A Decision-making Process-line for Selection of Software Asset Origins and Components

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

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

Abstract

Selecting sourcing options for software assets and components is an important process that helps companies to gain and keep their competitive advantage. The sourcing options include: in-house, COTS, open source and outsourcing. The objective of this paper is to further refine, extend and validate a solution presented in our previous work. The refinement includes a set of decision-making activities, which are described in the form of a process-line that can be used by decision-makers to build their specific decision-making process. We conducted five case studies in three companies to validate the coverage of the set of decision-making activities. The solution in our previous work was validated in two cases in the first two companies. In the validation, it was observed that no activity in the proposed set was perceived to be missing, although not all activities were conducted and the activities that were conducted were not executed in a specific order. Therefore, the refinement of the solution into a process-line approach increases the flexibility and hence it is better in capturing the differences in the decision-making processes observed in the case studies. The applicability of the process-line was then validated in three case studies in a third company.

Keywords: Component-based software engineering; decision-making; case study.

1 Introduction

In the early days of software development, it was not uncommon to internally develop software product features, operating system or even programming languages and compilers (e.g. AXE10 developed by Ericsson used an operating system and programming language developed in-house). As the software business matured, two significant trends emerged: specialization and commoditization [12]. Specialization may be viewed as a result of commoditization, as many companies embraced specialization as a means to stay competitive. The commodity parts of their products were most often taken off the shelf. The increasing popularity of Open Source Software (OSS) helps accelerating the commoditization process and encouraged 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.

Nowadays, companies need to decide what to develop themselves and what to get from elsewhere. On the strategic (executive) level, the strategy of mergers and acquisitions is a relevant option for obtaining software and the organizations that develop it [38]. However, acquisitions may not always be feasible or possible, including for example OSS assets. Decision-making efficiency in relation to software assets becomes important as they can be realized using internal development resources (in-house), buying Components off-the-shelf (COTS), subcontracting (outsourcing) or utilizing OSS software. The four asset sourcing alternatives provide different benefits and consequences (e.g. competitiveness), and hence affects or shapes the business models. For example, using OSS software is in many cases related to joining and participating in a software ecosystem [21]. Furthermore, the selection of one of the four alternatives directs the company towards one of the four business model archetypes: creator, distributor, lessor and broker [32]. However, for many software companies, the time for being only creators and solve technical challenges is history.

We use the term “software asset” to denote any type of software, including components that can be used for achieving the business objective for a specific system or product being developed. Software assets may be divided into four main types based on the source or origin of the asset (henceforth denoted asset origin): in-house, COTS, OSS and outsourcing. Within each of these asset origins, different assets may fulfil the identified needs, e.g., 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, COTS and OSS components are provided from an external source and outsourcing is here used as a sourcing option outside the organization that needs a software asset [40]. Component-based software engineering has been an important area of research for almost three decades [42] and [43].

The asset sourcing strategy that is the most optimal for an asset is an important decision for companies. Should it be developed in-house or should it be sourced/looked for elsewhere? To date, research has focused on comparing just a few of these asset origins, e.g. in-house versus COTS, and in-house versus outsourcing. To the best of our knowledge, no paper has addressed all four asset origins [3]. To address this gap, we proposed a set of different decision-making activities and packaged them in a decision-making process for selecting software asset origins [45]. Three types of descriptive models: decision model, property model and context model, as well as a knowledge repository were used as inputs for formulating the set of decision-making activities and the process. The input helped in identifying the different activities that support in answering several key questions required to make a decision. To build a decision-making process, the scope of the actual decision needs to be determined, i.e. ***what*** to decide. The decision-making 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 solution presented in our previous work is presented as a process [45]. However, based on the validations conducted in the case studies presented here, we concluded that the concept of a common process might not be the best fit to reality. In addition, our view of the previous solution was closer to a checklist rather than a prescribed process. Therefore, from now on we refer to the previous solution as a checklist, which here more appropriately is described as a process-line, and not a process. The initial goal of the set of activities captured in our previous solution [45] was to support the selection between the four different types of asset origins (in-house, COTS, OSS and outsourcing), although the solution proposed in [45] was expected to be adaptable also to select between different components of the same type of asset origin. Therefore, in this paper the selection between different components of the same type of asset origin is considered as one of the cases for validating the checklist. We do not focus here on mergers or acquisitions as a sourcing strategy for software assets [38]. The evaluation is aimed at validating the coverage of the activities represented by the checklist. We also noticed during the validation that the decision-makers neither executed the decision-making activities orderly nor did they follow the proposed order of execution in our previous work [45].

The main contribution of this paper is to provide a process-line from which decision-makers can include or exclude activities depending on their case. Therefore, the process-line presented in this paper is a set of activities (without any prescribed order) that enables the decision-makers to build a tailored decision-making process to select between the four asset origins and between different components. Note that the goal is to provide a set of decision-making *activities* and not decision-making *criteria* for the selection of asset origins and components.

The extension of our previous work [45] is focused on validating the coverage of the activities represented by the checklist, formulating a process-line and validating the applicability of the process-line in supporting the decision-makers to build their

decision-making process, in particular the following additional aspects are covered: 1) an in-depth description of research methodology related to the design of the checklist and the process-line and the execution of the five case studies; 2) description of the case studies that validate the proposed checklist; 3) an extended discussion of the nature of the decision-making process-line; and 4) validating the applicability of the process-line from a coverage point of view and the implications that can be drawn from conducting the industrial case studies.

The remainder of the paper is outlined as follows. Section 2 presents background information from general decision-making theory, and a specific taxonomy intended to help formulating three descriptive models for the decisions discussed in this paper. It also introduces the frame of reference in terms of the three descriptive models and a knowledge repository that were used as an input to identify the set of decision-making activities that form the checklist. The related work on decision-making related to different asset origins and decisions in software business is discussed in Section 3. In Section 4, we present the research methodology utilized to formulate the checklist, the process-line and its validation using five case studies. The process-line is presented in Section 5. Section 6 presents the two case studies where the checklist was validated at two companies and three cases where the process-line was validated. A discussion of implications for research and practice is provided in Section 7. Finally, Section 8 provides a summary and pointers to further work.

2 Background

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. it rained and even though burdened by the umbrella, you are dry). There are many textbook introductions to the subject, e.g. [35], as well as extensive literature reviews on theories of decision-making under risk [41].

In the area of software engineering research, decision theory has been applied to diverse problems such as evaluating software components [26], testing [39], architecture [7] and requirements engineering [19]. Decision theory is also one of the cornerstones in the theory of value-based software engineering [5]. Empirical research includes studies on how people make decisions about service level agreements [18].

The objective here 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 to choose the software component itself. In so doing, we use decision theory terminology and concepts to reason about the problem and present a decision-making process-line 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 GRADE taxonomy

The work presented in this paper is grounded in the GRADE taxonomy [29]. The taxonomy summarizes the relevant concepts and definitions for decision-making for selecting between asset origins. On the highest level, it combines five fundamental concepts of decision-making for software intensive systems: *Goals*, *Roles*, *Assets*, *Decision* and *Environment* (GRADE) as illustrated in Figure 1. These five fundamental concepts can be used as building blocks for creating models supporting decision-making.

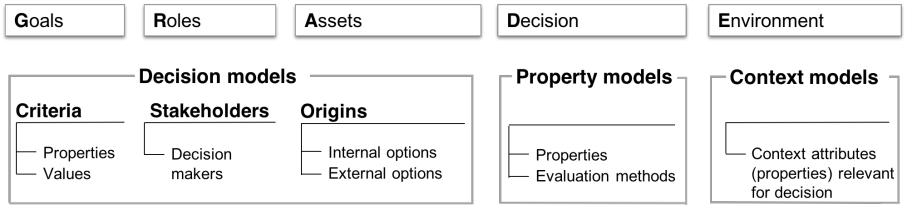

Figure 1: Mapping of GRADE to concepts in the decision model and the supporting models.

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.
Roles represent individuals involved in the decision-making. The roles are classified into types, functions, levels and perspectives.
The assets describe the decision assets (often encapsulated in a software component) characterized by: origin, attributes, type, usage and realization options.
The *decision* 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* describes the environment before the decision was analysed or made. It includes the characteristics of organizations, products, stakeholders, markets and business prior to making a decision.

Our previous work [45] is based on three descriptive models that capture the concepts for decision-making as proposed by the GRADE taxonomy. We present these related concepts in the following section. It should be noted that the descriptive models and the evidence-based knowledge repository described below were used as inputs to formulate the checklist. The checklist provides an overview of the set of activities that could be considered in the decision. The working of each activity is case dependent, for example, the checklist recommends that the decision-makers should consider the selection of the appropriate property model/s as an activity to estimate or evaluate the criteria. However, the selection of the specific property model depends on the specific

decision. A brief description of the descriptive models such as the property model is provided below. The detailed working of the different models is described in [8], [9] and [37], and it is considered out of the scope of this paper as it is only used as an input to identify the set of decision-making activities.

2.3 Descriptive models

Three descriptive models are built from the GRADE taxonomy to ensure that no decision-making aspect is missed. 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).

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 more detail in Section 2.4.

2.3.1 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 stakeholders involved into the decisions can be categorized into: initiators, influencers/contributors (preparation) and decision-makers [31].
The stakeholders have different perspectives (as described through the *Roles* concept in GRADE) that should be taken into account in the decision-making process. The perspectives include product requirements aspects that are more short-term (business, functionality and quality aspects) as well as life-cycle aspects that are usually more long-term (architecture, support and maintenance) [46].
Origins – which type of asset origins should be considered (in-house, OSS, 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 needed in a product or system is developed, obtained or acquired. The actual choice maybe between all relevant asset origins or a subset of the asset origins. This may also mean that the suitability of only one asset origin is evaluated to select 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. 99.99 % service availability or gaining 1 000 000 users of a software service within two months after the service component is launched). Thus, criteria should be possible to evaluate, e.g., 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 is most likely 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, e.g., 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 2.3.2), they should describe the context of the decision (see Section 2.3.3) and they should help in identifying similar historical evidence and experiences using the evidence-based knowledge repository (see Section 2.4). The latter includes prioritizing among important factors to compare with historical evidence.

2.3.2 Property model

The **decision** concept in GRADE includes both models to estimate specific properties and methods to, e.g., weigh different criteria. The property models come into play in estimating outcomes of the non-functional[1] criteria [37] for different asset origins, i.e. there is a need to make the estimations with respect to different criteria for the relevant origins. Non-functional properties of the component candidates correspond to all properties beyond functionality and describe "how" a component performs or delivers its functionality. Properties are closely related to quality aspects and external aspects of a component, e.g. number of active users, source code quality or dependability. The origin of the component determines the scope of property attributes and often constrains estimation methods for these properties.

The property model ontology [37] introduces the two main elements of a property model: 1) non-functional properties that have name, data format and documentation and 2) evaluation method(s) that have names, output, unit, applicability, parameters, driver, formula, description and implementation. A valid property model has to include at least one property and at least one evaluation method [37]. Recent research on property model ontology identified the following non-functional properties as

[1] Non-functional criteria/properties are sometimes also referred to as extra-functional or quality criteria/properties.

being important in the automotive domain: cost (development effort), performance, configurability (variability), flexibility, maintainability, testability, power consumption, reliability, safety, evolvability and security.

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 [27]. The evaluation method may be quite simplistic, e.g., expert opinion or based on a sophisticated formal mathematical decision model [1]. Recent work identified that COCOMO or estimation by analogy or expert estimation are used in the automotive domain for estimating the development effort and worst-case execution time is used to estimate complexity [37]. The decision could either have positive (goals achieved) or negative (goals not achieved) outcome. Documenting the decisions irrespective of the method used to log the success and failure stories is beneficial. This is further discussed in Section 2.4.

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 [21].

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, e.g. 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, e.g., methods such as AHP [36] and HCV [4]. Some initial work on these prioritization and trade-off problems can be found in [17].

As part of the decision-making, it should be decided, e.g., 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, e.g., to make estimations both for a short-term and a long-term perspective respectively.

2.3.3 Context model

The context model is a representation of the environment in which the decision is made. 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 an evidence-based knowledge repository (see Section 2.4). An example of a context model representation is presented in [30]. 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 the decision-maker to understand the context in the future and is integrated with an evidence-based knowledge repository described in Section 2.4.

In [8], the context model is structured into the following five dimensions: 1) organization characteristics including organizational structure (management model, business strategy, maturity, capacity, velocity, etc.), 2) product characteristics and all correct contextual information associated with the product before the decision (maturity, technical debt, complexity, openness, certification, etc.), 3) stakeholder characteristics (level of involvement in the decision, experience, competence etc.), 4) development method and technology (development process and methods, practices, environment and tools used etc.) and 5) market and business (type and structure of the market, market trends, ecosystem effects and agreements etc.). Carlson et al. [8] suggested an open hierarchical model for context representation that can be dynamically adjusted and provide various granularity levels. Moreover, understandability and readability are important quality attributes of the context model. 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 [30] to better cover aspects such as the market, ecosystems and also business models.

2.4 Evidence-based knowledge repository

Historical information should be structured so that it is possible to find relevant or similar cases, e.g., similar context, similar prioritized criteria or an interest in the same asset origins. The stored information may facilitate decision-making, and also 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 should 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 publicly available data or data shared between trusted partners that can help support different activities of decision-making.

Information from previous decisions can represent an important support in the decision-making process, at least to avoid errors made in the past. However, 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. Hence, 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.

In our context, the technical implementation of the knowledge repository is realized by taking into account the following observations. Storing decisions about selecting various software assets revolves around the Decision entity of the GRADE taxonomy [9]. The knowledge repository ontology decomposes the decision into the seven key entities on the first level: Environment, Aspect, Value Perspective, Asset Usage, Role Level, Decision Level, and Method Family. 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 [20], OGD eight principles [28] or Web Information Quality assessment [6].

Concretely, the decision-making activities and the knowledge repository have been embedded in a prototype application[2]. The application allows to enter (a subset of) the decision items for documenting a certain decision case, and after they are stored in the repository. The repository is realized as a graph database through Neo4J technology[3] . The choice of graph databases is motivated by their efficiency and scalability for large amounts of data, characterized by flexible structure and arbitrary relationships. Decision items are represented as nodes while relationships are used to trace links between items pertaining to the same decision case. Moreover, the storage and retrieval of information to/from the database is controlled by the application and is completely transparent to the user. In this way, it is possible to keep the required degree of consistency across different decisions and hence to perform queries seeking for similar scenarios.

3 Related work

3.1 Deciding on origin

The research related to selecting between different software asset origins is quite limited. In a recent systematic literature review [3], 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 [10], [11], [22], [23], [33] and [42] 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

[2] https://github.com/orion-research/coach
[3] https://neo4j.com

considered in all the proposed optimization models. The optimization models consider single objective or multiple objectives in the decision model.

The objective in the optimization models proposed in [10], [11], [33] and [42] is to minimize cost under reliability and delivery time constraints. The CODER framework proposed in [10] consists of a decision model based on optimization and accepts UML notations as an input. In [39] and [42], the authors propose an architecture optimization approach based on a swarm intelligence algorithm. The CODER framework [10] is extended in [32] and [33], allowing decision-making as early as requirements are available. Similarly, a general non-linear optimization model is proposed in [11] for the same objective and constraints, i.e. minimizing cost under reliability and time constraints.

Multi-objective optimization models have been proposed in [22] and [23]. A decision model for fault-tolerant systems is proposed in [20] and [22] 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 [23]. The objectives in [23] 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 – [24] and [25] – were identified. The model in [24] provides tool support for requirements clustering to find a cohesive group of requirements using a graph-based model. In [25], Kramer (2011) et al. 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).

3.2 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 takeovers [38], via tactical decisions on which ecosystem to join and support [21] to highly technical decisions on how to realize customer requirements in software. An increasing number of software companies are evolving 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 [32]. 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 centre 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 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 ecosystems [21] and businesses are faced with increased decision complexity and frequency that they need to cope with to succeed with their business endeavours. An example here is decision-making in cloud computing environments for selecting appropriate services from different providers [27].

3.3 Application of process-line in software engineering

Software process-lines has gained more interest recently. However, it has been identified as an immature area and there are few papers reporting the use of process-lines according to a systematic literature review conducted on software process-lines in 2014 [13]. The main use of process-lines is in areas such as software design and architecture with concepts similar to product lines such as variability management [2], product management and project management [14] and [15]. In our study, we use process-lines for the decision-making process, which is a rather new application. Since, process-lines have not been applied in relation to decision-making we do not discuss the above studies in detail.

4 Research methodology

In this section, we discuss the overall research approach used to construct and validate the decision-making checklist and process-line. The relations between the decision-making process-line, checklist and the associated studies are presented in Figure 2.

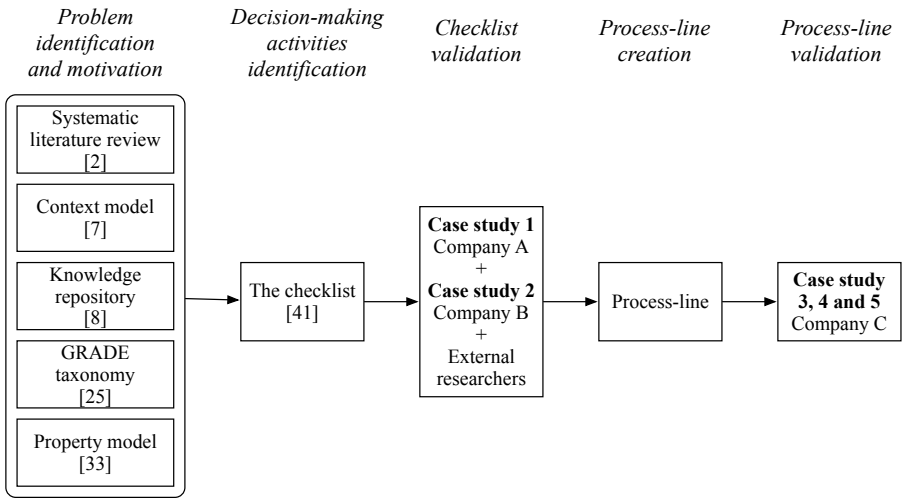

Figure 2: An illustration of how the GRADE taxonomy, knowledge repository and descriptive models provided the starting point for the decision-making checklist and process-line.

The decision-making process-line is formulated based on our previous work [45], which was built upon the GRADE taxonomy [29] and the descriptive models (context model discussed in [8] and property model discussed in [37]) as illustrated in Figure 2. A knowledge repository [9] then also supports the checklist. These studies and a recent literature review [3] on software asset origin selection have influenced and inspired the construction of the checklist. The research team (authors of this paper) reviewed findings from the checklist validated by the industrial experts and external researchers and prioritized goals and challenges to create the focus for the decision-making process-line.

The overall research approach followed in this study is a combination of design science [44] and case study [16]. The tasks in the design science approach and case study are presented in Figure 3 and elaborated in Sections 4.1, 4.2 and 4.3. The design science approach consists of three iterations, which begin with problem investigation, followed by design and finally validation. In addition, case studies in Companies A, B and C were conducted. Note that the checklist and the process-line were not implemented in a real context. In design science, the solution needs to be validated before it is implemented in a real context. Therefore, in this study we validate the process-line, which is an important step before implementing and evaluating the process-line in use. The limitation of the process-line evaluation is discussed in Section 4.4. The details of each of the iterations and tasks followed in each iteration are described in the following subsections.

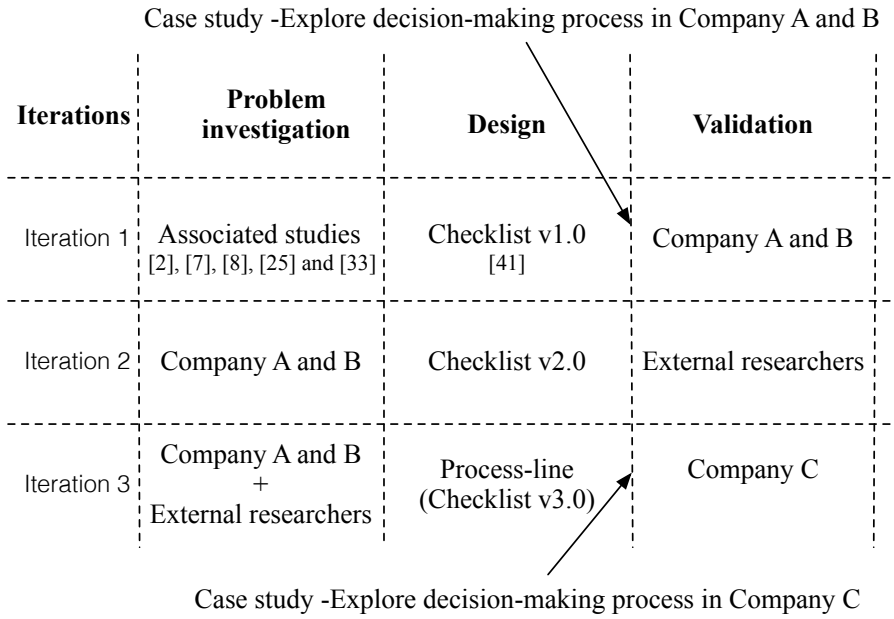

Figure 3: Tasks carried out in the design cycle and case study.

4.1 Iteration 1

Problem investigation: In this step, we identified and formulated the design problem. Based on the template designed by Wieringa [44], we formulate our design problem for iteration 1 as follows: Identify different decision-making activities to be considered, and hence to support decision-makers in selecting suitable software asset origins and components.

Checklist v1.0 design – identification of the decision-making activities: The researchers involved in this study selected the relevant objectives from the associated studies. This helped in identifying the first set of the decision-making activities. The decision-making activities were iteratively discussed via several brainstorming sessions where the researchers discussed opinions, compared possible solutions and scrutinized them. This version of the checklist (v1.0) was published in [45].

Explorative case studies at Companies A and B: We conducted two exploratory case studies in Companies A and B to explore the decision-making process followed in the companies. The case studies were based on semi-structured interviews with stakeholders involved in two decisions about two software components, one decision for each company. The interview questions were rather broad and did not impose any particular decision-making process.

The questionnaire consisted of three parts: introductory, decision-making process details and concluding remarks. The introduction consisted of questions related to general questions about the interviewee, organization, project and product. In the next part, the interviewees were asked to describe their decision-process for each case. If the interviewees did not mention any particular activity in Checklist 1.0, they were specifically asked about it. In order to not influence the research outcome, such responses were differentiated as activities followed but not explicitly mentioned (See Section 6.3 and Figure 6). In the last part, questions related to outcome (positive or negative) of the decision were asked. Three interviews in Company A and four interviews in Company B were conducted. Two researchers were involved in conducting the interviews that were recorded and transcribed. No information of the checklist was discussed in the interview nor any specific questions regarding the checklist activities were asked. Coding rules were established so that the coding was done consistently, the text related to the decision-making activities in [45] should be highlighted and tagged in the transcript with the corresponding activity name. The details of the exploratory case studies in Company A and B are provided in Section 6.

Checklist v1.0 validation: As the solution is not prescriptive, the validation here is not in terms of "effectiveness" but to validate the Checklist v1.0 coverage. The coverage is validated by ensuring that none of the decision-making activities followed by Companies A and B are missing in the checklist. The decision-making processes followed by the companies were mapped to Checklist v1.0 and were sent to the industrial experts, i.e. to the interviewees in Companies A and B after the interview for validation. It is to be noted that the researchers did not intervene and the validation was done independently by the practitioners. The outcome of the validation indicated that the industrial experts perceived that none of the activities

they performed were missing from Checklist v1.0 however, the authors of this study identified the need to refine the Checklist v1.0 activity descriptions so that they are aligned with the activities carried out in Companies A and B. The refinement was carried out in iteration 2.

4.2 Iteration 2

Problem investigation: The validation conducted in the previous iteration indicated a need to add details to the activity descriptions based on the execution of the decision-making activities by Companies A and B. Particularly, the activities needed to be adapted to the process of replacing the component in use and selecting between different components within the same asset origin, which were the two cases in Companies A and B.

Checklist v2.0 design – reformulating Checklist v1.0 into Checklist v2.0: The refinement was done by adding descriptions to include details on how the decision-making activities could be executed. In order to refine the Checklist v1.0, the decision-making processes followed by Companies A and B were considered as an input. The refinement outcome is discussed in Section 7.1.

Checklist v2.0 validation: Checklist v2.0 and the descriptions of the decision-making process followed in the companies were validated by external researchers. The validation was done to ensure that the Checklist v2.0 is capturing the decision-making processes followed in the companies. The aim of the validation was to correct any potential inconsistencies. The validation pointed out some inconsistencies in the representation of the Checklist v2.0 in particular, regarding the order of execution of the activities. Therefore, indicating that the representation of the solution needs to be refined. The refinement of Checklist v2.0 is carried out in iteration 3.

4.3 Iteration 3

Problem investigation: The validation conducted in the previous iteration indicated that the representation of the Checklist v2.0 suggests a prescriptive order of execution, which is misleading, as the goal of this study is not to propose a prescriptive solution. Therefore, the Checklist v2.0 needs to be packaged differently than into a process. The proposal is to formulate the Checklist v2.0 as a process-line which serves as a checklist that supports the decision-makers in building their own decision-making process without any prescribed order of execution.

Process-line design – reformulating Checklist v2.0 into a process-line: The refinement was done to change the sequential representation of the Checklist v2.0 to a process-line which acts as a checklist consisting of a list of possible activities divided into preparation, investigation and decision-making phases. The descriptions of the decision-making activities followed by Companies A and B and the outcomes of the

validation done by external researchers were considered in the formulation of the process-line. The details of the process-line are described in Section 5.

Case studies at Company C: We conducted three exploratory case studies at Company C to explore the decision-making process followed in three different decisions for different products. Semi-structured interviews were used to explore the decision-making processes followed by Company C. Three interviews were conducted, two researchers were involved in conducting the interviews that were recorded and transcribed. No information of the process-line was discussed in the interview nor any specific questions regarding the process-line activities were asked. The same coding rules as mentioned in iteration 1 were followed. The details are provided in Section 6.

Process-line validation: After the interview, the process-line designed in iteration 3 was presented to the industrial experts in Company C for validation. The objective was to validate the applicability of the process-line in building the decision-making process as perceived by the industrial experts.

4.4 Validity discussion

Researcher bias: There is a threat of introducing researcher bias in the design, and validation tasks of the design cycle and in the exploratory case studies.

- **Decision-making process-line and checklist design:** Data triangulation helped to minimize the researcher bias in the design phase. The checklist was constructed by considering inputs from various associated studies [3], [8], [9], [29] and [37]. In addition, the inputs from the decision-making activities followed in the industrial context and external researchers were considered in the formulation of the process-line. The validation using expert opinions of external researchers was done to ensure that all aspects of the associated studies and the facts from the interviews were considered in the process-line. Thereby, individual researcher bias was minimized during the design process.
- **Validation:** The validations were conducted without any intervention, i.e. Checklists v1.0 and v2.0 as well as the process-line were not discussed in the interviews. We believe that this approach minimizes the risk that our interviewees consider some decision-making activities as performed because they are confronted with the checklists and process-line. Since the aim of the process-line is not to suggest one way of making decisions the limitation of not evaluating the effectiveness is not applicable in our study.
- **Exploratory case studies:** The open-ended interview questions minimized the research bias in the data collection. The interviews were recorded, transcribed and coded to capture all the information. Coding rules were established and the coding process was reviewed to avoid inconsistencies and avoid misinterpretation in the coding process. The most relevant stakeholders responsible for the decision in Companies A, B and C were interviewed. Therefore, the threat of not identifying all activities followed in the decision-making process by Companies A, B and C is minimized.

Generalizability: The inability to generalize from the five cases can be perceived as a threat. Flyvberg [16] addresses five case study misunderstandings, one of which is generalizability. He suggests that if knowledge cannot be formally generalized, it does not mean that it cannot be used to accumulate knowledge in a given field or in a society. This indicates that the knowledge can be used to contribute towards generalizing the findings. To contribute towards generalizability, the cases have been described in as much detail as possible without compromising the confidentiality. In addition, the selection of case studies also provides support towards generalizability [16]. The selected cases are diversified in terms of the size of the company, decision goals and the decisions. This is further elaborated in Sections 6.1 and 6.2. We selected decisions (cases) after recommendations from the practitioners to ensure their relevance and representativeness for the study.

5 Process-line for decision-making using a checklist

The main objective of the checklist designed in iterations 1 and 2 as discussed in Sections 4.1 and 4.2 was to provide a set of decision-making activities to select between different software asset origins and components.

The solution proposed in our previous work consists of a set of decision-making activities that are not prescriptive and should be regarded as a checklist rather than a process. This may be captured in a process-line which serves as a checklist, i.e. activities that may be selected (or not) to create different decision-making processes. Process-line has been defined as "a set of software processes with a managed set of characteristics that satisfy the specific needs of a particular organization and that are developed from a common set of core processes (referred as activities in this study) in a prescribed way" [2]. Note that the set of core processes/activities are not prescriptive rather, the development of the process from a common set of core processes is prescriptive.

Each user of the process-line may decide to include or exclude specific activities and hence make informed decisions of which activities to include and exclude. The process-line was designed in iteration 3 as discussed in Section 4.3. Figure 4 depicts the process-line consisting of three phases (preparation, investigation and decision-making) and a repository. Note that the activity numbers mentioned in Figure 4 are identifiers and do not represent the order of execution. The arrows between the three phases represent the dependencies between the phases, and the arrows between the phases and the repository represent the information flow (one- or two-way communication). The input to the repository from the preparation phase is the context information for the specific decision. Based on the context information, similar cases from the repository are retrieved that could be used to execute the other activities in the preparation phase (elaborated further in Section 5.1) as well as in the investigation phase. In the decision-making phase, similar cases and decisions could be retrieved from the repository and after the decision is made, the final decision could be

documented in the repository. The activities within each block are selective activities that could be used. The description of each activity is provided in Section 5.1.

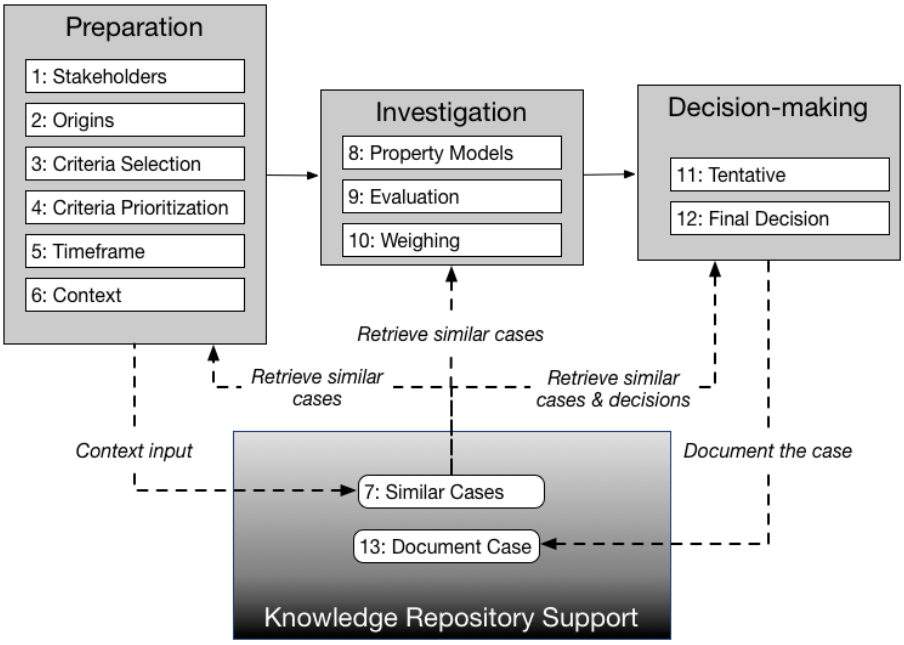

Figure 4: A process-line for decision-making supported by a knowledge repository.

Using the process-line, decision-makers can build their own decision-making process. One possible decision-making process with one possible order, including all activities, is depicted in Figure 5 using the numbering of the recommended activities below. The arrows between the activities indicate the information flow (input and output). However, it should be stressed that the activities do not necessarily have to be followed in the order depicted in Figure 5. Some activities may be perceived as more important than others. However, it has been chosen to present all activities as recommended activities, since the actual usefulness and effectiveness of the different activities and preferred order of the activities may vary from case to case. Thus, the order of the activities should be seen as one possible suitable order. Furthermore, not all activities may be perceived as needed for all decisions. However, it is better to make conscious decisions to not conduct all activities. Thus, the process-line should not be seen as prescriptive; it is intended to make decisions more transparent and to provide support to decision-makers so that important aspects are not overlooked. Furthermore, an evidence-based knowledge repository may not be available in all cases, and hence those activities may not be applicable in all cases. It should also be noted that iterations are expected. They may appear between any activities depending on the specific decision, or the specific circumstances in relation to a decision.

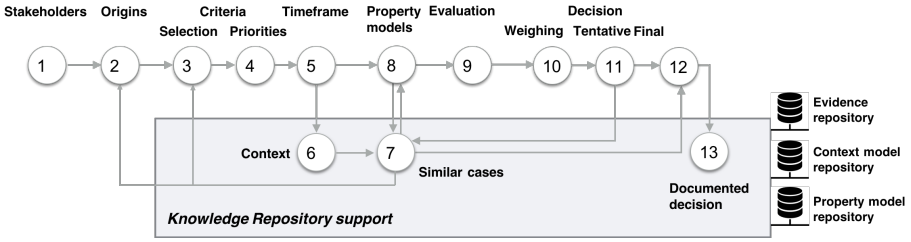

Figure 5: A possible decision-making process including having a knowledge repository.

5.1 The recommended activities in the decision-making process-line are as follows:

Activity 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.

Activity 2: **Screen and evaluate the suitability of the asset origins** – The possible origins for a software asset should be identified. The selection of an asset origin could be for a new component or to replace a component in use with another component by considering two or more asset origins or even from the same origin. For example, there might be a need to replace a component in use developed in-house with an OSS or COTS component. 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, OSS solutions are not an option. Thus, the possible asset origins need to be identified carefully.

Activity 3: **Decide criteria from goals** – Both business and technical criteria are decided based on the goals and targets. The targets should be set so that different asset origins can be evaluated and compared with each other. The goal could be either to choose an asset origin to replace a component in use or choose an asset origin for a new component. In case of replacing a component, the process most likely begins with generic criteria to overcome the existing challenges or shortcomings. For example, a company might want to reduce the time for correcting defects of a component in use, then "reduced time for correcting defects" becomes the generic criterion. Apart from generic criteria there might be additional specific criteria such as "better performance" based on which the selection of an alternative asset origin is made. In most cases, risk needs to be considered as one criterion, since it may differ substantially for different asset origins (in-house, COTS, OSS and outsourcing).

Activity 4: **Decide on priorities of criteria** – It is also important to decide how the criteria should be prioritized, e.g. using AHP [36] or HCV [4]. It may also be

the case that certain stakeholders have more power in a decision, i.e. different stakeholder roles may need to be weighed differently in the prioritization process.

Activity 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 not very good for the long-term architecture of the product. The time aspect is highly relevant when long term maintenance cost is substantial (e.g. when developing in-house) or can be minimized (e.g. by adapting OSS). 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- and long-term, and the trade-off between them has to be agreed upon.

Activity 6: Identify and describe the context – Context information such as product context, organizational context and product context may have an impact of the decision-making process. For example, organizational context such as governance structure of the organization might indicate the types of stakeholders that should be considered in the decision (Activity 1). In addition, depending on the regulations/policies some asset origins might not be possible to use. For example, the organizational policy might not allow any code to be open. Thus, OSS might not be a suitable asset origin (Activity 2). The selection criteria and priorities (Activities 3 and 4) depend on the context of the product. For example, if the product has many users, then one of the criteria would be to choose a component that support many users. Therefore, identifying the context is important in a decision-making process. The context description is important to enable comparison with previous cases internally and externally as well as with the research literature, for this purpose 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 [30]. Independently, it is crucial to capture these aspects to enable identification of similar cases and hence relevant evidence and experiences.

Activity 7: Look for similar cases in a knowledge repository – The identification of similar cases is done using the context information 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 Activity 6) as well as a focus on similar criteria (from Activities 3 and 4) and similar suitable asset origins (Activity 2). Similar cases are identified and studied to identify relevant evidence and experience 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, other asset origins, criteria, property models or decisions have been considered in other similar cases. Thus, it is important to be able to challenge the choices made in the other activities as illustrated in Figure 4 and Figure 5 .

Activity 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 using a knowledge repository, this can be done by retrieving valuable information in terms of what others have used in similar cases (Activity 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.

Activity 9: **Evaluate the criteria, make estimations using the property models** (including expert judgment) **and evaluate the impact of the decision** – The criteria can be either measured or estimated. For example, it is possible to measure the performance of an existing COTS or OSS component. However, the cost to develop a new component can only be estimated. 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 Activity 5. The impact of the decision and the changes that the new decision will bring must also be evaluated. Such evaluations provide better insight into the decision and to perform cost-benefit analysis. The benefits of the decision should outweigh the costs introduced due to the changes needed to implement the decision. Such evaluations are particularly important when an component in use is replaced with a new component from another source (asset origin). The benefits should not only outweigh the costs but also should be better than the component in use. Trade-offs should also be considered. For example, reducing license cost might require additional effort.

Activity 10: **Weigh the estimation results of the selected properties based on the priorities of 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.

Activity 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 Activity 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. The tentative decision may be implemented in the form of prototypes.

Activity 12: **Make a final decision** – This has been the objective of the decision-making process and hence it is a very important activity for the development. It is important that the stakeholders are able to communicate both the actual decision and the rationale for the decision.

Activity 13: **Store the case in the decision knowledge repository** – The case information, including the context information, the criteria used, the stakeholders

involved and the asset origins considered should be carefully documented. This activity 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.

The decision-makers can build their own decision-making processes by including or excluding the set of activities. 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 Validation and Case studies

6.1 Overview of the companies

The validation was done in two phases, first the coverage of the checklist designed in iteration 1 (Section 4.1) was validated in Companies A and B. The validation outcome indicated that the checklist was not missing any decision-making activities however, some reformulation of the activities was needed. The reformulated checklist was then reviewed by external researchers based on which the representation of the checklist was reformulated into a process-line. The coverage of the process-line was validated in Company C. The case studies were conducted at three companies. Due to the strategic nature of this type of decision, the names of the companies are confidential. However, it does not affect the outcomes from the case studies, since the sole purpose is to validate the coverage of the checklist and process-line and not in detail report data from the company. The case studies were designed as described in Section 4. The overview of the companies including the size of the company in terms of number of employees, profile and customer category is presented in Table 1.

Table 1: Overview of the companies

Company	Size	Profile	Customer Category
Company A	~ 100000	Global networking and telecommunications equipment and services	B2B and B2C
Company B	~5000	Global telecommunications	B2B and B2C
Company C	~600	Global provider of cloud-based Business Support Solutions (BSS)	B2B

From Table 1 we can see that the companies are diversified in terms of the size of the companies. The size of the company might affect the decision-making process adopted by the companies. Therefore, identifying the decision-making activities

followed by diversified companies will help in validating the coverage of the process-line. In addition, the companies have diversified customer category therefore, the process-line can be validated in different contexts.

6.2 Overview of the cases

Case 1 description at Company A – In this case the company wanted to replace a component already in use. Due to the existing problems such as lack of timely support from suppliers and substantial license costs, a need to replace the component was identified. In-house and outsourcing were not suitable options due to the immense development effort required. Based on the positive result of a proof-of-concept, and success stories of using the same OSS component by other companies, the decision to use the OSS component was made. The stakeholders involved in the decision, i.e. the architect, line manager and system manager were interviewed. The architect was the decision initiator, contributor and was also involved in the final decision along with the line manager and system manager.

Case 2 description at Company B – The hardware for the products is partly internally developed and partly purchased from suppliers. The software that runs on this hardware is based on an OSS platform with extensive adaptations and unique functionality developed in-house. The stakeholders involved in the decision were interviewed, i.e. the product experience planner who was the decider, project scope manager who contributed in the decision with the required information and knowledge and personnel from the legal and sourcing department who was the influencer in the decision. The case at Company B was a decision between two COTS suppliers. The decision-makers had already selected COTS as the asset origin. However, other options were considered, but they were ruled out very early in the process, and hence the actual decision-making became a decision between two different COTS components. Considering OSS components, it became apparent that OSS could not provide the necessary functionality and support. The in-house and outsourcing options were the least favourable already from the beginning due to the immense effort required to provide the solution comparable with the currently available COTS components. Thus, the focus was set on deciding between two COTS components. The decision process took several months and was heavily influenced by technical investigations led by so-called product experience planners. Top management used their technical recommendations when making the decision. The sourcing department at the company negotiated the contract details with the selected COTS supplier. Legal aspects (have control over the patent portfolio and enough resources to fight in the court if some other company initiates or pursues legal procedures) were significant in this case and considered next to technical functionality and quality aspects. Finally, the interaction process between the suppliers and Company B, as well as how they supported the components took a substantial role in the decision.

Case 3, 4 and 5 descriptions at Company C – The architects were mainly involved in the decision. The strategic decision as a company was to use as much OSS as

possible. Therefore, there was no approval needed for using an OSS component. The decision was mainly focused on fit to functionality criterion and the chief technical officer agreed to decisions proposed by the architects. Hence, we interviewed only the architects for exploring the cases in Company C. In all three cases, OSS was the preferred choice of asset origin. In Case 3, a decision to use a new OSS plugin was made to improve the functionality of an OSS component already in use. In Case 4, the decision was to replace the COTS deployment framework in use with an OSS deployment framework. In Case 5, the decision was between changing to a new OSS component, or continue to use the current OSS component. Therefore, all three cases were distinct since, Case 3 was a new decision, Case 4 was to change from one asset origin to another, and Case 5 was a decision between two components from the same asset origin. The overview of the cases in terms of the goals, final decision and outcome is presented in Table 2.

Table 2: Overview of the cases with respect to the goal of the decision, the final decision made and the decision outcome

Company	Case	Goal	Final decision	Decision outcome
Company A	Case 1	Replace component in use	COTS→OSS	Positive
Company B	Case 2	Choose a COTS vendor for building a new component	COTS vendor chosen	Positive
Company C	Case 3	Add new functionality for a component in use	OSS component chosen	Not positive
	Case 4	Change the deployment framework component	COTS→OSS	Positive
	Case 5	Replace component in use	No replacement	Positive

The goal of three decisions (Cases 1, 4 and 5 in Table 2) were to replace the current COTS component with a new OSS component. In Case 1, the decision was to switch to the new component however, in Case 5 after the investigation the decision was to continue using the component in use. In Case 5, the component in use was perceived to be a sub-optimal solution in terms of functionality fitness. Therefore, the goal was to replace the component in use with a component having better functionality. However, the team was not familiar with the new component and it required additional training or hiring trained personnel. The additional cost and time to train or hire new personnel prevented the decision-makers to switch to the new component. In retrospect, the decision-makers perceived the decision to be positive as switching to the new component would increase the cost (increased head count/training) and delivery time. Therefore, maintaining the cost and timeliness received more importance than functionality fitness (this is further elaborated in Section 6.5-Activity 4). In conclusion, it was considered better to keep the component although it was not the original goal. It was perceived as positive to have done the evaluation, although it did not result in any change. In Case 4, the decision was to change the COTS framework in use to a new OSS framework.

In Case 2, the goal was to choose a new COTS component. Even though the other asset origins were usually considered in the decision-making process by Company B. For this component, the other asset origins did not have any suitable solution. Therefore, the decision was mainly to choose a COTS vendor.

6.3 Overview of the decision-making process followed by the companies

The activities followed or not followed in the companies are illustrated in Figure 6 and elaborated in Sections 6.4 and 6.5. In addition, we depict how the activities were mentioned (explicitly or not) in the interviews. The total number of activities followed by the companies is indicated on the x axis and the frequency of the activities followed in the cases is indicated on the y axis. The circle symbol ● in Figure 6 represents the coverage of the activities followed by the companies. The circle with diamond inside ⧁ indicates that the activities did not come up in the interviews when asked about the decision-making process. However, when asked about the specific activities they mentioned that they followed the activities. The diamond symbol ◆ indicates that the activities were followed however, either partially or not for its intended purpose.

Figure 6 represents the coverage of the activities followed by the companies using the process-line (and not checklist), as the process-line is the final solution of our study. The cross on the dotted arrow lines indicates that there was no information flow between the phases and the repository. The number of activities followed by the cases is similar. However, Case 3 followed the least number of activities and it was also the decision that was perceived to be negative as shown in Table 2. Activity 5: Timeframe that was not followed was perceived to be an important activity that resulted in having a negative outcome.

Seven activities of the process-line were followed in all five decisions as shown in Figure 6. The activities related to the repository (i.e. 7: Similar cases and 13: Document case) were the least followed activities. In addition, the documentation was done however, not for reusing in future decisions. Success stories from similar cases were considered in the decisions however, the success stories were based on the decision-makers' knowledge and were not retrieved from a repository as they were not documented.

In the preparation phase, 4: Criteria prioritization and 5: Timeframe activities were not followed in all the cases. Since criteria are not always prioritized, 10: Weighing activity is also not followed in all the cases. Tentative decisions were made in the three out of four cases where the decision was made to take an action.

Overall, the practitioners did not perceive any decision-making activities to be missing in the process-line. The details of how the activities are followed by the companies are discussed in Sections 6.4 and 6.5.

6.4 Detailed description of the decision-making activities followed in *all* the cases

Activities 1, 2, 3, 6, 8, 9 and 12 were followed in all the cases. The descriptions on how these activities were followed is provided below.

Activity 1: Identify stakeholders to be involved in the decision -

Table 3 provides the stakeholders involved in the decision committee and their roles for each case. In most cases the architects played an important role in the decisions. Except for Case 2, the architects were the initiators, contributors and deciders as mentioned in

Table 3. In Case 1, the budget approval was required from the design centre head and product manager. In Case 2, the legal and sourcing department was responsible for negotiating and signing the contract with the component suppliers. The number of roles involved in the decision varies from case to case.

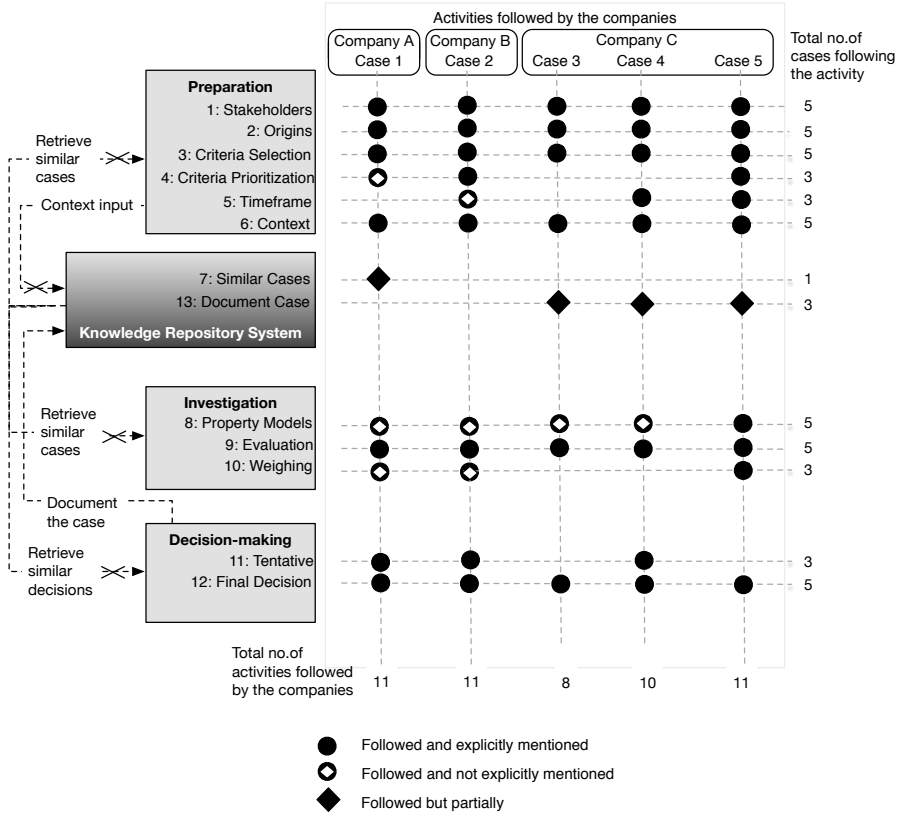

Figure 6: The process-line activity coverage in related to the decision-making process followed by the companies.

Table 3: Stakeholders involved in the decision committee and their roles in the decision.

Case no.	Decision committee	Decision-making role
Case 1	Design centre head and product manager	Budget approver
	Line and system manager	Decider
	Architect	Decision initiator and contributor and decider
Case 2	Product experience planner and project manager	Deciders
	Project scope manager, developers and testers	Decision contributors
	Legal and Sourcing department	Decision influencers
	Component providers	Decision influencers
Case 3, 4 and 5	Chief technical officer	Decider
	Architect	Decision initiator and contributor and decider

Figure 7 provides the number of roles involved in the decision. Case 2 has the highest number of stakeholders with different roles involved in the decision. As the decision was based on selecting between different COTS vendors, involving COTS vendors in the decision and negotiating the legal aspects were part of the decision. In Case 1, there was no stakeholder responsible for discussing legal aspects however, it was realized as an important role in the decision-making and was considered for future decisions. Since, Cases 3, 4 and 5 were conducted in a comparatively small company, the decision committee consisted of only two members. The decision was made by the architect and chief technical officer agreed to the decision in all three cases in Company C.

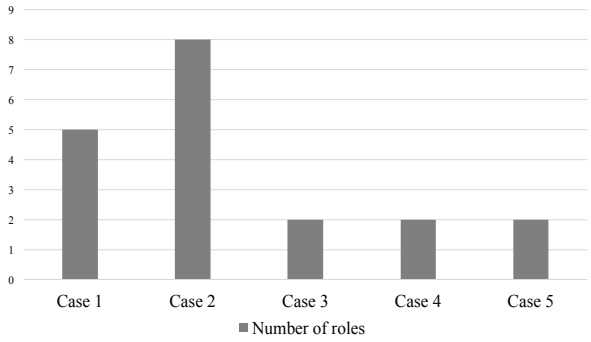

Figure 7: Number of stakeholder roles involved in each decision.

The number of stakeholders involved, their roles and the personality of the stakeholders affect the inclusion or exclusion of the decision-making activities and the time spent on each activity as perceived as one of the architect interviewed. For example, some stakeholders might need more information to approve a decision.

Activity 2: Screen and evaluate suitability of the asset origins – In all cases OSS was the preferred origin and only two origins were considered as shown in Figure 8. In case 2, various OSS components were available and evaluated regularly but due to lack of full support for text input in over 60 languages they could not be considered. COTS remained as the only option with two potential COTS suppliers. In-house and outsourcing were not considered as suitable alternatives due to the extensive effort (100+ developers working for several months) in Cases 1 and 2. Company C has a strategy to use OSS as the first choice. Developing the component was considered as the last option.

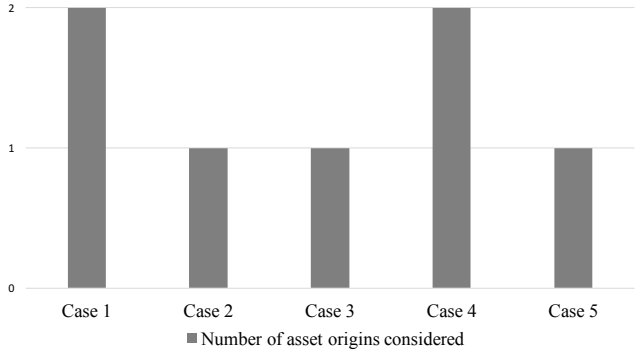

Figure 8: Number of asset origins involved in each decision.

Activity 3: Decide criteria from goals – The number of criteria considered is case dependent as shown in Figure 9. In Case 1, the goal was to replace the component in use with a new component that overcomes the problems identified with the component in use. The general criteria selected based on this goal were: maintenance effort, technical support, costs, distribution, fault tolerance and fitness in terms of functionality. Apart from these criteria, specific criteria related to the actual decision

to select the component were: scalability (architecture for scaling), security, performance for big data, replication, source code availability for defect fixing, maturity of the OSS community and the number of users using the component.

In Case 2, the criteria used in the decision were: support for multiple languages, support for defect fixes, ability to survive "patent fights", reduced costs, better user experience, possibility to add customized user experience elements (like buttons), and technical features (next word prediction and text recognition from sloppy writing).

In Cases 3, 4 and 5, functionality fit was the criterion to select the component. In case 5, the skills needed to maintain the new OSS component was also considered as a criterion in addition to functionality fit.

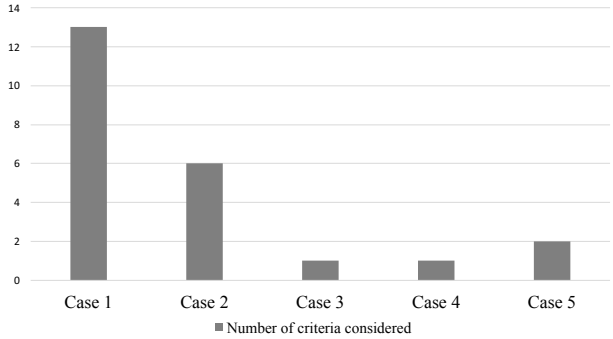

Figure 9: Number of criteria considered in each decision.

Activity 6: Identify and describe the context – In Case 1, organizational, product and supplier contexts were considered in the decision. The governance structure of Company A was such that all relevant stakeholders should be involved in the decision. The product context such as complexity, criticality, the number of customers, and subscribers or users using the product was considered. The context of the supplier community such as maturity and number of recent releases was considered.

In Case 2, the context is implicitly discussed but not modelled. The company does not have enough knowledge within text recognition to internally develop the component and *all* competitors use external suppliers. The text recognition component is important for the product offering but not considered as the main driver for purchasing the product.

In Cases 3, 4 and 5, the context was considered important. For each decision, the following question is answered – "Are we the organization that can support the decision (i.e. do we have the required skills or support required to implement the decision?) or can we become that organization?"

Activity 8: Decide on property models to use – In all cases, expert opinion was used to evaluate the criteria. Deciding on how to evaluate/estimate the criteria did not come up in the interview but when asked about the process the interviewees mentioned that they followed the activity. The evaluation was based on stakeholders' opinions, experience and knowledge.

Activity 9: Evaluate the criteria, make estimations using the property models and evaluate the impact of the decision – In Cases 1, 4 and 5, where the goal was to change or replace the component in use, the experts were aware of the attributes such as time, cost and effort for the component already in use. The evaluation was done using expert opinion method to find alternatives that improve the existing values and not make it worse. In Case 3, the functionality of the plugin was evaluated by the decision-makers by testing the functionality of the plugin.

Company B (Case 2) had the following properties: level of support (measured by checking if the supplier has regular meetings with the procurer and how quickly they react for issue requests), and code quality was important but only externally measured by checking how many crashes the component had. Internal code quality could not be measured. Support for multiple languages was measured by checking how good the functionality and the next word prediction is in several languages. User testing and prototype-based evaluations were heavily used in this case.

In Case 1 the scalability and redundancy of the OSS component were evaluated. The impact of the decision in terms of risks was also evaluated. For example, if the OSS component did not evolve, then the need to modify the component is identified. In order to modify the component, the source code needs to be fully understood. This involves a learning curve and results in increased head count. The patents and proprietary rights were also evaluated.

The number of additional resources required to use the new component was estimated in Cases 1 and 5. The trade-off between the risk of having to contribute to the OSS community and monetary value gained by not having to pay for external support was evaluated in Case 1. Similarly, in Case 5, the trade-off between the additional time and cost to train employees or hire new employees and its fit to functionality was evaluated.

Activity 12: Make a final decision - A final decision was made in all cases. A COTS component was replaced with an OSS component in Cases 1 and 4. In Case 2, a decision to acquire a COTS component from a COTS vendor and a new OSS plugin component in Case 3 was made. In Case 5, it was decided to continue using the component in use and not change to a new component.

6.5 Detailed description of the decision-making activities followed in a sub-set of the cases

Activities 4, 5, 7, 10, 11 and 13 were followed only in some of the cases. The descriptions on how these activities were followed is provided below.

Activity 4: Decide on priorities of criteria – In case 1, prioritization of the criteria was not mentioned explicitly in the interviews. However, the criteria driven from current issues i.e. reduced maintenance time and lower license cost was perceived more important than finding the best possible component replacement based on criteria such as scalability. Therefore, timely support and cost of license were prioritized over other criteria.

In Case 2, the selection criteria were prioritized. Support for multiple languages and support for maintenance and defect fixes received the highest priorities. Technical support and cost were prioritized over accessibility of the source code. Code quality and intended user experience were also important.

In Cases 3 and 4, there was no prioritization as only one criterion was considered. In Case 5, the skills required to maintain the OSS component were considered more important than functionality fit. Mismatch in the required and available expertise of the current team was a bigger risk than the benefit of functionality fitness. If the people need to be trained to use the new functionality of the OSS component or new people need to be hired, then the OSS component was not considered as a good alternative.

Activity 5: Decide on how to handle the time aspect - The short- and long-term impact of the solution was not explicitly mentioned in the interviews. In Case 2, the interviewees talked about long-term implications of the decision (support and patent fights and support for maintenance over 2-5 years) versus short-term (providing bug fixes on time and quick updated for the upcoming product releases).

In Cases 4 and 5, the long-term implications of the decision were considered. In addition, there was no need for an immediate solution as the current solution acted as a backup.

However, in Case 1 the patent issues were realized much later after the decision was taken, which indicates that patent issues have a long-term impact. In addition, in Case 3, the compatibility issues were perceived to have a long-term impact as the stakeholders realized that the updates between the component and the plugin were not always synced.

Activity 7: Look for similar cases in a knowledge repository – In Case 1, the decisions were not documented and similar cases were not retrieved from the repository. However, the same OSS component was previously used by the company successfully, also other large companies used the same OSS component. Although the criteria in the previous decisions were not reused the successful implementation of the component in previous cases increased the confidence in making the decision.

In Case 2, it was not perceived that there were similar cases. There was no component that offered a similar level of functionality, quality aspects and long-term support

from the supplier. However, "similar" could be interpreted a little broader, i.e. to learn from similar cases in the company and not only in the current product-line. Thus, implicitly the previous decision to select between COTS providers could be considered as similar. In a broader sense, any COTS component purchase creates a question about later maintenance and bug fixes and also support for additional functionality.

In Cases 3, 4 and 5, there were no similar decisions made as perceived by the interviewees.

Activity 10: Weigh the estimation results of the selected properties based on the priorities of criteria – Weighing activity was conducted in Cases 1, 2 and 5. It was explicitly mentioned in Case 5 and in Case 1 and 2 the interviewees mentioned it when asked about the weighing activity.

As discussed in Activity 4 followed in Case 1, reduced maintenance time and lower license cost were prioritized criteria. These criteria were traded-off with the additional people (head count) needed to build a local support team in order to replace the technical support provided by COTS vendor. The additional head count cost was traded-off with the COTS vendor license cost. At the end of the weighing, it was perceived that adding additional people to support the maintenance locally was better (faster and more transparent) than relying on the COTS vendor. In addition, the cost of adding people was perceived to be lower than the COTS vendor license cost.

In Case 5, the benefit of having better functionality was trade-off with the addition time and cost in training or hiring the people. At the end of the weighing, it was perceived that using the current component was better (workable solution) than adding additional people to support the maintenance of the component.

In Case 2, the company implicitly compared the service level agreements between the COTS providers. Cost was weighed against the other decision criteria in the final phase and at the end cost was *not* a factor since both offerings had the same price. Initially, the prices were different, but at the end the price was the same. In other words, one supplier lowered their price to ensure that the price as such was not a decisive factor. Thus, the level of support and the "relationship with the supplier" were the key criteria for the decision.

Activity 11: Make a tentative decision - A proof-of-concept was created and the tentative selection to use the component on a smaller product in Cases 1 and 4 was made. In addition, in Case 2, a tentative decision was made to narrow it down to two options. These two options were evaluated after which one of the components came out as better with respect to some aspects. The key aspects evaluated here were the performance levels in different languages and possible differences in that and how they will impact the expected user experience. As a part of the proof-of-concept, the company also requested several adaptations from suppliers and experienced their response to extension requests.

Activity 13: Store the case in the knowledge repository - The decision was not documented in the classical sense in Cases 1 and 2. It turned into the knowledge about what the decision-makers selected and why. The minutes of the meetings with the decision committee were stored in Cases 2, 4 and 5. However, they were not stored for the purpose of reusing the knowledge for future decisions.

Finally, it should be noted that no new activities were identified through the interviews and discussions. Different asset origins were considered as viable options by Company B. However, only one asset origin was a suitable option and the decision-making was mainly to select a component from the asset origin. It is noteworthy that the decision-making process-line fits well with the decision to select between different asset origins and to select between different suppliers (two components from the same asset origin, i.e. COTS in this case).

6.6 Discussion and implications from the cases

It is noteworthy that the decision-making process-line presented in Section 5 was found useful for selecting between components of the same asset origin. This is validated through the component selection at Company B. Selecting between asset origins is primarily a strategic decision, while the selection between components is mostly performed on a tactical level [1]. The process-line presented in this paper is likely to support both strategic as well as tactical level decisions.

The decision-making process followed by the companies is case dependent as we see in Section 6 and Figure 6. Thus, the process-line requires substantial flexibility not only in ***how*** the activities are conducted but ***what*** activities are conducted. The ***order*** of the activities (which activity comes first and which second) is also flexible. However, there is sequential order of execution between the preparatory, investigation and decision/making phases.

Although the activities executed were similar there were some variations in the way the processes were executed. This led us to identify the possible variation in the execution of all the activities in the process-line based on the cases. In addition, the execution of some activities could be dependent on the execution of other activities. The variations and dependencies are presented in Figure 10.

Each activity execution could vary, for example, as shown in the first box in Figure 10 a decision-maker could identify all the relevant stakeholders at the beginning of the process or only a sub-set of the stakeholders and add more stakeholders at later stages whenever required. There were no variations identified based on the interviews for the activity to identify and evaluate the context, criteria and the weighing activity. Similarly, no variations in the tentative decision and the final decision were identified in the interviews. The “weighing” activity depends on the execution of the “criteria prioritization” and “evaluate” activities. If the criteria are prioritized and evaluated (represented by AND gate in Figure 10), then the weighing activity can be executed

by weighing the priority of the criteria together with the estimated results obtained from the evaluation.

The execution of the process-line could be iterative for example, a sub-set of stakeholders could initiate the decision and in each iteration more stakeholders could be added. After each iteration, a decision if more investigation is needed to reach to a decision could be made. If further investigation is needed, then activities could be executed again. This finding is from Case 5, where the decision was made to not continue the investigation and use the component in use without any replacement.

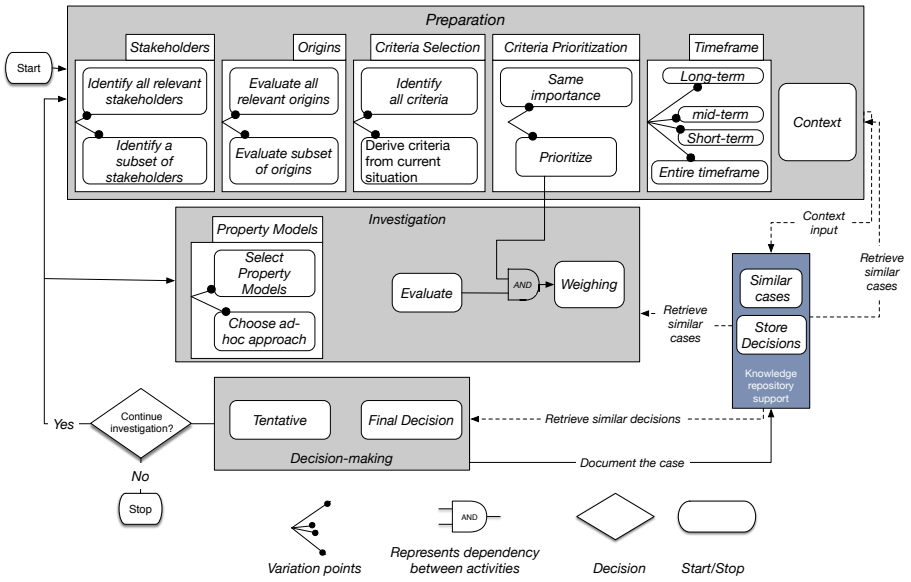

Figure 10: The process-line for decision-making including variations in activity execution.

The process-line in Figure 4 and the process-line with variations in execution in Figure 10 were shown to the interviewees in Company C. The interviewees' perceived the process-line in Figure 4 to be applicable from a managerial perspective and the process-line with variations in Figure 10 was perceived to be applicable from a technical perspective. In addition, having a process-line with a set of activities was perceived to be useful for the decision when the time is short as the process of selecting the decision-making activities to include in their decision-making process could be faster. On the other hand, having more time would allow the decision-makers to use the process-line and conduct the activities more thoroughly according to another interviewee. Another use stated by the interviewees was that the process-line is applicable when there is a problem in the current decision/making process and they are looking for ways to improve it. However, a wider survey is needed to capture the applicability of the process-line.

7 General discussion and validation implications

7.1 Checklist v1.0 validation and refinement into Checklist v2.0

The Checklist v1.0 validation conducted in Companies A and B, resulted in the formulation of Checklist v2.0. The following changes were made to the activity descriptions based on the validation.

- Evaluating the suitability of only one origin is added to the description of the "Screen and evaluate the suitability of the asset origins" activity.
- In "Decide criteria from goals" activity, we made a distinction between criteria used to reject an asset origin or replace an existing component, and the actual criteria used to make the decision.
- The context description activity is refined to identify the context information that could impact the decision in addition to describing the context information.
- As some criteria can be measured and evaluated, we reformulated the activity to estimate the criteria to include evaluations and not only estimations as described in the original proposal [45]. For example, scalability and functionality of an existing component can be evaluated. However, time to build a new component may potentially be estimated using some form of cost estimation model. In addition, the impact of the decision, cost-benefit analysis and changes that need to be made in order to incorporate the new decision are added to this activity.
- We also noticed that the tentative decision implemented in the form of prototypes and proof-of-concepts are more reliable and the decision-makers are more confident about their decision when they implement prototypes and proof-of-concepts. Thus, we have added these types of evaluations to the tentative decisions.

In both cases the decision was not documented. Documenting the decision and storing it in a repository allows information to be reused from previous similar decisions. As we can see a shift in the way software is development, reusing existing components is becoming increasingly common. Hence, as more such decisions to select an asset origin will be made, documenting such decisions will create a rich repository that can be used for further decisions.

7.2 Checklist v2.0 validation and refinement into process-line (Checklist v3.0)

We mainly changed the representation of the solution to eliminate a specific order between each activity. The following changes have been made –

- The activities are divided into three phases: preparation, investigation and decision-making.
- The information flow between the phases and repository is made explicit.

- The variation in execution and dependencies between activities and phases is also represented as shown in Figure 10.

8 Summary and further work

The development of today's software products, systems and services is far from trivial and often result in rather complex decision support models and decision-making processes. The decisions of choosing software components from different origins, such as in-house development vs. COTS, OSS and outsourcing, are most often strategic and have significant consequences on competitiveness. The decision-making process-line presented in this paper provides a starting point for supporting such decisions and addresses the research gap identified in a recent systematic literature review [3].

The presented decision-making process-line for selection of software asset origins can be applied for both B2B and B2C contexts as long as 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 further evaluate the generalizability of the decision-making process-line to support different levels of decisions, i.e. strategic decisions (to select between different asset origins), tactical decisions (to select between different suppliers or different components) and selection between different services. In addition, we plan to survey several business scenarios that involve diverse business models, asset origins, company characteristics and ecosystem participation models. We aim at clearly outlining short- and long-term consequences of each variation of activity execution. These should form guidelines that software business practitioners may use when considering various sourcing options.

We plan to implement the process-line in practice and evaluate the implementation as part of future work. The applicability of the process-line is important to evaluate through a large survey. A survey to collect the feedback on the applicability of the process-line is planned as future work. Moreover, we plan to expand our research on the evidence-based knowledge repository in the following ways: to create the first implementation of a repository that can support decision-makers and to create tool support for executing the process and storing the data in the knowledge-based repository. Finally, we plan to conduct an empirical study to evaluate the use of presented decision-making process-line to formulate specific decision-making processes 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. We would also like to thank our colleagues in the ORION project for fruitful discussions. Finally, we would like to thank the reviewers for valuable input 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] Armbrust, O., Katahira, M., Miyamoto, Y., Münch, J., Nakao, H., and Ocampo, A. (2009). Scoping software process lines. *Software Process: Improvement and Practice*, *14*(3), 181-197.

[3] Badampudi, D., Wohlin, C. Petersen, K.: Software Component Decision-making: In-house, OSS, COTS or Outsourcing - A Systematic Literature Review, *Journal of Systems and Software*, Vol. 121, No. 11, November, pp. 105-124, 2016.

[4] 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

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

[6] 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

[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] Carlson, J., Papatheocharous, E., & Petersen, K. (2016, April). A Context Model for Architectural Decision Support. In IEEE Proceedings of the *1st International Workshop on Decision Making in Software ARCHitecture (MARCH) (2016)* 9-15)

[9] 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)

[10] 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

[11] 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

[12] 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)

[13] de Carvalho, D. D., Chagas, L. F., Lima, A. M., & Reis, C. A. L. (2014, November). Software process lines: A systematic literature review. In *International Conference on Software Process Improvement and Capability Determination* (pp. 118-130). Springer.

[14] de Carvalho, D. D., Chagas, L. F., & Reis, C. A. L. (2014, September). Definition of Software Process Lines for Integration of Scrum and CMMI. In *Computing Conference (CLEI), 2014 XL Latin American* (pp. 1-12). IEEE.

[15] Aleixo, F. A., Freire, M. A., dos Santos, W. C., & Kulesza, U. (2010, June). A Model-driven Approach to Managing and Customizing Software Process Variabilities. In *ICEIS (3)* (pp. 92-100).

[16] Flyvbjerg B., Five misunderstandings about case-study research, in: Qualitative Research Practice, SAGE, 2007, pp. 390–404.

[17] Franke, U.: Towards Preference Elicitation for Trade-Offs between Non-Functional Properties. In *Enterprise Distributed Object Computing Conference (EDOC), 2016 IEEE 20th International* (pp. 89-98). IEEE. DOI: 10.1109/EDOC.2016.7579389

[18] 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

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

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

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

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

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

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

[29] 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)

[30] 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

[31] Petersen K., Badampudi D., Shah S., Wnuk K., Gorschek T., Papatheocharous E., Axelsson J., Sentilles S., Crnkovic I. and Cicchetti A.: Choosing Component Origins for Software Intensive Systems: In-house, COTS, OSS or Outsourcing?--A Case Survey. IEEE Transactions on Software Engineering. Vol. PP, No. 99, March, pp.1-1, 2017.

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

[33] 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

[34] Regnell, B. and Brinkkemper, S.: Market-driven Requirements Engineering for Software Products, in *Engineering and Managing Software Requirements* (editors. A. Aurum and C. Wohlin), Springer (2005) 287-308.

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

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

[37] Sentilles, S., Papatheocharous, E., Ciccozzi, F. and Petersen, K.: A Property Model Ontology. *Proceedings of the 42nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA*) (2016).

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

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

[40] 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

[41] 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

[42] 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

[43] 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

[44] Wieringa, R. J.: *Design science methodology for information systems and software engineering*. Springer (2014)

[45] Wohlin, C., Wnuk, K., Smite, D., Franke, U., Badampudi, D. and Cicchetti, A.: Supporting Strategic Decision-making for Selection of Software Assets. *Proceedings of the 7th International Conference on Software Business* (ICSOB), (2016) 1-15, Springer LNBIP 240.

[46] Wnuk, K.: Involving relevant stakeholders into the decision process about software components. In 2nd MARCH Workshop at WICSA and CompArch 2017, April 6, Gothenburg, Sweden) (2017).