Supporting Strategic Decision-making for Selection of Software Assets

Claes Wohlin[1], Krzysztof Wnuk[1], Darja Smite[1], Ulrik Franke[2], Deepika Badampudi[1] and Antonio Cicchetti[3]

[1]Blekinge Institute of Technology
371 79 Karlskrona, Sweden
claes.wohlin@bth.se; darja.smite@bth.se; krzysztof.wnuk@bth.se; deepika.badampudi@bth.se
[2] Swedish Institute of Computer Science (SICS), Box 1263
164 29 Kista, Sweden
ulrik.franke@sics.se
[3]Mälardalen University, Box 883
721 23 Västerås, Sweden
antonio.cicchetti@mdh.se

Abstract. Companies developing software are constantly striving to gain or keep their competitive advantage on the market. To do so, they should balance what to develop themselves and what to get from elsewhere, which may be software components or software services. These strategic decisions need to be aligned with business objectives and the capabilities and constraints of possible options. These sourcing options include: in-house, COTS, open source and outsourcing. The objective of this paper is to present an approach to support decision-makers in selecting appropriate types of origins in a specific case that maximizes the benefits of the selected business strategy. The approach consists of three descriptive models, as well as a decision process and a knowledge repository. The three models are a decision model that comprises three cornerstones (stakeholders, origins and criteria) and is based on a taxonomy for formulating decision models in this context, and two supporting models (property models and context models).

Component-based software engineering; service-oriented software engineering; decision-making.

1 Introduction

In the advent of software development, companies developed their own operating systems, proprietary programming languages and compilers (e.g. AXE10 developed by Ericsson). Later, companies moved away from this approach to focus their software development efforts on their core business (e.g. telecommunication systems

and features). This maturing software business has spawned two significant trends: specialization and commoditization [11]. Specialization became a direct result of commoditization as companies discovered that to stay competitive they needed to specialize and optimize the costs of developing the commodity parts of their products. At the same time, the increasing popularity of Open Source Software (OSS) accelerated the commoditization process and forced many software companies to look for alternative or multiple revenue streams and new sources of novelty and value. As a result, the primary focus is now on developing software that provides a competitive advantage (e.g. killer apps).

Thus, it is very important for companies to decide what to develop themselves and what to get from elsewhere. On the strategic (executive) level, the strategy of mergers and acquisitions becomes a relevant option of obtaining software and organizations that develop it [31]. However, acquisitions may not always be feasible or possible, e.g. for open source communities that may not be "for sale". Thus, decision-making efficiency also becomes critical for software components that can be realized using internal development resources (in-house), buying COTS, subcontracting or utilizing OSS software. Each of these four sourcing alternatives provides different benefits and consequences, and hence impacts or shapes the business models. For example, obtaining OSS software is often related to joining and participating in a software ecosystem [16] that entails changes in ways of working and potential challenges. Moreover, the selection of one of the four strategies directs the company towards one of the four business model archetypes: creator, distributor, lessor and broker [26]. For many software companies, the time when they could only focus on being creators and thus solving technical challenges is history.

Component-based software engineering has been an important area of research for almost three decades [34] and [35]. As a complement to components, the concept of service-oriented software engineering has emerged [14]. An attempt to bring the two paradigms closer and to use them in a complementary way has been presented in [6]. Here, we use the term "software asset" to denote any type of software, including components and services that can be used for achieving the business objective for a specific system, product or service being developed. Software assets may be divided into four main types when it comes to the source or origin of the asset (henceforth denoted asset origin): in-house, COTS, open source and outsourcing. Within each of these asset origins different assets may fulfill the identified needs, for example, several different COTS may provide the same functionality to the user. In-house refers to assets developed or reused internally within an organization. Thus, in-house includes software having been developed within the same organization, independent of location (e.g. sites in another country), subsidiaries or organizational structure (e.g. different business area). The other three types of asset origins are external, and hence outsourcing is here used as a sourcing option outside the organization that needs a software asset [32].

A key decision to make is what sourcing strategy is the most optimal for an asset. Should it be developed in-house or should we look elsewhere? To date, research has focused on comparing just a few of these asset origins, in particular, in-house versus COTS, and in-house versus outsourcing, and to the best of our knowledge no paper

has addressed all four asset origins [2]. To be able to support these types of decisions in industry, a decision-making approach is outlined here that will form the basis for further research on the topic. The approach consists of three types of descriptive models: decision model, property model and context model, as well as a decision process and a knowledge repository. The main focus here is to look at decision-making between the four different types of asset origins (in-house, COTS, open source and outsourcing), although the models and process described in the paper are expected to be able to adapt to also selecting between different components or services of the same type of asset origin. We do not focus here on mergers or acquisitions as a sourcing strategy for software assets [31].

The remainder of the paper is outlined as follows. Section 2 presents related work from general decision-making theory, decision-making related to different asset origins, and a specific taxonomy intended to help formulating the three descriptive models for the purpose of making the types of decisions discussed in this paper. In Section 3, the proposed models are presented, and in particular their different parts are discussed. Section 4 introduces the concept of an evidence-based knowledge repository to support the decision-making process. A decision-making process outlining how the three descriptive models can be used is presented in Section 5. Section 6 provides a summary and pointers to further work.

2 Related work

2.1 Decision-making

Decision theory largely deals with actors making decisions (e.g. bring an umbrella or not) in the face of uncertain events (e.g. rainfall or not), leading to different outcomes (e.g. wet or dry) and pay-offs (e.g. dry and burdened by umbrella though there is no rain). There are many textbook introductions to the subject, e.g. [28], as well as extensive literature reviews on theories of decision-making under risk [33].

In the area of software engineering research, decision theory has been applied to diverse problems such as evaluating COTS [21], determining optimal intervals for testing and debugging [30], evaluating software designs [7] and assessing non-functional requirements [13]. Decision theory is also one of the cornerstones in the theory of value-based software engineering [4]. Empirical research includes studies on how people make decisions about service level agreements [11] and [12].

The purpose of this paper is not to make a theoretical contribution to decision theory in software engineering and software business, but rather to *apply* it to a particular problem class: how to select an appropriate asset origin for a particular piece of software (component or service). In so doing, we use decision theory terminology and concepts to reason about the problem and present an approach that will make it possible to reuse previous experience and published results alike to make the best possible decision, given the knowledge available.

2.2 Deciding on Origin

The research related to selecting between different software asset origins is quite limited. In a recent systematic literature review [2], which is summarized here, no papers addressing all four types of asset origins were identified. However, some papers addressing two or in a few cases three origins were found.

The decision models for in-house vs. COTS are mainly based on optimization models. The optimization models proposed in [9], [10], [17], [18], [27] and [34] help to decide which components should be developed in-house and which should be bought. Cost, delivery time, and reliability are the common objectives and constraints considered in all the proposed optimization models. The optimization models either consider single objective or multiple objectives in the decision model.

The objective in the optimization models proposed in [9], [10], [27] and [34] is to minimize cost under reliability and delivery time constraints. The CODER framework proposed in [9] consists of a decision model based on optimization and accepts UML notations as an input. In [31] and [34], the authors propose an architecture optimization approach based on a swarm intelligence algorithm. The CODER framework [9] is extended in [26] and [27], allowing decision-making as early as requirements are available. Similarly, a general non-linear optimization model is proposed in [10] for the same objective and constraints i.e. minimizing cost under reliability and time constraints.

Multi-objective optimization models have been proposed in [17] and [18]. A decision model for fault-tolerant systems is proposed in [15] and [17] with two objectives – to maximize reliability and minimize cost under a time constraint. In addition, coupling and cohesion have been considered in the decision model proposed in [18]. The objectives in [18] are to maximize intra-modular coupling density and functionality under time, cost and reliability constraints.

Two papers focus on deciding between in-house and outsourcing [19] and [20] were identified. The model in [19] provides tool support for requirements clustering to find a cohesive group of requirements using a graph-based model. In [20], the authors propose a decision model using decision tables. The input is the knowledge specificity (business, functional and technical), and interdependencies (priority between software components and communication intensity among developers).

2.3 GRADE Taxonomy

The work presented in this paper is grounded in the GRADE taxonomy [22] and [24]. The GRADE taxonomy summarizes the relevant concepts and definitions for building models related to decision-making and supporting decision processes. On the highest level, the GRADE taxonomy combines five fundamental concepts of decision-making for software intensive systems: *Goals*, *Roles*, *Assets*, *Decision* and *Environment* (GRADE). These five fundamental concepts can be used as building blocks for creating models supporting decision-making.

Goals represent the starting point for a decision. They represent the internal business goals and customer goals, and have a broad impact on the entire product or even organization. The goals form an important input to the decision-making.

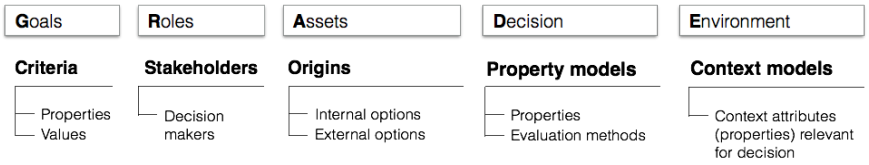

Fig. 1. Mapping of GRADE to concepts in the decision model and the supporting models.

Roles in the GRADE taxonomy represent individuals involved in the decision-making. The roles are classified into types, functions, levels and perspectives.

The assets concept in the GRADE taxonomy describes the decision assets (often encapsulated in a software component or software service) characterized by: origin, attributes, type, usage and realization options.

The *decision* concept of GRADE contains the decision methods that can be used for estimating outcomes for a specific option among those evaluated in the decision-making process.

The *environment* concept of the GRADE taxonomy describes the environment before the decision was analyzed or made. It includes the characteristics of organizations, products, stakeholders, markets and business prior to making a decision.

2.4 Decision-making in software business

Running a software business requires making several decisions on multiple levels [1], ranging from strategic decisions about mergers, acquisition and take-overs [31], via tactical decisions on which ecosystem to join and support [16] to highly technical decisions on how to realize customer requirements in software. An increasing number of software companies evolve from the pure creator business archetype that implies code ownership but also development risk, high maintenance cost and full responsibility for delivering the required quality towards mixed or hybrid business models that imply taking on several business archetype roles [26]. At the same time, small and large companies take on outsourcing initiatives to reduce development costs and obtain valuable knowledge and inspiration. This shifts the center of gravity towards integration work and coordination of outsourced (often also offshored) sites into software products that deliver the value that customers expect. Finally, joining or creating an ecosystem entails a series of decisions regarding growing a healthy ecosystem [16], participating in ecosystem development and gaining importance and influence or disrupting markets by commoditization of ecosystem software. Each of the mentioned four asset origins thus has different implications both in the short term and in the long term. They come with different costs and prices and can bring different benefits. Decision-makers responsible for running their software businesses are faced with increased decision complexity and frequency that they need to cope

with to succeed with their business endeavors. An example here is decision-making in cloud computing environments for selecting appropriate services from different providers [22].

3 Descriptive models

Three descriptive models are built from the GRADE taxonomy to ensure that no aspect is missed in the decision-making. Thus, the descriptive models become part of an instantiation of the taxonomy. The instantiation includes two main parts: description of the concepts based on GRADE (the three descriptive models) and the actual decision-making process.

The main objective of the decision approach presented is to enable a systematic way to select between different software asset origins, including potentially both software components and software services. The types considered represent four main asset origins: in-house, COTS, open source and outsourcing.

The three descriptive models correspond to the five fundamental concepts in GRADE, as described and mapped in Figure 1. In particular, the five concepts comprise: 1) the three decision model cornerstones: stakeholders (roles), origins (assets) and criteria (goals); and 2) two supporting models – property models (decision) and context models (environment). The decision model with its three decision cornerstones are described in Section 3.1, the property models are discussed in Section 3.2 and the context models are further elaborated in Section 3.3.

In addition to experience of the involved stakeholders, it is beneficial to support the decision-making with related historical evidence and experiences. This can be captured in an evidence-based knowledge repository, which is elaborated in some more detail in Section 4.

3.1 Decision Model

The decision model consists of three main cornerstones:

Stakeholders – which stakeholders (and hence different perspectives) need to be involved? The stakeholders should be identified from the roles in GRADE that should be involved in the decision-making. The creator, distributor, lessor and broker business archetypes [26] help in identifying relevant stakeholders that influence the value creation and delivery processes. The current model involved both internal stakeholders as well as end customers and external stakeholders. As many software companies currently run hybrid business models with additional revenue streams originating from cross-selling and complementariness, the set of potential stakeholders is much broader than in the in-house scenario.

The stakeholders have different perspectives (as described through the *Roles* concept in GRADE) that should be taken into account in the decision-making process. This could be exemplified with the following five software engineering areas: 1) business and requirements engineering, 2) non-functional properties, 3) life-cycle perspective, 4) architecture, and 5) implementation and integration, including

verification and validation. The business and requirements engineering perspective is responsible for capturing the customer value and translating it to the form that can be used for decision-making. Business analysts and requirements engineers play key roles in capturing and prioritizing customer needs. Other perspectives may also be relevant, for example the strategic management perspective.

Origins – which type of asset origins should be considered (in-house, open source, COTS and/or outsourcing)? In this case, the asset concept in GRADE is defined as potentially coming from four different asset origins. Thus, it is assumed that the main decision to be taken relates to where a software component or service needed in a product or system is developed, obtained or acquired. The actual choice of, for example, a specific COTS component is not considered, i.e. the selection between competing alternative assets of the same origin.

Criteria – which criteria should be evaluated to ensure an informed decision? The criteria are based on the *Goal* concept in GRADE. Since the goals may be quite general, some goals may not be relevant for a specific decision. It is important to acknowledge here that criteria can have at least three perspectives: customer perspective, internal-business perspective, and community (or ecosystem perspective). The goals and criteria should be identified and tagged by the relevant perspective and potential conflicts between perspectives should be identified and mitigated. The involved stakeholder roles should review the goals, mitigate potential conflicts and translate them into defined decision criteria to be used in the decision-making. Criteria should be more detailed than the goals and need to be measurable, i.e. contain a threshold for a certain property attribute (e.g. a specific attribute of software quality or gaining 1 000 000 users of a software service within 2 months after the service is launched). Thus, criteria should be possible to evaluate, for example, they could state that a certain property should be above a certain threshold, and each criterion should be evaluated for each viable asset origin. The chosen criteria should be evaluated, where business risk most likely is always one of the criteria. Risk is a criterion by itself in relation to a specific asset origin, e.g. the risk of a COTS supplier going bankrupt. However, risk is also related to the uncertainty in specific decisions, their criteria, and the data they are based on, e.g. uncertainty in historical cost or reliability figures.

The **stakeholders** contribute to the decision model as experts in their own area, for example, business, architecture or requirements. They are involved in evaluating possible asset **origins** viable for the specific case and formulating the **criteria** for the decision based on the goals. Furthermore, the experts provide input to the property models (see Section 3.2), they should describe the context of the decision (see Section 3.3) and they should help in identifying similar historical evidence and experiences using the evidence-based knowledge repository (see Section 4). The latter includes prioritizing among important factors to compare with historical evidence.

3.2 Property Model

The **decision** concept in GRADE includes both models to estimate specific properties and methods to, for example, weigh different criteria. The property models come into play in estimating outcomes of the criteria for different asset origins, i.e. there is a need to make the estimations wrt to different criteria for the relevant origins.

A property model is an estimation model with respect to a decision criterion. The property model consists of a well-defined property and an evaluation method. For example, the property can be the number of active users and the evaluation method can be to check how many of these users have used the service the last seven days. A property model may contain other property models. Examples of properties include coordination costs, IT service costs and maintenance costs for selecting cloud computing services [22]. The evaluation method may be quite simplistic, for example, expert opinion or based on a sophisticated formal mathematical decision model [1]. Property models can also be more advanced, e.g. for the reliability criterion using software reliability growth models (SRGM) based on historical data from similar situations. Furthermore, some evaluation methods use generic statistical methods such as regression analysis, while others are based on general methods but still are tailored for a specific purpose such as SRGMs. Properties can and should also be estimated for aspects relevant for communities, ecosystems and markets and not only for a company's internal or a project's internal aspects. A good example here could be the degree of influence on ecosystem members or the state of a company's reputation in a given ecosystem [16].

Property models provide estimates of values for the different criteria, and in most cases the property models only handle one or a few properties at the time. Thus, there is a need to decide the priorities of the different criteria and hence the weighing between them, for example is cost more or less important than security. The methods for managing the priorities between criteria, or for combining outcomes in different ways are referred to as decision methods. For this purpose, it would be possible to use, for example, methods such as AHP [29] and HCV [3].

As part of the decision-making, it should be decided, for example, whether the stakeholders should try to take different time perspectives into account "manually" or if the property models should instead be used more than once, for example, to make estimations both for a short-term and a long-term perspective.

3.3 Context Model

The context model is a representation of the environment in which the decision is taken. There are two main objectives of the context model. First, it helps in identifying relevant criteria, property models and solutions previously used by others. Second, it structures the decision at hand for future use in the evidence-based knowledge repository. An example of a context model representation is presented in [25]. It comprises six dimensions of the environment, four that capture the organizational characteristics (including practices and tools) and two that are external to the organization (business environment characteristics). The context model also

extends the environment concept in GRADE as it helps to understand the context in the future and is integrated with the evidence-based knowledge repository described in Section 4.

The context model should capture the current situation within an organization with respect to 1) product before the decision, 2) people involved in relation to the decision, 3) processes as well as 4) practices, techniques and tools. Furthermore, 5) the organization as such should be captured, 6) the market should be described as a part of the context and other relevant aspects from the ecosystem that a company is involved in. We believe that for a comprehensive context description that includes business characteristics and can be effectively used for guiding business decisions, a possible future area of research is to expand the six dimensions described in [25] to better cover aspects such as the market, ecosystems and also business models.

4 Evidence-based knowledge repository

Historical information should be structured so that it is possible to find relevant or similar cases, for example, similar context, prioritized similar criteria or an interest in the same asset origins. The stored information may facilitate decision-making, but also to provide what is generically known as traceability of a decision: what a decision was about, who made the decision, and why the decision was made. This is often referred to as the rationale for a decision. In this respect, any repository ought to record all relevant aspects of a decision-making scenario. Furthermore, a repository ought to contain other available information such as research articles on the topic, and in particular systematic literature reviews, as well as publically available data or data shared between trusted partners that can help support different steps of decision-making.

Former decision information can represent an important support in the decision-making process, at least to avoid errors made in the past. Therefore, if the repository was considered as a mere post-decision storage support, it is difficult to justify and motivate the effort of documenting decisions in detail. Furthermore, the repository would miss a lot of its potentials: 1) as mentioned before, recurring decisions might contain important lessons learned; and 2) multiple decisions could entail an agreement about a more general development vision (e.g., different properties derivable from the same goal by different stakeholders), thus requiring consistency. Thus, continuous and reliable data collection, as well as use of the data, should be performed to unlock the full potential that an evidence-based knowledge repository offers.

The repository should be able to smoothly manage large amounts of data and should offer meaningful mechanisms to retrieve decisions as filtered by their prominent characteristics (i.e., the cornerstones of the decision model), and pointers to relevant studies on the topic. Compatibility and interoperability are important quality attributes of a good decision knowledge repository and therefore we recommend using open data standards supported by reliable quality management measures, e.g. ISO/IEC 25012 SQuaRE [15], OGD eight principles [23] or Web Information Quality assessment [5].

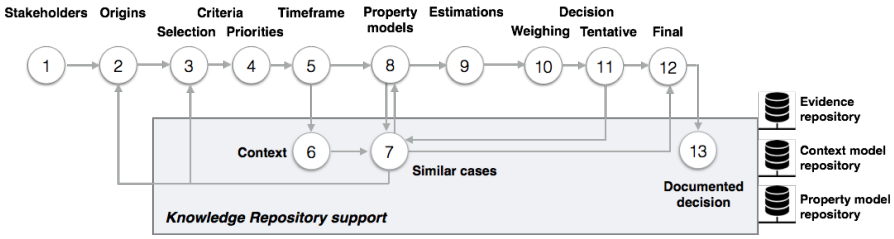

Fig. 2. The decision-making process including having a knowledge repository.

5 Decision-making process

The decision-making process represents the actual conduct of decision-making, and it is illustrated in Figure 2 using the numbering of the recommended steps below. Some steps may be perceived as more important than others. However, it has been chosen to present all steps as recommended steps, since the actual usefulness of the different steps and preferred order of the steps may vary from case to case. Thus, the order of the steps should be seen as one possible suitable order. Furthermore, an evidence-based knowledge repository may not be available in all cases, and hence those steps may not be applicable in all cases. It should also be noted that iterations are expected. They may appear between any steps depending on the specific decision, or the specific circumstances in relation to a decision. Thus, Figure 2 only illustrates the expected iterations based on the evidence-based knowledge repository.

The recommended steps in the decision-making process are as follows:

1) Identify stakeholders to be involved in the decision – It is important to ensure coverage of roles and persons to make sure that the decision made is possible to implement efficiently. Each stakeholder that is relevant for the decision and its consequences for the business should be identified here.
2) Evaluate the suitability of the four asset origins – The possible origins for a software asset should be identified. This includes investigating the technical and business compatibilities and the short and long term costs of selecting each asset option. In certain cases, not all asset origins are allowed or suitable. In some cases, the main decision is whether to do development in-house or going externally. Sometimes, open source solutions are not an option. Thus, the possible asset origins need to be identified carefully.
3) Decide criteria from goals – Based on the goals of the development, criteria (both business and technical) have to be decided and suitable targets have to be set. The latter should be done so that different asset origins can be evaluated and compared with each other. In most cases, risk needs to be considered as one criterion, since it may differ substantially for different asset origins (in-house, COTS, open source and outsourcing).

4) Decide on priorities of criteria – In addition to deciding on targets for each criterion, it is also important to decide how they should be prioritized, e.g. using AHP [29] or HCV [3]. It may also be the case that certain stakeholders have more say in a decision, which has to be taken into account, i.e. different stakeholder roles may need to be weighed differently in the prioritization process.
5) Decide on how to handle the time aspect – Certain solutions may be perceived better or worse in the short-term and long-term respectively. For example, a certain solution may be very good to get a product on the market, but it is not very good for the long-term architecture of the product. The time aspect is highly relevant for decisions that concern ecosystem participation or OSS involvement as in these two cases the competitive advantage created based on the ecosystem or OSS commodity layer comes with a long term maintenance cost. Thus, selective revealing should be considered and based on competitive advantage time estimates. The degree of commoditization or commoditization index should be projected onto the average sale time for new products. To cope with the time aspect, the decision-makers either have to take time aspects into account when prioritizing between different asset origins or evaluations have to be done separately for different time aspects, e.g. short-term and long-term, and the tradeoff between them has to be agreed upon.
6) Describe the context – To enable comparison with previous cases internally and externally as well as with the research literature, the case has to be described. This should be done using the context model, where salient aspects have to be captured. This may include business model(s) used, application domain, system size and development method as well as a range of other aspects [25]. Independently, it is crucial to capture these aspects to enable identification of similar cases and hence relevant evidence and experiences.
7) Look for similar cases in a knowledge repository – The identification of similar cases is done using the context model as well as the asset origins considered as suitable and the criteria. Thus, a similar case is defined as having some key aspects of the context in common (from Step 6) as well as a focus on similar criteria (from Steps 3 and 4) and similar suitable asset origins (Step 2). Similar cases are identified and studied to identify evidence and experience that are perceived important in the current case and to uncover potential alternative decision scenarios [8]. The knowledge repository could be solely based on internal cases or a more elaborate database containing both internal and external cases. The information in the knowledge repository may indicate that in other similar cases other asset origins, criteria, property models or decisions have been considered. Thus, it is important to be able to challenge the choices made in the other steps as illustrated in Figure 2.
8) Decide on property models to use – Once the criteria are decided, there is a need to decide how the criteria should be evaluated. If having a knowledge repository, this can be done by retrieving valuable information from the knowledge repository in terms of what others have used in similar cases (Step 7). If there is no knowledge repository, the property models for each criterion

have to be decided without additional support, whether they are expert opinions or more advanced estimation models.

9) Make estimations using the property models – Given the chosen property models, estimations need to be done for each criterion for the asset origins under consideration and potentially for different time aspects based on the approach decided in Step 5.
10) Weigh the estimation results of the selected properties based on the priorities of criteria – Based on the priorities of the different criteria, the estimation results from the different property models should be weighed together. This is non-trivial given that the values as such cannot be combined easily in many cases. It is rather the estimation of each criterion and its distance from the targets that need to be weighed together.
11) Make a tentative decision – Once the outcomes from the property models have been weighed together, it should be possible for the decision-makers to make a tentative decision. If a knowledge repository is available, it is recommended to browse previous decisions and review relevant tentative scenarios and compare the tentative decision with decisions from similar cases as described in Step 7. Relevant business context factors should be evaluated here based on similar cases. This should be done to make a final evaluation of the decision, and ensure that the reasoning done is as correct as possible and that no relevant available information is ignored.
12) Make a final decision – This has been the objective of the decision-making process and hence it is a very important step for the development. It is important that the stakeholders are able to communicate both the actual decision and the rationale for the decision.
13) Store the case in the knowledge repository – The case information, including the context model, the criteria used, the stakeholders involved and the asset origins considered should be carefully documented. This step is important as it allows for transparency if the case is properly documented (including the decision rationale) and helps to organically grow the evidence-base knowledge repository. It is important to add new cases given the speed of change and hence ensure that recent cases are available for decisions to come.
14) At the end of the decision-making process, the objective is that the stakeholders should have come to either a consensus or at least that the involved stakeholders know why the decision was made, and are able to communicate it in the organization.

6 Summary and further work

The decision support models and process may seem complex, but they address a challenging area for companies. The development of today's software products, systems and services is a complex endeavor. The decisions of choosing software components (or services), whether being in-house development vs. external options such as COTS, open source and outsourcing, are most often strategic decisions and

they heavily influence competitiveness. The approach presented in this paper provides a starting point for supporting such decisions and address the research gap identified in a recent systematic literature review [2].

The approach addresses several key questions to make a decision with respect to selecting the origin of software assets (components and services). However, before using the approach the actual decision needs to be determined, ***what*** to decide. The decision process as such illustrates ***how*** a decision may be made. Furthermore, ***who*** makes the decision is determined by the identification of the stakeholders. The main reasons for the decision, i.e. ***why*** a decision is made, are captured through the criteria in the decision model.

The focus of this work is on selecting between different types of asset origins, and not between different actual components or services of the same type. The objective is to integrate selection of competing specific alternatives into the models and process, including both the tradeoffs between components and services as well as between different components or services of the same asset origin. This is part of further work as well as to empirically evaluate the proposal through case studies.

The presented models and process is based on the assumption that the stakeholders involved into the decision-making process capture customer needs and values. Thus, the model can be applied for both B2B and B2C contexts as long as all relevant stakeholders are identified and involved in decision-making. For B2C contexts, end users and other external stakeholders need to be involved and accurately represented.

In future work, we plan to survey a number of business scenarios that involve diverse business models, asset origins, company characteristics and ecosystem participation models. We aim at characterizing these scenarios by identifying common and variable parts and clearly outlining short- and long-term consequences of each decision alternative. These should form guidelines that software business practitioners may use when considering various sourcing options. Moreover, we plan to expand our research on the evidence-based knowledge repository and create the first implementation of a repository that can support decision-makers. Finally, we plan to conduct an empirical study that will evaluate the presented decision-making approach and identify future work directions.

Acknowledgments.
The work is supported by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden. Furthermore, we would like to thank our colleagues in the ORION project for fruitful discussions and the external reviewers that have helped improving the paper.

References

[1] Aurum A. and Wohlin, C.: The Fundamental Nature of Requirements Engineering Activities as a Decision-Making Process, *Inf. and Soft. Tech*., 45, (2003) 945–954

[2] Badampudi, D., Wohlin, C. and Petersen, K.: Software Component Decision-making: In-house, Open source, COTS or Outsourcing - A Systematic Literature Review. *In revision after review for journal publication* (2016)

[3] Berander, P. and Jönsson, P.: Hierarchical Cumulative Voting (HCV) - Prioritization of Requirements in Hierarchies. *International Journal of Software Engineering and Knowledge Engineering* 16, (2006) 819–849

[4] Biffl, S., Aurum, A., Boehm, B., Erdogmus, H. and Grünbacher, P. (eds.). Value-Based Software Engineering. Springer Science & Business Media (2006)

[5] Bizer, C. and Cyganiak, R.: Quality-Driven Information Filtering Using the WIQA Policy Framework, *Web Semantics: Science, Services and Agents on the WWW*, 7 (2009) 1-10

[6] Breivold, H. P. and Larsson, M.: Component-Based and Service-Oriented Software Engineering: Key Concepts and Principles. *Proceedings of the 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA (2007)*13–20

[7] Cárdenas-Garcia, S. and Zelkowitz, M. V.: A Management Tool for Evaluation of Software Design. *IEEE Transactions on Software Engineering* 17, (1991) 961–971

[8] Cicchetti, A., Borg, M., Sentilles, S., Wnuk, K., Carlson, J. and Papatheocharous, E.: Towards Software Assets Origin Selection Supported by a Knowledge Repository. In *1st MARCH Workshop at WICSA and CompArch 2016, April 5, Venice (Italy)* (2016)

[9] Cortellessa, V., Marinelli, F. and Potena, P.: Automated Selection of Software Components Based on Cost / Reliability Tradeoff. *Proceedings of the 3rd European Workshop on Software Architecture (EWSA'06)*, (2006) 66–81

[10] Cortellessa, V., Marinelli, F. and Potena, P.: An Optimization Framework for "Build-or-buy" Decisions in Software Architecture. *Computers and Operations Research* 35 (2008) 3090–3106

[11] Cusumano, M. A.: The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad, Simon and Schuster, (2004)

[12] Franke, U. and Buschle, M.: Experimental Evidence on Decision-Making in Availability Service Level Agreements. *IEEE Transactions on Network and Service Management*, 13, (2016) 58-70

[13] Gregoriades, A. and Sutcliffe, A.: Scenario-Based Assessment of Nonfunctional Requirements. *IEEE Transactions on Software Engineering* 31, (2005) 392–409

[14] Huhns, M. and Singh, M. P.: Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing* 9, (2005) 75–81

[15] ISO/IEC 25012: http://iso25000.com/index.php/en/iso-25000-standards/iso-25012

[16] Jansen, S. Brinkkemper, S. and Cusumano, M. A.: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, Edward Elgar Publishing, (2013)

[17] Jha, P. C., Bali, S., Kumar, U. and Pham, H.: Fuzzy Optimization Approach to Component Selection of Fault-tolerant Software System. *Memetic Computing* 6, (2014) 49–59

[18] Jha, P. C., Bali, V., Narula, S. and Kalra, M.: Optimal Component Selection Based on Cohesion & Coupling for Component Based Software System Under Build-or-buy scheme. *Journal of Computational Science* 5, (2014) 233–242

[19] Kramer, T. and Eschweiler, M.: Outsourcing Location Selection with SODA: A Requirements Based Decision Support Methodology and Tool. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7908 LNCS, (2013) 530–545

[20] Kramer, T., Heinzl, A. and Spohrer, K.: Should this Software Component be Developed Inside or Outside our Firm? - A Design Science Perspective on the Sourcing of Application Systems. In *New Studies in Global IT and Business Service Outsourcing*. Springer Berlin Heidelberg, (2011) 115–132

[21] Lawlis, P. K., Mark, K. E., Thomas, D. A. and Courtheyn, T.: A Formal Process for Evaluating COTS Software Products. *Computer* 34, (2001) 58–63

[22] Martens, B. and Teuteberg, F.: Decision-Making in Cloud Computing Environments: A Cost and Risk Based Approach, *Information Systems Frontiers*, 14, (2012) 871-893

[23] Open Government Data (OGD): https://opengovdata.org/

[24] Papatheocharous, E., Petersen, K., Cicchetti, A., Sentilles, S., Shah, S. M. A. and Gorschek, T.: Decision Support for Choosing Architectural Assets in the Development of Software-Intensive Systems: The GRADE taxonomy. *Proceedings of the 1st International Workshop on Software Architecture Asset Decision-making, Article No. 48 (*2015)

[25] Petersen, K. and Wohlin, C.: Context in Industrial Software Engineering Research. *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, (2009) 401–404

[26] Popp, K. M.: Software Industry Business Models, IEEE Software, 28 (2011) 26-30

[27] Potena, P. L.: Composition and Tradeoff of Non-functional Attributes in Software Systems. *European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, (2007) 583–585

[28] Resnik, M. D.: *Choices: An Introduction to Decision Theory*. University of Minnesota Press (1987)

[29] Saaty, T. L.: Decision Making with the Analytic Hierarchy Process. *International Journal of Services Sciences* 1, (2008) 1-83

[30] Singpurwalla, N. D.: Determining an Optimal Time Interval for Testing and Debugging Software. *IEEE Transactions on Software Engineering* 17 (1991) 313–319

[31] Schief, M., Buxmann, P and Schiereck, D.: Mergers and Acquisitions in the Software Industry*, Business & Information Systems Engineering*, 5, (2013) 421-431.

[32] Šmite D., Wohlin, C., Galviņa, Z. and Prikladnicki, R.: An empirically Based Terminology and Taxonomy for Global Software Engineering. *Empirical Software Engineering* 19, (2014) 105–153

[33] Starmer, C.: Developments in non-expected utility theory: the hunt for a descriptive theory of choice under risk. *Journal of Economic Literature* 38, (2000) 332–382

[34] Ssaed, A. A., Wan Kadir, W. M. N. and Hashim, S. Z. M.: Metaheuristic search approach based on in-house/out-sourced strategy to solve redundancy allocation problem in component-based software systems. *Int. J. of Soft. Eng. and its Applic.* 6, (2012) 143–154

[35] Vale, T., Crnkovic, I., de Almeida E. S., da Mota Silveira Neto, P. A., Cerqueira Cavalcantic, Y., and de Lemos Meira, S. R.: Twenty-eight years of component-based software engineering. *Journal of Systems and Software* 111, (2016) 128–148