creation and validation of SVM and impact evaluation patterns. We would further like to extend our thanks to the reviewers from Ericsson for reviewing the paper and providing valuable feedback, which has helped in approving the paper. This work is part of the BESQ+ research project funded by the Knowledge Foundation (grant: 20100311) in Sweden.

REFERENCES

1. Biffl S, Aurum A, Boehm B, Erdogmus H, Grunbacher P. *Value-Based Software Engineering*. Springer: Germany, 2006.
2. Jeffery R. *Value-Based Software Engineering*. Springer: Germany, 2006.
3. Halling M, Biffl S, Grünbacher P. The Role of Valuation in Value-Based Software Engineering. *6th International Workshop on Economics-Driven Software Engineering Research - Proceedings of 26th International Conference on Software Engineering* Edinburgh, Scotland, 2004, 7–10.

DOI: 10.1002/smr

4. Barry B. Value-based software engineering: reinventing. *SIGSOFT Software Engineering Notes* 2003; **28**:3.
5. Gorschek T, Fricker S, Palm K, Kunsman S. A Lightweight Innovation Process for Software-Intensive Product Development. *Software, IEEE* 2010; **27**:37–45.
6. Aurum A, Wohlin C, Porter A. Aligning Software Project Decisions: A Case Study. *International Journal of Software Engineering and Knowledge Engineering* 2006; **16**:795–818.
7. Khurum M, Aslam K, Gorschek T. MERTS – A method for early requirements triage and selection utilizing product strategies. *Asian Pacific Software Engineering Conference (APSEC)* Nagoya Japan, 2007; 97–104.
8. Kontio J, Warsta J, Makela MM, Ahokas M, Tyrvainen P, Poyry P. Software business education for software engineers: Towards an integrated curriculum. *Proceedings of Software Engineering Education Conference*, Oahu, HI, United states, 2006.
9. Rönkkö M, Frühwirth C, Biffl S. Integrating Value and Utility Concepts into a Value Decomposition Model for Value-Based Software Engineering. *Product-Focused Software Process Improvement*. Lecture Notes in Business Information Processing: Springer, Berlin Heidelberg, 2009; **32**:362–374.
10. Sward D. *Measuring the Business Value of Information Technology. Practical Strategies for IT and Business Managers*. Intel Press: Santa Clara, 2006.
11. Lei Z, Ren S, Garcia FP, Zuzhao L. Multiple-value decision supporting application in software production facing global market. *IEEE International Conference on Systems, Man and Cybernetics*, Piscataway, NJ, USA, 2000; 346–51.
12. Song M, Parry ME, Kawakami T. Incorporating network externalities into the technology acceptance model. *Journal of Product Innovation Management* 2009; **26**:291–307.
13. Harmon RR, Laird G. Linking marketing strategy to customer value: implications for technology marketers. *Portland International Conference on Management and Technology*, Portland, OR, USA, 1997; 896–900.
14. Yongtae P, Gwangman P. A new method for technology valuation in monetary value: Procedure and application. *Technovation* 2004; **24**:387–394.
15. ISO. ISO/IEC 9126–1, ISO/IEC 9126–2, ISO/IEC 9126–3. ISO, 2001.
16. Kim C-K, Lee D-H, Ko I-Y, Baik J. A lightweight value-based software architecture evaluation. *Proceedings of Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 2007; 646–649.
17. Kazman R, Asundi J, Klein M. Making Architecture Design Decisions: An Economic Approach. Software Engineering Institute, Pittsburg, USA 2002.
18. Rappaport A. *Creating shareholder value: A guide for Managers and Investors* (2nd edn). The Free Press: New York, 1998.
19. Pasi O. Experiences of implementing a value-based approach to software process and product assessment. *Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications* Acapulco, Mexico: World Scientific and Engineering Academy and Society (WSEAS), 2008.
20. Boehm B, Sullivan K. Software economics: status and prospects. *Information and Software Technology* 1999; **41**:937–46.
21. Dakin K. Establishing a fair price for software. *IEEE Software* 1995; **12**:105–6.
22. Finkelstein L, Leaning MS. A review of the fundamental concepts of measurement. *Measurement* 1984; **2**:25–34.
23. Melton AC, Gustafson DA, Bieman JM, Baker AL. A mathematical perspective for software measures research. *Software Engineering Journal* 1990; **5**:246–54.
24. Fenton N *Software Metrics: A Rigorous Approach*. London: Chapman-Hall, 1991.
25. Fenton N. Software measurement: a necessary scientific basis. *IEEE Transactions on Software Engineering* 1994; **20**:199–206.
26. Bauer H, Hammerschmidt M, Braehler M. The Customer Lifetime Value Concept And Its Contribution To Corporate Valuation. EconWPA, 2004.
27. Hassenzahl M, Tractinsky N. User experiencea research agenda. *Behaviour and Information Technology* 2006; **25**:91–97.
28. Wohlin C, Aurum A (eds). *Engineering and managing software requirements*. (1st edn) Springer: New York, NY, 2005.
29. Potts C. Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. *Proceedings of the Second IEEE International Symposium on Requirements Engineering* Los Alamitos, 1995; 128–130.
30. Karlsson L, Dahlstedt Å, Natt och Dag J, Regnell B, Persson A. Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study. *Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02)*, Essen, Germany, 2003; 101–112.
31. Gorschek T, Wohlin C. Requirements Abstraction Model. *Requirements Engineering journal* 2006; **11**:79–101.
32. Khurum M, Aslam K, Gorschek T. MERTS – A method for early requirements triage and selection utilizing product strategies. *APSEC 2007* Nagoya Japan, 2007.
33. van de Weerd I, Brinkkemper S, Nieuwenhuis R, Versendaal J, Bijlsma L. Towards a reference framework for software product management. Los Alamitos, CA, USA, 2006; 312–15.
34. Seacord RC, Elm J, Goethert W, Lewis GA, Plakosh D, Robert J, Wrage L, Lindvall M. Measuring Software Sustainability. *Proceedings of the International Conference on Software Maintenance*: IEEE Computer Society, 2003.
35. Carlshamre P, Regnell B. Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes. *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*: IEEE Computer Society, 2000.

DOI: 10.1002/smr

36. Potts C. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. *Proceedings of the Second IEEE International Symposium on Requirements Engineering*: IEEE Computer Society, 1995.
37. Regnell B, Beremark P, Eklundh O. A market-driven requirements engineering process: Results from an industrial process improvement programme. *Requirements Engineering* 1998; **3**:121–129.
38. Petticrew M, Roberts H. *Systematic Reviews in the Social Sciences: A practical guide*. Blackwell Publishing: Oxford, 2006.
39. Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic Mapping Studies in Software Engineering. *12th International Conference on Evaluation and Assessment in Software Engineering (EASE)* University of Bari, Italy, 2008.
40. Khurum M, Gorschek T. The Software Value Map. http://www.bth.se/tek/aps/mkm.nsf/pages/softaware-value-map. 2011.
41. Fleiss J. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 1971; **76**:378–382.
42. Robson C *Real World Research*. Blackwell Publishing: Cornwall, 1993.
43. Greenhalgh T, Peacock R. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. *BMJ - Information in Practice* 2005.
44. Wohlin C, Runeson P, Höst M, Ohlson MC, Regnell B, Wesslén A. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic: Boston MA, 2000.
45. Wohlin C, Runeson P, Host M, Ohlson MC, Regnell B, Wesslen A. Experimentation in Software Engineering: An Introduction. (6th edn) International Series in Software Engineering, Kluwer Academic, 2000.
46. Kitchenham BA. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, 2007.
47. Dieste O, Grimán A, Juristo N. Developing Search Strategies for Detecting Relevant Experiments for Systematic Reviews. *Empirical Software Engineering* 2009; 513–539.
48. Brereton P, Kitchenham BA, Budgen D, Turner M, Khalil M. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software* 2007; **80**:571–83.
49. Aurum A, Wohlin C. A Value-Based Approach in Requirements Engineering: Explaining Some of the Fundamental Concepts. *13th International Working Conference on Requirements Engineering: Foundation for Software Quality*, Trondheim, Norway, 2007; 109–115.
50. Kaplan RS, Norton DP. The Balanced Scorecard—Measures that Drive Performance. *Harvard Business Review*, 1992.
51. Kaplan AS, Norton DP. *The balanced scorecard: translating strategy into action*. Business School Press: Boston, Harvard. United States of America, 1996.
52. Steven BD. Using the balanced scorecard process to compute the value of software applications. *Proceedings of the 28th international conference on Software engineering* Shanghai, ACM: China, 2006.
53. Liebowitz J, Suen CY. Developing knowledge management metrics for measuring intellectual capital. *Journal of Intellectual Capital* 2000; **1**:54 – 67.
54. Webster J, Watson RT. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly* 2002; **26**.
55. Gamma E, Helm R, Johnson R, Vlissides JM. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.
56. Pettersson F, Ivarsson M, Gorschek T, Ohman P. A practitioner's guide to light weight software process assessment and improvement planning. *Journal of Systems and Software* 2008; **81**:972–995.
57. Gorschek T, Davis AM. Requirements engineering: In search of the dependent variables. *Information and Software Technology* 2008; **50**:67–75.
58. Gorschek T, Garre P, Larsson S, Wohlin C. Industry evaluation of the Requirements Abstraction Model. *Requirements Engineering journal* 2007; **12**:163–190.
59. Simmons E. Requirements Triage: What Can We Learn from a "Medical" Approach? *IEEE Software* 2004; **21**:86–88.
60. Davis AM. The art of requirements triage. *IEEE Computer* 2003; **36**:42– 49.
61. Khurum M, Gorschek T. A systematic review of domain analysis solutions for software product lines. *Journal of Systems and Software* 2009; **82**:1982–2003.

AUTHORS' BIOGRAPHIES:

Dr. Mahvish Khurum is a post-doctoral researcher in Software Engineering at the School of Computing at Blekinge Institute of Technology in Sweden. She received her PhD degree of in Strategic Software Engineering in 2011. Dr. Khurum's research area is Strategic Software Engineering, which touches both software engineering and industrial technology and management. She is also working with practical innovation.

Dr. Khurum bases her research on challenges identified in industry, then develops solutions in collaboration with industry practitioners, and ultimately validates and tests the solutions in a real industrial setting. Some of the current Industrial partners include IBM, Qtema and Ericsson. She is also a member of Software Product Management Organization (http://ispma.org/personalmembers/mahvish-khurum/)

DOI: 10.1002/smr

Dr. Tony Gorschek a Professor of Software Engineering at Blekinge Institute of Technology (BTH). He has over ten years industrial experience as a senior executive consultant and engineer, but also as chief architect and product manager. Currently he manages his own consultancy company, works as a CTO, and serves on several boards in companies developing cutting edge technology and products. His research interests include requirements engineering, technology and product management, process assessment and improvement, quality assurance, and practical innovation. Dr. Gorschek bases his research on challenges identified in industry, then develops solutions in collaboration with industry practitioners, and ultimately validates and tests the solutions in a real industrial setting. Industrial partners include, but is not limited to IBM, Qtema, Ericsson, Daimler AG, and Volvo Cars. For more information/publications/contact: www.gorschek.com

Magnus Wilson is a Program manager at Ericsson AB. He works with tool development and methodologies including training programs) for 1) Business models (Osterwalder + BMUM Business decision flow) including Value analysis and Requirement management, 2) Enterprise Architecture, and Service Oriented Architecture. All with a domain focus on Business Support Systems (BSS within the ICT industry). In addition he is also the representative from Ericsson for joint research conducted together with academia within Innovation, Value, and Business modeling.

DOI: 10.1002/smr