Cross-Organizational Software Development: Design and Evaluation of a Decision Support System for Software Component Outsourcing

Tommi Kramer
University of Mannheim
kramer@uni-mannheim.de

Armin Heinzl
University of Mannheim
heinzl@uni-mannheim.de

Tillmann Neben
University of Mannheim
neben@uni-mannheim.de

Abstract

While the decision to outsource software development tasks was mainly considered strategically and economically, it relies on technical properties of single components and their integrability into complex systems, as well. This paper suggests a decision model that evaluates technical properties of software components to support the outsourcing decision with its implications on the cross-organizational distribution of development tasks. Following a design science approach decision criteria are deduced and logically combined in order to design a decision model. The model is then used to implement a mobile prototype for a decision support system in order to classify all software components regarding their outsourcing applicability. Both model and tool are evaluated in depth: we examine the quality of model and tool in a naturalistic and experimental evaluation setting. The overall satisfaction with utility, ease of use and intention to use is very positive.

1. Introduction

IT Outsourcing (ITO) has become a common phenomenon within software development projects. It leverages competitive hurdles such as cost pressures and the so-called war for talents. However, outsourcing decisions are often made in an unstructured, at best heuristic manner. Thus, coordination of development tasks in cross-organizational settings is uncontrolled. This research supports outsourcing managers in identifying software components that can be outsourced and, thus, developed cross-organizational with the help of a decision support systems (DSS). As this paper also comprises the development of such a system, we follow a design science research approach [1]. Theoretical insights into the interfaces between decision support, outsourcing and component based software development, as well as the technological contribution in form of a decision support software to perform such decisions allow for a better structuring of outsourcing decisions.

For more than 20 years different outsourcing models have evolved within the software industry; especially the rising demand for cloud services has increased and diversified the software outsourcing market [2]. The multitude of possible solutions made outsourcing decisions even more complex. Additionally, interoperability of different outsourced services or resources has to be considered after the outsourcing decision has been made.

Literature about software outsourcing so far has rather focused on the strategic fit for the buying firm and less considered technical impacts [3]. In a software world of modularity, though, composing different technical components to build an integrated system as a whole may predominantly have strategic impact and improve cross-organizational collaboration by assigning tasks to the right organization.

Current research in outsourcing decision-making emphasizes on appropriate supplier selection, location selection, or considers communication and coordination aspects for the assignment of development tasks [4, 5]. Thus, applicable approaches mainly comprise best practices and guidelines for relationship management [5, 6]. However, the advantages of a component based software product and its development process have not yet been examined and evaluated together in an outsourcing scenario. On the one hand, modularity in a collaborative setting enables cost efficiencies by flexible staffing of working capacities and increased production speed [7, 8]. On the other hand, cultural differences and a lack of exchange of data and information may hinder an efficient and successful development process in outsourcing scenarios [9, 3].

Considering these aspects, the existing body of knowledge concerning cross-organizational collaboration so far helps to answer the *when* and the *how* but not the *what* of cross-organizational outsourcing. Although a software product is the focus of software development, existing outsourcing studies have widely neglected product characteristics to be considered

URI: http://hdl.handle.net/10125/41190
ISBN: 978-0-9981331-0-2

when deciding about outsourcing. Therefore, in our paper we specifically investigate the following research question: *Which components of a software system can be outsourced to a vendor during development?*

Accordingly, we intend to make a contribution for theory and practice by developing and providing a normative decision-making approach for the development of software components in cross-organizational collaboration settings. We apply design science approaches in doing so [1]. Thus, the remainder of our paper structures as follows: At first, we develop and provide a decision support model for software components and expand existing outsourcing knowledge in doing so. Second, we design and develop a software prototype instancing the previously defined decision support model in order to support practitioners facing such outsourcing challenges. Finally, a rigorous evaluation reveals the usefulness of the decision model and the ease of use of its implementation as well as an intention to use the researched solution within collaborative software development projects. The paper concludes with a summary and limitations.

2. Outsourcing decision support model

The first part of our contribution represents a decision support model that contains several characteristics of a software component, which have an impact on the sourcing decision. A differentiation of these characteristics is made between the structure of a component, the procedural influence on the development process, and required knowledge specifics. According to the design science research approach [1] we deduce our characteristics from established information systems (IS) theories. Thus, we determine the design for a software tool which instances the decision support model developed in this section.

To get there, at first the outsourcing decision has to be specified. Then, the deduction of appropriate characteristics and their categories is described. Furthermore, a decision support model for classifying each component of a software product is developed out of the given characteristics. Adjacent decision logic is also part of this section and enables for repeatable classifications of components with equivalent results.

2.1. Decision to be supported

In line with our research question we pursue the goal to answer the question: What shall be outsourced? Hence, the object of investigation is the decision which software component qualifies for internal and which one for external development. In this context we take a holistic perspective on each software component in terms of a cohesive and discrete logical unit - the atomic entity of an entire system [10]. Assumptions and decisions about the structure of these entities can be made after the design phase within the development process. Consequently, an according decision model can be established with a target function to provide guidance (develop the selected component internally, externally or either way) using decision criteria (structure of a component, development process specifics or knowledge specifics) in order to evaluate the decision field (optimal cross-organizational task allocation within the development process). Therefore, we determine the following design requirements:

For the decision criteria search process we distinguish between three distinct groups: These criteria can either address *structural*, *procedural* or *knowledge- based* attributes of a software component. We enrich our search for criteria with several established theories used in IS research. By that, we assess the organizational impact of component-based outsourcing and the structure of the decision model.

Transaction cost economics: Limited rationality and opportunistic behavior foster costs occurring for every transaction. Within outsourcing they are the only costs besides production. For the decision to outsource components, hence, the transaction costs of outsourcing have a major impact on the sourcing decision [3]. As a consequence, components that entail high transaction costs have to be identified and kept for in-house development.

Resource-based view: Markets are supposed to learn quickly, inducing the need for single corporations to use explicitly the resources from the market complementing their profile [11]. Without adequate resources, outsourc- ing becomes necessary. Resources constituting a com- petitive advantage, however, must not be outsourced. Therefore, the decision support model must contain criteria that help to identify components that comprise critical resources and prevent them from draining off.

System theory as complex system are composed from single interacting fragments, the developer has to secure effective interaction [12]. This can be achieved by minimal inter-component dependency, but strong intra-component dependency and well defined interfaces [13]. To this end, central components must be kept in-house whereas loosely coupled components are rather suitable for outsourcing.

From the theoretical definition of DSS we draw additional design requirements: Selecting and defining software component attributes must be *flexible*

and editable [14]. Also, all weights used within the decision logic have to be individually adjustable [15].

2.2. Decision model

According to the previously defined design requirements we propose an initial decision model in this section as presented in figure 1. In the following, the structure of our decision model is explained in detail. The essential decision logic for our model is defined and illustrated subsequently.

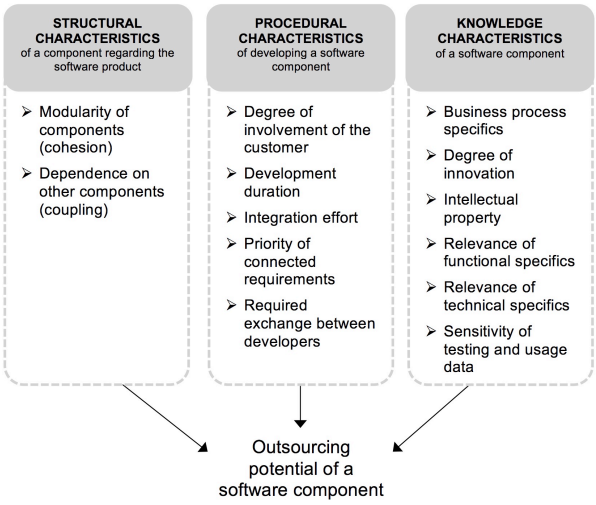

Figure 1: Decision model

At first, structural attributes of a component in relation to the software product can be well explained through systems theory, which recommends a modular setup for complex systems [12]. To do so, the aggregation of functions into components must be exhaustive, while mostly independent from each other. In the model we cover that by cohesion (component modularity) and coupling (component interdependence). A component can be called modular if it can execute the necessary functions itself at most. High modularity suits outsourcing, as it enables handing over strictly defined items and decreases the need for coordination. The degree of interconnectedness defines the coupling of components [13]. Coupling can be measured by evaluating the defined relations of components within requirements. Highly coupled components may act as communication interfaces and are crucial to the functionality of the whole system. Therefore they should not be outsourced.

Second, we focus on procedural attributes of the development process for software components while applying the lens of transaction cost economics. Those processes that induce high transfer costs (such as direct communication, personal meeting, etc.) decrease the outsourcing potential due to higher development duration, integration effort etc. as shown in the decision model above.

Components with a high development period are likely to be on a critical path. If the critical path is violated externally, high transaction costs may arise. Further, priorities connected to requirements indicate the attention of interest groups. The failure of high priority components for this reason may lead to more (communication) transactions. More specifically with a higher degree of customer interaction, higher communication effort will follow if problems occur. In the interest of customer satisfaction, as well, components with high customer involvement do not suit outsourcing. As another procedural aspect, if developers must interact intensively, e. g. because of high coupling, such a component would also not qualify for outsourcing. This accounts especially within distributed software development. At last if integration needs high effort, the component should be developed internally as well, as a continuous integration is important within the development process.

Third, knowledge-based attributes can be embedded by applying the resource-based view. Central to this perspective is the term specificity that may be described by characteristics that embody an advantage of specialization compared to the market [16]. A business process specificity can be identified where software components match business processes more specifically than standard software on the market [17]. Functional specificity expresses in how far a software component can fulfill the specific needs of a business function [18]. At last, technical specificity is used to determine the degree of integrating highly specific technology, such as core banking software [17]. Highly specific components are difficult to imitate and may hold a competitive advantage. As a result they should be kept in-house. Out of a knowledge-based perspective additionally the novelty of the application should be considered as well as the sensitivity of test and user data.

Having defined all structural, procedural and knowledge-based characteristics, the decision criteria of the decision model are now completely derived from aforementioned theoretical approaches. They now have to be applied by decision logic in order to receive reliable outsourcing results.

A target function is required for repeatable and valid outsourcing decisions regarding every single component of the software product. In order to achieve such a function our decision logic comprises three discrete evaluation potentials (high / medium / low) for each of the decision criteria (e.g. integration

effort). These potentials are the only ones that outsourcing managers have to adjust when applying our decision model. Optionally, they also may shift default weightings ad libitum but are not required to.

For defining the target function, decision support systems literature differentiates between evaluating action alternatives due to given criteria (multi attribute decision-making - MADM) or calculating a preferred alternative out of non-distinct solution sets (multi objective decision-making - MODM) [19]. Evaluating the outsourcing potential of each related component of a software product when the potential can be either high or medium or low results in an almost uncountable amount of different solutions. Therefore, MODM approaches do not fit the required needs for our decision logic. Instead many attributes must be accounted for and that is why we pick decision matrices from MADM as an appropriate mean to apply in our decision logic. They offer possibilities to enhance attributes, enable a multi-stage decision process, traceability, lucidity and re-usability of the initial configuration [20, 21]. Thus, it has been selected to enrich the goal function of the decision model. Within the decision table, all conditions are listed in the upper rows, while all alternative actions are listed below. Rules for decision-making are embedded via columns that connect all combinations of conditions. Hence, single cells define concrete conditions for a rule that deduces a specific guidance below [22].

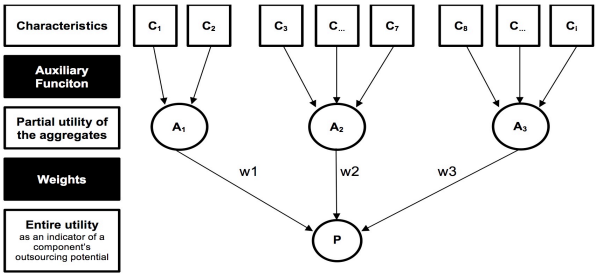

Figure 2: Decision logic

Having subdivided the decision criteria of the model into three categories, the groups must be weighted, resulting in a preference function for individual decision makers. Such a weighting function can be implemented using a priority list that indicates the individual use of the criteria groups [23]. Priorities can be addressed on an ordinal scale and will then be normalized to a weight between 0 and 1. This easy process makes prioritizing more transparent for the decision maker [24]. Finally, the amalgamation of all weighted criteria into one synthetic value expresses the total utility value of the outsourcing alternative. That value is returned from the target function. The logic of the model including the decision criteria as characteristics, aggregated values and the final utility value is illustrated in figure 2.

2.3. Applying the decision model

This decision model and its amalgamating calculation can be used to compare components to outsource through a single score. To do so, an adapted utility value analysis will be presented to determine a comparable value for each component. A criterion can be rated high, medium or low indicating how distinctively a criterion is fulfilled by the considered component. For each different rule the criteria can be rated respectively and summarized. This may serve to support the outsourcing decision. To apply this strategy on decision tables, all rules must be specified in before. The ordinal scale can easily be transferred to a cardinal one in order to find a median for each criteria category. During this process the specific influence of each category on the outsourcing decision can be calculated. Only when this is done, the weighted criteria values can be amalgamated. One exception has to be made for the structural criterion cohesion. Due to its inverse function, it must be corrected by subtracting the value for cohesion from 4. However, this can be neglected since the values for cohesion and coupling can automatically be calculated by a clustering algorithm.

The total utility value can, hence, be calculated as following: The single utility value of each decision category will be calculated by the average of its respective characteristics. These categorical values will then be multiplied by the normalized weight. As the weight is normalized from 0 to 1 and the single assessments of the characteristics can be valued from 1 to 3, the amalgamated final outsourcing recommendation value lies within 1 to 3 as well, with values from 1,00 - 1,66 indicating low specificity and hence outsourcing, while values from 2,34 - 3,00 indicate low outsourcing potential due to high specificity and neutral otherwise.

3. Implementation of the model

The second part of our design-oriented research contains the instantiation of the previously developed decision model and its logic. In order to receive a viable prototype that is useful for a rigorous evaluation, we set our focus on technical feasibility and ease of integration. Thus, the resulting tool is intended to be easily applied in real cross-organizational collaboration scenarios.

3.1. Architecture

The architecture of the decision technology is based on three layers (cf. figure 3). The first layer (from bottom to top) is a mobile application used to perform the decision support process. It is the main application of the suggested solution and is called *SmartSourcer*. In order to easily distribute our prototype in the evaluation phase to the stakeholders of the DSS we have decided for a mobile application.

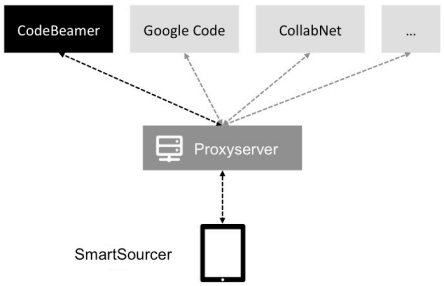

Figure 3: Architecture of the DSS

The second layer of the architecture serves as communication layer between the mobile app and different collaboration platforms that could be connected to our DSS. The so called proxy server translates the different field names of known collaboration platforms to a unique and identifiable field for *SmartSourcer*. Thus, the proxy is not only used to read from platforms but also to write back resulting information from the mobile app.

The third and last layer represents a set of existing collaboration platforms for software development. We include existing tools at this place in our solution since software companies that engage in cross-organizational collaboration heavily rely on them. They provide the means for development process support in collaborative scenarios like outsourcing. As a result, we can make use of already pre-defined software components that are stored in these platforms.

As indicated by the grey box in the upper right corner of figure 3 the given architecture can simply be extended by further collaboration platforms. The proxy server will then take over the task of correctly translate corresponding fields. Thereby, we support our claim for ease of integration.

3.2. Implementation details

For replication purposes of our research we provide implementation details for the aforementioned architecture. At first, the mobile application (*SmartSourcer*) is based on the iOS operating system and is implemented for tablet usage. The programming language used for coding is ObjectiveC and delivers a native iOS application. Thereby, optimal use of the operating system's hard- and software components are guaranteed. The integrated development environment (IDE) *XCode* helped extensively to generate graphical user interfaces, corresponding storyboards, clearly arranged screens and especially with data modeling and handling. The storage concept of the application provides meta-data as well as decision criteria of the decision model to be saved within the local storage. For versioning the code, GitHub as a free online versioning tool was chosen. Therefore, the source code is freely available with the authors. Additionally, a model-view-controller pattern is used for implementation.

For the implementation of the proxy server *Eclipse for Java EE Developers* was used as IDE. The compiled code is deployed on an Apache Tomcat server. This server guarantees a bidirectional mapping of corresponding data fields for details about planned software components. It also exchanges management reports and final assessment data between the mobile application and the collaboration platform.

The third layer is only an abstraction and can be seen as the connection (interface) to existing collaboration platforms used in distributed software development. In our case, we used the comprehensive CodeBeamer platform for establishing a connection of the proxy server to a collaboration platform. CodeBeamer operates as a Java webserver instance on a self-hosted server. The platform offers support for almost all phases of the software development process and provides communication and collaboration means for stakeholders of a software project. Therefore, we were able to test our DSS on all three layers of its architecture.

The communication of the proxy server to CodeBeamer is realized via a Web-API on the proxy server. Thus, CodeBeamer information can be read and manipulated via the Web-API that is built in Java as well as the CodeBeamer server. Web services of the proxy server enable our mobile app to exchange data in form of the JSON format with the proxy. The JSON format was chosen, as it enables a lightweight exchange without defining variables within the data sent. Thereby, the reduced amount of data sent reflects appropriate characteristics for mobile applications.

In order to setup an equal DSS environment as we use in our research, the following must be guaranteed: a collaboration platform (as indicated in the architecture section) must be in place in order to be

able to define the requirements, an architecture, the design, and corresponding components for a software product to be developed. Additionally, for a quick installation of the mobile application, iPads are required. Apps for different devices had to be prepared otherwise.

3.3. User interface

SmartSourcer as the implementation of the decision model is at the core of this paper's contribution. It facilitates the assessment of a software components outsourcing potential for the person in charge and can easily be used wherever the iPad is accessible. SmartSourcers main functionality can be described by three main functions:

3.3.1. Project selection. After starting the app, a user can select a project for which its components can be evaluated regarding their outsourcing potential. Alternatively, previously evaluated components can be retrieved from system storage. Personal login data and decision model configurations can be specified within the settings.

3.3.2. Evaluation. In a second step the user can evaluate all components drawn from a CodeBeamer repository. In this step, a label (high / medium / low) is assigned to each decision criteria of the decision model (cf. figure 1). Meta information for each component is pulled from the collaboration platform. Additionally, the categories of the decision model must be weighted (cf. figure 4). In case the user has no preferences, the weights are distributed equally among all categories. The weights are normalized within the process of choosing weights.

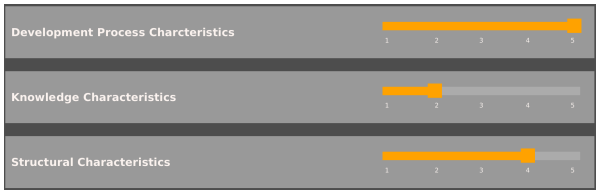

Figure 4: Selection of the weightings

3.3.3. Results. When weights are provided and each criterion is evaluated, the user can continue to a result screen. The calculation takes place with respect to the specified values as explained in the decision model before. An overall outsourcing recommendation is given, as well as advice for each component of the system. As indicated in figure 5, users can look up a detailed result view that makes each calculation transparent. Results can further be exported as a PDF for sharing, e.g. via mail. Additionally the results can be processed back to CodeBeamer and there be used for making decision within this collaboration platform transparent.

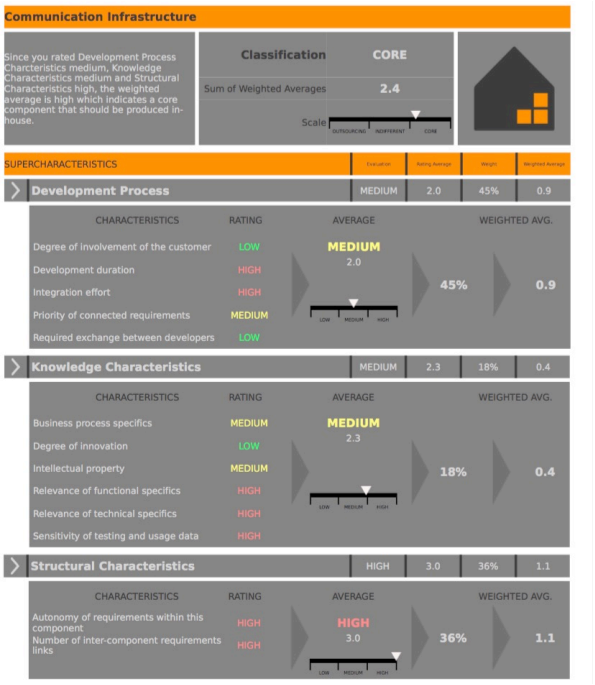

Figure 5: Detail screen on SmartSourcer

3.4. Calculations of results

Decision-making within the mobile application are aligned to the developed decision model and can be broken down in three central substeps. Finally, an outsourcing decision for each software component is suggested by the system.

3.4.1. Value assignment to decision criteria. Before any calculation can be executed by the decision support system, the manual assignment of the labels *high*, *medium* and *low* to each decision criterion of a component has to be translated into a utilizable value. In line with the decision logic of our model (cf. figure 2) only then a reliable and repeatable calculation can be realized. According to table I, the value assignment to each criterion of a component is conducted.

Table I: Value assignment of labels

High	3
Medium	2
Low	1

3.4.2. Calculation of aggregated values. According to the decision logic as introduced in figure 2, the aggregated values A_1, A_2 and A_3 represent partial utility values for the decision categories (structural / procedural / knowledge characteristics). These values are calculated by the arithmetic mean of all decision criteria of a category. The result is a value between 1 and 3. In order to provide clearly understandable intermediate results that are transparent for the users and to comply with the decision logic, we transfer the arithmetic results back to previously used labels.

In table II the corresponding intervals are transferred into labels. This is coherent with the usage of decision tables as indicated in the decision logic of our model in section 2.2.

Table II: Labels and intervals

1.00 – 1.66	Low (in-house)
1.67 – 2.33	Medium (indifferent)
2.34 – 3.00	High (outsourcing)

3.4.3. Amalgamation. The final calculation step is required to amalgamate the previously calculated partial utility of the aggregates. Thus, a decision recommendation can be derived for the sourcing of each software component. The calculation of the outsourcing potential P follows the formula

$$P = w_1 * A_1 + w_2 * A_2 + w_3 * A_3$$

and results again in a value between 1 and 3, as the weights are normalized. By translating the resulting value with the help of table II we immediately receive decision support for the software component under investigation.

A high outsourcing potential recommends to have a supplier developed the respective component. A low potential indicates that the component should be developed in-house. Medium values do not have the power to provide a clear decision support. A detailed screen of all resulting values and recommendations given by the system is presented in figure 5.

4. Evaluation

As an integral part of the design science approach, the artifact must be evaluated [1]. The advantages of the technology developed must rigorously made visible for both the theoretical knowledge base, as well as for practical operation. The tool under investigation is the decision support system, including the decision model and *SmartSourcer*.

4.1. Importance of the evaluation

In order to demonstrate its proposed usefulness, the tool designed has to prove practical relevance and formalized knowledge gained [25]. According to [1], the evaluation is an essential part of the design science paradigm and systematically gives proof of usefulness and quality of the artifact. [26] suggests besides the common artificial evaluation a naturalistic one taking place in the environment of the user and thereby including interference not visible in a laboratory setting.

[27] refine this approach and develop a systematic distinction between artificial or naturalistic, ex ante or ex post and process vs. product evaluations. Other parameters are the current development phase, goals and requirements for research as well as costs, resources and time restrictions.

[25] at last provides the most comprehensive evaluation guidelines within the design science approach. It classifies the artifact due to its context properties and suggests an appropriate evaluation strategy based on the aforementioned options.

4.2. Evaluation methodology

We subdivide the evaluation of our DSS in a quantitative and a qualitative part. For the quantitative section we make use of essential constructs of the technology acceptance model (TAM). Our qualitative analysis is based on criteria for measuring quality within the Soft Systems Methodology introduced by [28] and extended by [27, 1, 29]: effectiveness, efficiency, ethnic aspects and elegance. However, collection feedback about the utility and the quality of our artifact is in the scope of this evaluation.

4.2.1. Quantitative evaluation. Perceived usefulness and perceived ease of use are central constructs of the technology acceptance model (TAM), which suits ex ante as well as ex post analyses over different time periods [30, 31]. In order to rigorously evaluate our decision model and its implementation within SmartSourcer, we derive our evaluation model as presented in figure 6 from TAM. Therefore, it contains TAM's central constructs and fits well to our evaluation strategy.

Within this context, perceived usefulness can here be seen as the degree to which a person expects increased quality from the structured outsourcing decision within SmartSourcer. Perceived ease of use may here be the degree to which using SmartSourcer is possible without additional effort and can be measured using established items. The same holds for the intention to use of SmartSourcer, which is a good indicator for the actual use of our system.

To measure these constructs, measurement parameters have been specified. Perceived usefulness is derived through the quality of the decision, broken down into information quality and quality of the logic, and the perceived improvement through higher information quality. Perceived ease of use is defined as the quality of implementation and the perceived ease of using the implemented decision model. The intention to use can simply be measured directly by asking the user.

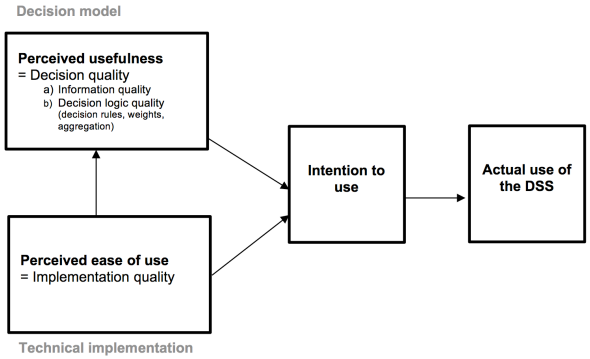

Figure 6: Evaluation model

4.2.2. Qualitative evaluation. The qualitative part of our study focuses on the implementation of our DSS. Thereby, we gain precious feedback of real users during the development process of our application in order to inform the release cycles of our prototype. On the one hand we further improve our prototype in doing so. On the other hand we also collect feedback for further development after our study.

We apply the framework of [32] for measuring and comparing information systems as it summarizes several evaluation criteria for such systems that analyzes their utility and usability [29]. The framework is subdivided into organizational, individual, information related, technology related, and systemic aspects. Thus, it serves as a basis for the questionnaire we use in our study.

4.2.3. Evaluation strategy. According to the quantitative and qualitative setup of our evaluation we decide to create a questionnaire covering the requirements from both types and addressing participants with outsourcing experience. In the quantitative part we ask the user in total for 29 items. These items cover the constructs that we discussed in our evaluation model. The questions are generally adapted from existing literature about TAM. All these criteria are measured on a Likert scale form 1 to 5.

The qualitative section of the evaluation questionnaire contains amongst personal data like age, experience with outsourcing or employment with the company additional 19 questions covering the categories of the applied evaluation framework.

For the data collection we applied an ex ante and ex post evaluation approach. At first, an artificial ex ante evaluation of our DSS is conducted. Its goal is to collect information for the improvement of the prototype in additional release cycles of the development. In total, we received 15 answered questionnaires from master students and academics with majors in business economics, business administration and information systems. They were invited to first receive an overview of SmartSourcer and the underlying DSS. A CodeBeamer project had been prepared and served as example case for the participants to go through the entire outsourcing decision process. Afterwards the DSS was assessed via the previously defined questionnaire.

Second, our DSS is evaluated in a naturalistic ex post scenario. On this occasion, branch experts within their regular environment guarantee a rigorous evaluation. Two medium sized software companies with proven experience in outsourcing supported our study. This naturalistic ex post evaluation delivered additional 15 results of outsourcing experts with experience between 1 and 13 years with an average of 6.5 years. The employees were confronted with decisions about outsourcing software components in their daily business. After presenting SmartSourcer and introducing its functionality (decision model and logic), a fictitious outsourcing project had to be conducted. At the end, all of them evaluated our DSS within the given questionnaire.

4.3. Quantitative results

In the quantitative part of our evaluation we use descriptive statistical methods to assess the constructs of the evaluation model (cf. figure 6). For analyzing the questionnaire results we use IBM SPSS software. Although we have two different evaluation groups (ex ante and ex post), the result of the t-test indicates that both groups do not significantly differ from each other. For that reason, we give a short overview of the descriptive analysis in which we hint at minor differences between the groups before we reveal the correlation of the constructs. In doing so we combine the results of both groups as claimed by the t-test.

Perceived ease of use (EASE) was the best-rated construct. It expresses the quality of implementation with a 4.49 as good to very good, with hardly any difference amongst both user groups. Especially the low complexity and stability were very positive. While the experts ranked stability higher, students expressed the application was easier to understand.

The familiar and safe use of the system was rated relatively low – however the system was indeed novel to the participants. Regarding style elements experts were pickier than the students.

The *perceived usefulness* (USEFUL) was rated good amongst both groups (4.03 and 4.01). However, the experts perceived the quality of the DSS slightly better, which is interesting due to their daily business. Regarding possibilities to interact with coworkers was as an outlier rejected with 2.65.

The *perceived intention to use* (INTUSE) was rated with a solid 4.27. The total results between both groups are marginal and do not prove different populations beyond. However, for N = 30 the t-test provides restricted validity only. A correlation matrix (cf. table III) further shows that perceived usefulness and perceived intention to use significantly correlate. The same holds true for perceived ease of use and intention to use. It can be stated that there is a high intention to use SmartSourcer and that this effect is increased due to the interaction of the constructs. With high quality of implementation and perceived usefulness, it is not surprising that the intention to use is high, as well.

Table III: Correlation matrix

	α	μ	σ	min	max	1	2	3
1 INTUSE	-	4.27	0.64	3.00	5.00	1.00		
2 USEFUL	0.78	4.06	0.37	3.00	5.00	0.52**	1.00	
3 EASE	0.62	4.51	0.31	3.00	5.00	0.4*	0.43*	1.00

N = 30, * $p < 0.05$, ** $p < 0.01$

4.4. Qualitative results

The answers to the open questions of our questionnaire provide feedback regarding the prototype's implementation. The analysis reveals some weaknesses of the user interface that have been improved in further release cycles of the mobile application.

For instance, many student respondents criticized that the additional multi-branched start menu were confusing and did not add value. It was therefore reduced to a simple one-page menu. Additionally, a more detailed listing of the calculation was favored. Regarding information flow, critic rose that the connection to other systems was not visible enough – settings were changed accordingly. Further, a reset function was missed and implemented afterwards. At last the portrait mode was said to be not utilized which was then also implemented afterwards.

The experts mainly provided organizational feedback. At first, it was stated that coworkers new to outsourcing decision-making could learn from such an application. Second, the decision model could be used as a company-wide standard if success is proven – making it necessary to save and share settings. The tablet implementation was mentioned positively, as well. One participant mentioned that the tool could be used to justify outsourcing decisions. Some mentioned that known graphic elements would make navigation easier. Time-driven budget functionality was demanded, as well. Information related aspect of the interviews were at most self-reflective for the candidates. Technologically the tool experienced positive feedback, a PDF- export function would be desirable. The integration into existing collaboration software was mentioned positively as well.

5. Summary

In line with the research question stated in the introduction section, the research contribution of this design science paper is the design, implementation and evaluation of an innovative artifact. The major contribution of this paper is the novelty of including technical characteristics of a software product for ITO decision-making. To the authors' knowledge, this is the first approach to consider technical characteristics of software components in order to facilitate a component based outsourcing decision. It supports decision makers whether to outsource a component or not and, thus, makes gut decisions superfluous.

The second contribution is the development of a normative decision model to conduct outsourcing decisions within software development teams. Normative models for outsourcing information systems have been rare; hence, this paper complements the knowledge base. The development of a holistic decision model based on established IS theories was achieved, as well.

Third, the model has proven utility and usability for both the decision model and the prototype. A correlation between usability and intention to use was proven to be significant for SmartSourcer.

5.1. Limitations and perspective

The praxeological-conceptual deduction of the decision model from theoretical concepts is certainly subject to criticism. Instead of rigorously deducing criteria, those that were suitable for practical application were chosen and included. Since we follow a design science approach there is no request for theoretical deduction but rather for creative and unprecedented design aspects. Nevertheless, further research could include a more stringent theory deduction part.

In our evaluation, the number of participants limits the explanatory power of the significant relations between the constructs of our evaluation model. A longitudinal setup with additional companies participating could further validate the insights gained in this study.

Despite its limitations this paper provides innovative artifacts to support the decision to outsource software components. It consists of both a normative decision model and the regarding decision logic to evaluate the individual components. The theoretically deduced and implemented decision criteria embody a new concept that includes technical characteristics into the decision. This approach was implemented via SmartSourcer in a successful manner according to the evaluation. It confirms the quality of both the decision model as well as the implementation and thereby the utility and usability of SmartSourcer to decide about outsourcing software components.

6. References

[1] A. R. Hevner, S. T. March, J. Park, and R. Sudha, "Design science in information systems research," MISQ, vol. 28, no. 1, pp. 75–105, 2004.

[2] M. Böhm, S. Leimeister, C. Riedl, and H. Krcmar, "Cloud Computing: Outsourcing 2.0 oder ein neues Geschäftsmodell zur Bereitstellung von IT-Ressourcen?" Inform. Mgmt. and Consult., vol. 24, no. 2, pp. 6–14, 2009.

[3] J. Dibbern, J. Winkler, and A. Heinzl, "Explaining variations in client extra costs between software projects offshored to India," MISQ, vol. 32, no. 2, pp. 1–30, 2008.

[4] V. Grover, M. Cheon, and J. Teng, "The effect of service quality and partnership on the outsourcing of information systems functions," Journal of Management Information Systems, vol. 12, no. 4, pp. 89–116, 1996.

[5] M. C. Lacity, S. A. Khan, A. Yan, and L. P. Willcocks, "A review of the it outsourcing empirical literature and future research directions," Journal of Information Technology, vol. 25, pp. 395–433, 2010.

[6] J. Lee, M. Q. Huynh, K. R., and S. Pi, "IT outsourcing evolution – past, present, and future," Communications of the ACM, vol. 46, no. 5, pp. 84–89, 2003.

[7] U. Apte, "Global outsourcing of information systems and processing services," The Information Society, vol. 7, pp. 287–303, 1990.

[8] E. Carmel and R. Agarwal, "Tactical approaches for alleviating distance in global software development," IEEE Software, vol. 18, no. 2, pp. 22–29, 2001.

[9] E. Carmel and P. Tjia, Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce. Cambridge, USA: Cambridge Univ. Press, 2006.

[10] J. Cheesman and J. Daniels, UML Components – A Simple Process for Specifying Component-Based Software. Addison-Wesley, 2001.

[11] R. M. Grant, "The resource-based theory of competitive advantage: Implications for strategy formulation," Calif. Mgmt. Review, vol. 33, no. 3, pp. 114–135, 1991.

[12] H. A. Simon, "The architecture of complexity," in Proceedings of the American Philosophical Society, vol. 106, 1962, pp. 467–482.

[13] C. Y. Baldwin and K. B. Clark, Design Rules: The Power of Modularity. Cambridge, USA: MIT Press, 2000.

[14] R. Grünig and R. Kühn, Successful Decision-Making. Springer, 2005.

[15] K. Manz, A. Dahmen, and L. Hoffmann, Entscheidungstheorie. München: Vahlen, 1993.

[16] O. E. Williamson, The Economic Institutions of Capitalism. New York, USA: Free Press, 1985.

[17] L. M. Applegate, F. W. McFarlan, and R. D. Austin, Corporate Information Strategy and Management: Text and Cases. New York: McGraw-Hill, 2003.

[18] J. Winkler, J. Dibbern, and A. Heinzl, "The impact of software product and service characteristics on international distribution arrangements for software solutions," in Proceedings of the 30th ICIS, Phoenix, Arizona, 2009.

[19] J. Jahn, Vector Optimization: Theory, applications, and extensions. Heidelberg: Springer, 2004.

[20] F. Eisenführ, M. Weber, and T. Langer, Rational Decision Making. Dordrecht: Springer, 2010.

[21] A. A. Mullur, C. A. Mattson, and A. Messac, "New decision matrix based approach for concept selection using linear physical programming," in Proceedings of the 44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference, 2003.

[22] L. J. Heinrich, F. Roithmayr, and A. Heinzl, Wirtschaftsinformatik-Lexikon, München: Oldenbourg, 2004.

[23] C. Zangemeister, "Nutzwertanalyse von Projektalternativen," Logistik Management, vol. 5, no. 2, pp. 50–59, 2003.

[24] M. Weber, Entscheidungen bei Mehrfachzielen. Wiesbaden: Gabler, 1983.

[25] J. Venable, J. Pries-Heje, and R. Baskerville, "A comprehensive framework for evaluation in design science research," in Design Science Research in Information Systems. Heidelberg: Springer, 2012, vol. 7286, pp. 423–438.

[26] J. Venable, "A framework for design science research activities," in Proceedings of the 2006 Information Resource Mgmt. Assoc. Conf., Washington, USA, 2006.

[27] J. Pries-Heje, R. Baskerville, and J. Venable, "Strategies for design science research evaluation," in Proceedings of the European Conference on Information Systems, 2008.

[28] P. Checkland and J. Scholes, Soft systems methodology in practice. Chichester, UK: Wiley, 1990.

[29] C. Sonnenberg and J. vom Brocke, "Evaluation patterns for design science research artefacts," in Practical Aspects of Design Science, M. Helfert and B. Donnellan, Eds. Berlin, Heidelberg: Springer, 2012, pp. 71–83.

[30] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, "User acceptance of computer technology: A comparison of two theoretical models," Management Science, vol. 35, no. 8, pp. 982–1003, 1989.

[31] V. Venkatesh and H. Bala, "Technology acceptance model 3 and a research agenda on interventions," Decision Sciences, vol. 39, no. 2, pp. 273–315, 2008.

[32] J. Palmius, "Criteria for measuring and comparing information systems," in Proceedings of the 30th IRIS in Scandinavia, 2007.