Choosing Component Origins for Software Intensive Systems: In-House, COTS, OSS or Outsourcing?—A Case Survey

Kai Petersen, Deepika Badampudi, Syed Muhammad Ali Shah, Krzysztof Wnuk, Tony Gorschek, Efi Papatheocharous, Jakob Axelsson, *Senior Member, IEEE*, Séverine Sentilles, Ivica Crnkovic, and Antonio Cicchetti

Abstract—The choice of which software component to use influences the success of a software system. Only a few empirical studies investigate how the choice of components is conducted in industrial practice. This is important to understand to tailor research solutions to the needs of the industry. Existing studies focus on the choice for off-the-shelf (OTS) components. It is, however, also important to understand the implications of the choice of alternative component sourcing options (CSOs), such as outsourcing versus the use of OTS. Previous research has shown that the choice has major implications on the development process as well as on the ability to evolve the system. The objective of this study is to explore how decision making took place in industry to choose among CSOs. Overall, 22 industrial cases have been studied through a case survey. The results show that the solutions specifically for CSO decisions are deterministic and based on optimization approaches. The non-deterministic solutions proposed for architectural group decision making appear to suit the CSO decision making in industry better. Interestingly, the final decision was perceived negatively in nine cases and positively in seven cases, while in the remaining cases it was perceived as neither positive nor negative.

Index Terms—Decision making, in-house, COTS, OSS, outsourcing

1 Introduction

Architectural decision making distinguishes different types of decisions, Kruchten [1] divided decisions into structural and behavioral decisions. Structural decisions are concerned with the elements and their interfaces of architectural components. The choice of which components to use in a software-intensive system is thus a structural architecture decision. The choice of the right components is an important factor for the success of a system developed with Off-the-Shelf (OTS) components [2], which includes Components off the shelf (COTS) and Open Source Systems (OSS). Currently, there is a lack of empirical evidence and understanding how practice selects OTS components, as pointed out by Ayala et al. [2] *"to improve OTS component selection practices; the research community must understand what the actual industrial OTS selection practices are in order to envisage more realistic and effective solutions"*. Ayala et al. [2] and other researchers [3], [4] have provided insights of how practice selects components. Their focus was OTS development. However, when choosing a component another consideration is the source of the component, leading to the following question: Should the component be developed in-house, should an OTS component be squired, or should the development of the component be outsourced?

This question is of high relevance as different component sourcing options (CSOs) have distinct characteristics that have to be taken into consideration. COTS based development implies a lack of control over evolution and the quality of the component (cf. Torchiano and Morisio [5]). Torchiano and Morisio also point out that the development with COTS also has an implication on the development process, which needs to focus on the combination and testing of the components, as well as dealing with the evolution of the components that is not in the control of the integrator. When choosing outsourcing the issue of the lack of control does not arise. Though, specific issues to outsourcing may materialize. In particular distances (cultural, temporal, and geographical) have an effect on the development process [6], which has to be adjusted to cope with the distances [7]. In addition, software architects are challenged in mentoring and facilitating learning at the outsourced organization, and have to guard the integrity of the architecture during the learning period [8]. Given the significance of the potential effects on the organization making a choice for a CSO, it is

- *K. Petersen is with the Department of Software Engineering, Blekinge Institute of Technology, Campus Gräsvik, Karlskrona 371 41, Sweden. E-mail: kai.petersen@bth.se.*
- *D. Badampudi, K. Wnuk, and T. Gorschek are with the Blekinge Institute of Technology, Campus Gräsvik, Karlskrona 371 41, Sweden. E-mail: {deepika.badampudi, krzysztof.wnuk, tony.gorschek}@bth.se.*
- *S. Muhammad Ali Shah, E. Papatheocharous, and J. Axelsson are with SICS Swedish ICT AB, Kista SE-164 51, Sweden. E-mail: {syed.shah, efi.papatheocharous, jakob.axelsson}@sics.se.*
- *S. Sentilles and A. Cicchetti are with Mälardalen University, Västerås 721 23, Sweden. E-mail: {severine.sentilles, antonio.cicchetti}@mdh.se.*
- *I. Crnkovic is with Chalmers, Gothenberg 412 58, Sweden. E-mail: ivica.crnkovic@mdh.se.*

Manuscript received 25 May 2016; revised 15 Feb. 2017; accepted 27 Feb. 2017. Date of publication 2 Mar. 2017; date of current version 21 Mar. 2018. Recommended for acceptance by M. Di Penta.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2017.2677909

important how organizations choose between them. The evidence of how decisions are made for CSOs is very limited with only two industrial case studies in the area [9].

The need to better understand decision making in industry for OTS selection [2] and CSO selection[9] has been highlighted. Furthermore, the need for evaluating the outcome of the decision is important [10]. To address the above mentioned needs we provide an analysis of 22 cases about *how decision making took place when choosing among CSOs for adding components in industrial software-intensive systems*. We identified the CSOs considered, the stakeholders involved, the criteria considered, and the actual decision taken. Furthermore, reflections on the decision reached were obtained, such as whether the *"right"* decision was reached.

We considered four CSOs: In-house developed components, components off-the-shelf, open source components, and outsourced development of components, which are defined as follows:

- In-house: The component is developed within the same company. Badampudi et al. [9] highlight that it is still considered in-house development when the development is distributed, as long as it takes place within the company.
- COTS: This option stands for "components off-the-shelf" or "commercial-off-the-shelf", which are already developed (pre-built) and for which the source code is usually not available to the buyer.
- OSS: Open source components are also pre-built, but the source code is available. OSS components are commonly built by a community.
- Outsource: Another company is developing the component and is given the contract by the company wanting to obtain the component.

The research method used was case survey [11]. The cases were gathered in the context of the ORION project.[1] The results add to the limited empirical knowledge (cf. [2]) of how component decisions have been made in industrial practice, in particular the choice between CSOs. The cases were collected based on the researchers' experiences in industrial settings and based on interviews with experts from companies. Overall, 22 cases are included in this case survey.

The remainder of the paper is structured as follows. Section 2 presents the related work. Section 3 describes the research method used. Section 4 explains the results, followed by their discussion (Section 5). Section 6 concludes the paper.

2 Related Work

We first present the decision making problem targeted in the paper by characterizing the alternatives for the decision (In-house, OSS, COTS, and Outsourcing) based on the knowledge presented in the literature. Furthermore, the different options are compared based on their strengths and weaknesses. Thereafter, we characterize the decision making for components based on the literature. We always indicate which findings were theoretical or empirical.

2.1 Decision Making Problem

Component decision making takes place on different levels. Fig. 1 depicts the different levels at which decisions are made.

1. http://orion-research.se

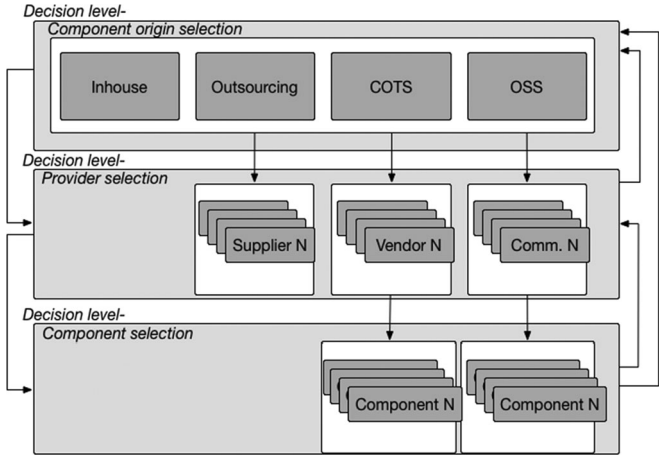

Fig. 1. *Decision levels when choosing components:* The figure shows the different levels of decision making, distinguishing between the CSO choice, vendor choice, and the selection of the actual component.

On the top (strategic) level, the CSO is selected. On the provider selection level, depending on the option, either vendors, suppliers, or communities may be chosen. Finally, on the lowest level a concrete component is chosen. Note that we do not imply a particular order in which the decisions are made.

To facilitate and support the choice of components, different researchers provided insights about the different CSOs. The selection of insights are summarized in Table 1 to characterize the options companies can choose from. Given that the CSOs have different characteristics they have different implications on the criteria governing the choice between them.

Badampudi et al. [9] synthesized the findings from the literature about the strengths and weaknesses of the CSOs in comparison to each other, and distinguished between theoretical and practical influences. A summary of their synthesis is shown in Table 2. Both empirical data (E) as well as findings based on theoretical reasoning (T). The synthesized evidence is based on a limited set of studies with varying scientific rigor and relevance (cf. [9]). Only nine empirical studies were identified by Badampudi et al. that compared CSOs.

In the case of the study by [3] one of the interviewees pointed out that *"we do not want our developers start writing programs from scratch. At least in the Java world we find very often that the highest quality libraries are the ones that are open source and not the commercial"*. In this case, the CSO has already been chosen (OSS). However, in this case survey study we explore decisions where the CSO choice had to be made, which is of interest to practice and research given the distinguishing characteristics of the CSOs (see Tables 1 and 2).

2.2 Characterizing Decision Making

As mentioned earlier the choice of a component is an architectural decision, specifically a structural decision according to Kruchten [1]. Thus, we first review the literature on architectural decision making to elicit important characteristics of the decisions made. Thereafter, we separately present the literature on the different decision levels presented in Fig. 1.

2.2.1 Architectural Decision Making

Theoretical. Kruchten [1] defined an ontology for architecture decision making. Kruchten divides architectural decisions into different categories:

TABLE 1
CSO Characteristics: The Table Shows the Distinguishing Characteristics Between the CSO Options as They Were Presented in the Literature

CSO	Characteristics cited from papers	Source
In-house	Control over features and evolution of the product	[12]
	Control over the development process and projects initiated	[12]
	Awareness and knowledge of the system lies with the developing organization	[8]
COTS	Lack of control (evolution, quality, functionality) as components are black-boxes	[5]
	COTS development influences the development process	[5]
	COTS are mostly used in real-time, embedded and distributed computing	[13]
OSS	Motivation for choosing OSS: higher quality, shorter time to market, and cost reduction	[3]
	Most important criterion for choosing between different OSS is the vitality of the community, functionality, standard compliance, and ease of integration	[3]
	OSS components are used without modification	[4]
	Cost of locating and debugging defects in OTS-based systems is substantial	[4]
OTS (COTS+OSS)	Issues and challenges in estimating integration effort and debugging	[14]
	Cost estimation factors are: time to understand the OTS, OTS inflexibility, and dealing with OTS evolution and corresponding updates needed by integrators.	[4]
	OTS rarely affect quality negatively (reliability, performance, security were problematic if problems occurred)	[4]

In the case of the characterization of the CSOs all findings are empirical.

- *Structural decisions*. These decisions are concerned which elements (such as components) to include in an architecture, as well as the design of their interfaces. Also, ban-decisions of what should not be included can be specified here. In addition, the behavior of the components is specified, which describes the interaction between different architectural elements.
- *Property decisions:* These decisions are concerned with how to design the architecture (e.g., architectural style) in order to achieve certain properties (e.g., performance). As relevant properties Kruchten mentions usability, security, politics, cost and risk.
- *Executive decisions:* These decisions are concerned with contextual factors, such as the technology or the processes used.

To support the documentation of architectural decisions van Heesch [31] proposed a framework. Elements to be documented were the state of the decision, the decision making group, the problem specifying why the decision was made, the decision taken, the alternatives considered,

TABLE 2
Synthesis of CSO Comparisons: The Table Shows the Synthesized Evidence (cf. Badampudi et al. [9]) of the Performance of CSOs in Comparison to Each Other, and States Whether They Are Empirical (E) or Theoretical (T)

Criterion group	Impacted criterion	COTS		OSS		In-house	
		E	T	E	T	E	T
Time	Time to test and integrate [15]	-				+	
	Time to market [16]	+		+		-	
Cost	Cost of components [16], [17], [18]	+		+			
	Total cost of ownership [19]		=		=		
	Cost of replacing components [16], [20]	-			+		
	Maintenance cost [21], [22], [23]	-		-		+	
Effort	Selection and integration effort [16], [22]	-		-			
	Development effort [16]	+		+		-	
Quality	Quality in general [16], [17], [20], [24], [25], [26]		-	+	+		-
Market trend	Component evolution [16], [19], [21], [23], [27]	+	-	-			
Source code	Access and use of source code [16], [18], [19], [22], [28], [29], [30]		-	+	+,-		
	Source code documentation [17], [27]			-	+,-		
Tehnical support	Vendor response time [16], [20], [29]	+		-	+		
	Support availability [19], [26]	-		+			
	Code customization [24]		-	-	+		
	Changes in requirements [21], [26], [27], [28]	+	-		-		
License	License fee [16], [18]			+			
	License obliations [17], [19], [20], [30]			-	-		

If a CSO affects a criterion positively in comparison to others, this is indicated by a "+"; if it is negatively affected this is indicated by a "-"; if no difference is observed this is indicated by a "="; If both, positive and negative effect is observed this is indicated by a "+,-".

system concerns, and the history of he decision. An interesting aspect mentioned is that decisions are often not taken in isolation, but rather in groups, Hence, van Heesch proposes to also capture links to all related decisions. According to Heesch typical stakeholders in the decision making are architects, reviewers, managers, customers, requirements engineers, new project members, and domain experts.

Given that architectural decisions are commonly prepared and made by a group of people, group decision making (GDM) plays an important role. The literature suggests a variety of approaches to support GDM. Malavolta et al. [32] created a meta-model allowing to capture the different viewpoints of stakeholders. The meta-model captures rationales, issues, concerns and criteria of the decision and is linked to a group decision model. The group decision model captures and tracks the decisions that are related to a group of stakeholders. Features for conflict resolution strategies and traces to other relevant artifacts are also captured. Similarly, Nowal and Pautasso [33] provided an approach for the systematic recording of argumentation viewpoints. Argumentation viewpoints should raise situational awareness of teams and hence provide a means to build a consensus. To record an argumentation viewpoint design issues, alternatives and the stakeholder positions in relation to the alternatives are captured. Nowal and Pautasso developed a tool supporting brainstorming and the evaluation of design elements. A Software architecture warehouse facilitates the tracking and sharing of decisions by a team. Zimmermann et al. [34] highlight that a common problem of architecture decision making is the lack of a documented rationale. Zimmerman et al. emphasize the identification of reusable decisions, and hence highlight the importance of capturing the information about the decision. They also point out that decisions are not made in isolation from other decisions. They propose a conceptual framework for proactive decision identification, collaboration and enforcement. The framework provides a wide range of features, such as proposals for templates and the identification of reusable decisions, process support for identifying, making and enforcement of the decisions, pushing to-do-lists to the architectural teams, and providing support techniques to reflect on the decisions. The feature of decision enforcement allows to directly inject decisions into the code. The proposal has been proven to be practical in a service-oriented architecture (SOA)-based application.

Empirical. Capilla et al. [35] describe how to capture architectural design decisions and provide tool (Architecture Design Decision Support System) support to facilitate knowledge transfer and provide deeper insights into the rationales behind architectural decisions. They conducted a case study with students using the tool and captured the effort of the architecting activities and the effort spent on the decision. The usability and ease of use was positively assessed. Reasoning activities for capturing the design decisions required 47 percent of the overall effort spent. To support reasoning activities Razavian et al. [36] introduced design reflections in student groups by asking reflective questions to the students. This led to backtracking and rethinking design decisions. Overall, external triggered reflections improved the quality of discourse and had a positive effect on reflections taking place within the groups.

Tofan et al. [37] evaluated a decision making process (GADGED) based on the repertory grid approach [38] to determine whether it increases the consensus in GDM for architectures. The repertory grid approach systematically elicits the decision alternatives, constructs by which they are compared and the ratings for each alternative in relation to the constructs. A statistical analysis is thereafter conducted [38]. The findings showed that GADGED was useful for group decision making, in particular for inexperienced architects.

To determine whether existing GDM approaches provide sufficient supports to groups wanting to make architectural decisions Rekha and Muccini [39] propose an evaluation framework. The framework checks the presence of features to facilitate learning, problem analysis, the ability to rate alternatives, as well as conflict resolution, to name a few. Rekha and Muccini applied the framework to existing GDM approaches and found that they do not fully support GDM in their current form. Examples of features missing are the ability for stakeholders to explicitly indicate their preferences, as well as conflict resolution mechanisms and rules determining how the preferences of stakeholders should be taken into account. Groher and Weinreich [10] asked student groups with significant practical experience and GDM knowledge to individually develop GDM tools. Later, the tools were compared against the framework proposed by Rekha and Muccini [39]. They found that the tools mostly fulfilled the features. New interesting ideas emerged, in particular the possibility to provide features for the review of decisions after they have been made and communication means between stakeholders inside the tool.

As architectures evolve continuously decision makers have to conduct assessments on a continuous basis. For each decision a trade-off has to be made between different properties of the architecture, such as cost, performance and reliability. Cortellessa et al. [40] proposed a model-based framework utilizing an optimization model with the aim of minimizing cost while thresholds are set with regard to reliability and performance. Initial evaluations showed that the framework performed better in decision making compared to humans, which provided indications of the usefulness of optimization models in decision making.

2.2.2 Choosing CSOs

Theoretical. The solutions identified in the literature by Badampudi et al. (cf. [9]) only aiming at deciding between CSOs were solution proposals without a rigorous empirical component. To decide between in-house and COTS development optimization models have been proposed [20], [23], [41], [42], [43]. As an example, an optimization model may be used to make a choice of components to maximize reliability, while minimizing delivery time. Different constraints can be defined, such as costs. For making a trade-off between in-house development and outsourcing Kramer and Eschweiler [44] propose to utilize clustering to group components and utilize requirements dependencies and priorities to make the choice. Kramer et al. [45] propose utilizes outsourcing potential, knowledge specificity, and interdependencies to make a decision utilizing decision tables.

2.2.3 Choosing Vendors

Empirical. The selection of vendors and suppliers has been considered in several studies. A secondary study has been

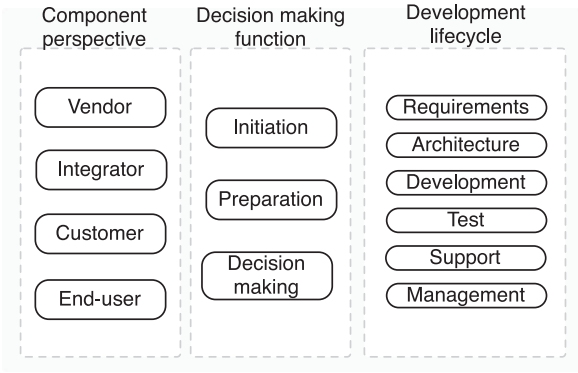

Fig. 2. *Stakeholder perspectives:* The figure shows the different stakeholder perspectives derived from the literature based on which a role can be described. For example, a software tester initiating the decision on integrator side.

presented by [46], which identifies the success factors in the selection of offshore outsourcing vendors. In total 22 factors were identified from the primary studies. Cost-saving, skilled human resource, appropriate infrastructure and quality of products were among the factors that were identified in more than 50 percent of the primary studies. Studies [22], [47] present findings related to vendor selection and community selection, respectively. The relationship between component integrator and external vendor (COTS vendor and OSS community) is considered important to minimize the time and effort on technical, legal and business negotiations [22]. In addition, the collaboration between the OSS developers and actives users is considered important to maintain good communication and relationship with the OSS community.

2.2.4 Choosing OTS Components

Empirical. Vale et al. [13] conducted an extensive systematic mapping study on component-based software engineering (CBSE) to find open issues and research trends. With regard to component selection they found that COTS are commonly selected for embedded, real-time and distributed systems. Looking at the field of CBSE limited evaluations and a lack of empirical knowledge have been identified. Similarly Ayala et al. [2] found limited evidence in relation to selection practices in practice. To address this gap multiple researchers conducted studies on how OTS components are chosen.

Ayala et al. [2] found a gap in the processes proposed in the literature versus what has been used in practice. For example, component repositories are proposed, but not often used in practice.

Ayala et al. [2] and Gerea [3] found that common steps in the component selection process are identification, evaluation, learning and knowledge management, use of the component, and choosing. Gerea also found that the process of selection process is impacted by the component size. Larger components were selected earlier in the development lifecycle. The process used for selection is rarely formal and rather ad-hoc in nature, which has been reported by multiple authors (cf. [2], [4], [5]). For COTS selection Li et al. [4] found that companies use prototyping to learn about COTS.

In the case of OSS the stakeholders involved in the process are the owners of the OSS, developers (distinguishing between core and general developer), system testers, user support as well as problem reporters and users [48]. The initiation of the decision for OSS as well as the preparation and investigation is mostly done by software developers. The leaders (such as the CEO) then take the decision and have the final word [3]. End-users are also involved in the decision making [3]. To systematize the stakeholder roles three perspectives can be identified, the decision function (initiation, decision preparation, and decision making) [3], the relationship to the component (vendor of the component, integrators, and customers/end-users) [5], as well as the roles in the development life-cycle (such as developers and testers) [48]. The perspectives and corresponding roles are summarized in Fig. 2.

Gerea et al. [3] found that the most Important criteria for decision making for OSS are the compliance to standards and the matching of the functionality provided by the component to the needs. Another important criterion for OSS component selection is the vitality of the OSS community. Architectural considerations are also of great importance [5]. In addition to the criteria, risk factors need to be considered during component selection, integration and maintenance, which were identified by Morandini et al. [49] and Li et al. [14]. The risk factors were related to ill-estimation of selection, integration and maintenance effort, wrong component selection, component integration and maintenance failure. Furthermore, legal risk with respect to intellectual property and license were identified. Risk reduction activities were to invest in learning the relevant components and integrating components that are unfamiliar first [14]. In addition extensive testing of the components is important [14].

The best practices for COTS selection in literature and in industry were identified by Rikard et al. [50] and three COTS selection methods (Comparative evaluation process, COTS-based requirements engineering and framework of COTS selection) were compared by Wanyama and Far [51].

3 Method

We first present the research questions, and thereafter provide details on the case survey method and how it has been used in this study. We utilized the guidelines by Larsson [11] during the design of the research.

3.1 Research Questions

The first research question was concerned with how CSOs were selected in the context of software-intensive system development. In order to determine how the decisions were made, several sub-questions needed to be answered:

- *RQ1: How are CSOs chosen?*
  - *RQ1.1: Which CSOs (a) were considered and (b) which CSOs among those considered were chosen?* The first sub-question (a) provides an insight of the decision making problem formulated by the companies. Knowing the combinations frequently considered by companies helps to guide future research, in particular it shows which CSOs should be compared empirically so that companies may use this information as input to their decision making process. In the second sub-question (b) we investigate whether trends are visible of one CSO being preferred over another, which provides early indications of preferences

between CSOs with respect to their advantages and disadvantages.

- *RQ1.2: Which stakeholders were involved in the decision process?* Whether a decision is taken individually or in a group is an important aspect when designing decision support for CSOs. Also, understanding the different roles involved in the stages of decision making provides practitioners with the possibility to reflect on whether the identified roles are also relevant in their decision scenarios.
- *RQ1.3: Which criteria (a) were considered for making the decision and (b) which criteria initially considered ended up as significant for the final decision?* Similar to RQ1.1, the criteria show which variables need to be studied when comparing CSOs empirically. Criteria considered in the preparation for the decision making, though not considered as relevant in the final decision, point to a potential for improvements in decision making processes. Thus, it is of interest which criteria are actually those that were essential in the final decision.
- *RQ1.4: Which decision making approach/model was used?* In the literature several methods are proposed (such as optimization models). Understanding the methods used in practice allows to determine whether the solutions proposed in research found their way into practice, and which methods used in practice need to be integrated into decision support systems.

The second research question aims at understanding the outcome of the decision making process:

- *RQ2: What were the decision outcomes of the CSO selection process?*
  - *RQ2.1: What was the effort invested in the decision making process?* When looking at solutions in software engineering (in this case how CSOs are chosen) the cost factor has to be considered. Thus, we investigated the estimated effort spent on decision making.
  - *RQ2.2: Were the CSOs chosen considered the "right" choice retrospectively?* In this question we reflect on the selected options for making a decision from the retrospective point of view: finding that decisions are mostly positive indicates a success of CSO decision practices. Negative results on the other hand point to the need for decision support and adjustments in the practices.

3.2 The Case Survey Method

We provide an introduction to the case survey method as it has not been widely applied in the software engineering context.

Studies often report only a small number of cases in a single publication. On the other hand, surveys focus on a large number of data points and are mostly quantitative. The case survey method is a compromise of the two approaches [11], as Larsson points out *"it can overcome the problem of generalizing from a single case study and at the same time provide more in-depth analysis of complex organizational phenomena than questionnaire surveys"* (cf. [11], p. 1566) . According to Larsson there are multiple benefits of using the case survey method. As cases are synthesized the case survey adds value to previous individual cases, and the richness of case studies can be incorporated to draw conclusions in the quantitative synthesis. A data extraction scheme is used for capturing the data from the cases, hence it is possible to easily extend the case survey by adding further cases. Overall, Larsson concludes that the case survey is a means to bridge the gap between positivist approaches (such as surveys) and humanistic/interpretivist approaches (such as case studies).

The process of the case survey method comprises of four different steps:

1) Select the cases of interest.
2) Design the data extraction form for elicitation
3) Conduct the coding
4) Use statistical approaches to analyze the coding

The process steps as executed in this study are outlined in Sections 3.2.1, 3.2.2, 3.2.3, and 3.2.4.

3.2.1 Step 1: Select the Cases of Interest

The focus of this paper is on choosing CSOs, such as choosing between in-house development versus open source for software-intensive systems. The components chosen should be utilized as parts of the system-intensive system developed. For example, if an automotive component is developed, and the choice is where to obtain the integrated development environment (IDE), this case would not be included in the study. On the other hand, if the focus of the software development is the IDE itself, then this case would be included. As pointed out by van Heesch et al. [31] architectural decisions are often not independent and thus are bundled into a single decision problem. This was also the case in this case survey.

The inclusion criteria thus can be summarised as follows:

- The case provides information of how the decision making between at least two CSOs has been taking place where the component should become part of a software-intensive system. For example, a database component becomes part of the system, while the development environment does not.
- The system for which the CSO decision is made is industrial (can involve academics if they are supporting the industry).
- Cases should at least be explicit about the CSOs considered, the persons involved in the decision making process, the CSO chosen, the methods used in decision making, and the criteria used when preparing and making the decision.
- Cases were elicited from two sources, namely researchers with industry experience of the ORION project reporting cases from industrial systems where they have been involved in the decision, and interviews with industry practitioners outside of the project.

The target population are cases of making decisions for CSOs for software-intensive systems. All cases are based on decisions for industrial systems. The sampling strategy was convenience sampling, i.e., we reported cases that we could access through the ORION project and industrial contacts.

TABLE 3
Sources for Cases: All Cases Focus on Decisions for Industrial Systems

Source	Number of cases	Case IDs
Cases reported by ORION researchers	11	1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 19
Interviews with industry stakeholders (outside ORION)	11	10, 11, 12, 13, 14, 15, 17, 18, 20, 21, 22

Own experiences refer to researchers who in the role of industry practitioners have been involved in CSO decision making (11 cases). In addition interviews were conducted with industry practitioners (11 cases).

In order to obtain the data, two approaches were used, namely members of the ORION project, and interviews with industry practitioners outside of the project (see Table 3). Members of the ORION project reported the cases using the data extraction scheme (Table 5). As the researchers had access to different networks a diverse sample could be obtained focusing on different domains and including experience from industrial and academic environments. The experiences of the research participants originate from knowledge obtained about relevant cases during former industrial projects where the subjects provided support as researchers (three cases), the work done in an own company or work (three cases), as well as consultancy work (five cases). The interviewees were identified through the researchers' networks. All decision cases presented were done for real-world industrial systems. The introduced data extraction scheme (Section 3.2.2) served as the interview guide. The interviews lasted for 45 minutes to 1 hour.

Table 4 provides the years of industry experience of the subjects in this study. Overall, the subjects have substantial experience in the software industry.

3.2.2 Step 2: Design Data Extraction Scheme for Data Elicitation

There is a risk that the participants in the study eliciting the cases misinterpret the items in Table 5. Thus, the data extraction scheme was reviewed by all ORION project participants. The quality of the form and its understandability were essential in the extraction process, and was important for the validity of the results. An initial form was designed by the first author of the study. The co-authors reviewed the form and submitted change requests to the first author, each reviewer could see the comments already submitted. Each review was considered and a rejoinder was written to keep track of changes, including a response motivating the change, and an action specifying what was changed. After no further comments had been received, the form was considered of sufficient quality to start the data extraction.

The initial version of the data extraction form was defined based on the literature on decision making for CSOs (see related work), and ongoing work to create a taxonomy to describe the decision making for choosing among CSOs, the so-called GRADE taxonomy [52]. The taxonomy defines criteria, roles, decision making methods, environments, and decision making goals. We also explicitly asked the practitioners to expand on context and decision criteria relevant to the decision if they were not captured yet. Thus, all factors presented here were the ones that the practitioners were aware of, or raised as essential.

TABLE 4
Experience of Subjects: The Table Shows the Experience of the Subjects That Were the Sources of the Cases

Measure	Experience (in years)
Average	14.44
Median	13
Minimum	4
Maximum	13

We state the average, median, minimum, and maximum experience in years.

3.2.3 Step 3: Conduct the Coding

Not all information in the form needed to be coded, and we only coded information directly linked to the research questions. As can be seen in Table 5 many items were either integer values, enumerations, or boolean (present or not present in the case). That is, only items 13, 14, 15, 27, 28, 29, 33, and 34 in Table 5 were coded. These concern, among others, decision outcomes, lessons learned, or decision criteria not captured in the initial version of the extraction form. The initial coding was conducted by the first author. An open coding strategy was followed. For example, items 12-14 in the data extraction scheme (Table 5) referring to the stakeholders can be found in Tables 9 and 10. Also, through the coding a terminology control was performed so that we consistently refer to roles and quality attributes. For example, the subjects provided the criteria *"API fit, compatibility with minimal adaptation"* and *"Very important that the solution is compatible with the other component in the system"*, which were grouped under *"Product-compatibility"*. The coding was reviewed by two co-authors of the paper, Deepika Badampudi and Syed Muhammad Ali Shah. The coding was done based on notes taken during the interviews as well as the information filled in by the ORION members.

3.2.4 Step 4: Analysis

Yin and Heald [53] report the results of the case survey in terms of vote counting. A similar approach is utilized in this case survey (e.g., the number of cases considering a CSO, the number of cases considering different criteria, etc.).

Odds ratio is used as a statistical analysis method to quantify how strongly the presence or absence of property A is associated with the presence or absence of property B in a given population. In this case, the association between the presence or absence of a decision criterion and the presence or absence of a decision option is measured through odds ratio (see RQs 1.3 and 2.3). To compute odds ratio we determine the following variables:

- a = Number of cases where the criterion is considered and the decision option is chosen
- b = Number of cases where the criterion is considered and the decision option is not chosen
- c = Number of cases where the criterion is not considered and the decision option is chosen
- d = Number of cases where the criterion is not considered and the decision option is not chosen

TABLE 5
Extraction Scheme: The Extraction Scheme Shows the Category of Extraction Items, the Name of the Item, Its Description, Value Domain (Data Type) and Links to Research Questions

Item ID	Category	Item	Description	Type	Research question	Comments
1	Meta-information	Code	Unique identifier for case	Integer	X	X
2	Meta-information	Author	ORION research participant providing the case	String	X	X
3	Meta-information	Company	Case company	String	X	X
4	Meta-information	Source	Origin of the case	String	X	X
5	Meta-information	Decision scenario	A short summary of the decision case	String	X	X
6	Context	Domain	Domain in which the decision was taken (e.g., automotive, avionics)	String	X	X
7	Context	Application type	Type of the application developed (e.g., embedded, information system)	String	X	X
8	Context	Company size	Size of the whole company	Integer	X	X
9	Context	Development unit size	Size of the development unit where the component was used	Integer	X	X
10	Context	Development methodology	Software Development Methodology used	String	X	X
11	Context	Other	Other relevant context factors to consider	String	X	X
12	CSOs	CSOs considered in the decision	CSOs = In-house, COTS, OSS, Outsource	Enumeration	RQ1.1	X
13	Stakeholders	Decision initiator	Stakeholders involved in the initiation of the decision (identification of the need to make a decision and formulation of the decision problem)	String	RQ1.2	requires coding
14	Stakeholders	Stakeholders in decision preparation	Stakeholders preparing the information and reflections needed to make a decision	String	RQ1.2	requires coding
15	Stakeholders	Decision makers	Stakeholders taking the decision	String	RQ1.2	requires coding
16	Decision criteria	Performance	Response time, timing behaviour of the system	Boolean	RQ1.3	X
17	Decision criteria	Maintainability	Ease of updating the system (corrective, enhancements)	Boolean	RQ1.3	X
18	Decision criteria	Reliability	Reliability of the system	Boolean	RQ1.3	X
19	Decision criteria	Security	Security of the system	Boolean	RQ1.3	X
20	Decision criteria	Time	Time (duration) to develop the system	Boolean	RQ1.3	X
21	Decision criteria	Cost	Cost to develop the system	Boolean	RQ1.3	X
22	Decision criteria	Market and contract	Information about the market (mass market or bespoke development)	Boolean	RQ1.3	X
23	Decision criteria	Access and control	Ability to access the code and control the evolution of the component	Boolean	RQ1.3	X
24	Decision criteria	Component usage in the system	Ease of use when using the component in the system	Boolean	RQ1.3	X
25	Decision criteria	Component history	Evolution of the component in terms of maturity and change history	Boolean	RQ1.3	X
26	Decision criteria	Certification	Certification of the component, certification of the company offering a component	Boolean	RQ1.3	X
27	Decision criteria	Other	Other criteria not mentioned	String	RQ1.3	requires coding
28	Decision method	Decision model	Method used to make the decision	String	RQ1.4	requires coding
29	Decision method	Property model	Method used to estimate the impact of the decision	String	RQ1.4	requires coding
30	Decision results	Decision outcome	CSOs chosen	Enumeration	RQ2.1	X
31	Decision results	Decision preparation effort	Effort in preparing the decision (estimated)	Integer	RQ2.2	X
32	Decision results	Decision making effort	Effort in making the decision (estimated)	Integer	RQ2.2	X
33	Decision evaluation	Evaluation of the decision and the decision impact	Important criteria of the decision and reflections on the success/failure	String	RQ2.3/RQ2.4	requires coding
34	Other information	Noteworthy comments	Remarks considered important by the person extracting the case	String	X	requires coding

Furthermore, we indicate where coding of the information was provided. The extraction scheme was used as the guide for interview.

The OR is thereafter calculated as

$$OR = \frac{a/c}{b/d} = \frac{a * d}{b * c}. \quad (1)$$

The confidence interval (cf. [54]) for OR values is calculated as

$$Upper95\%CI = e \wedge [ln(OR) + 1.96\sqrt{1/a + 1/b + 1/c + 1/d}] \quad (2)$$

$$Lower95\%CI = e \wedge [ln(OR) - 1.96\sqrt{1/a + 1/b + 1/c + 1/d}]. \quad (3)$$

The OR values are interpreted as follows:

- OR = 1 The criterion does not affect odds of decision option being chosen.
- OR >1 indicates that the criterion is associated with higher odds of the decision option being chosen.
- OR <1 indicates that the criterion is associated with lower odds of the decision option being chosen.

The values of confidence interval are considered to determine if the results are statistically significant. If the range of confidence intervals include the value 1 then the result is determined as not statistically significant, which is the case for the included cases (see Table 13). Power analysis is a statistical test to determine the sample size needed to detect an effect with a given degree of confidence. It is based on statistical assumptions and data characteristics. In particular, the characteristic that we would have to know is the relative precision implying skewness of the distribution of the odds ratio value in order to do a power analysis. However, we do not have historical data to specify the desired value with confidence given that our study is of exploratory nature.

TABLE 6
Overview of the Cases: The Table Shows Contextual Information About the Cases

Case ID	Company Size	Size development unit	Domain	Application type	Development methodology
Case 1	100.000	5.000	Automotive	Embedded systems	Iterative development, Lean manufacturing
Case 2	1.200	350	Utilities	Embedded + Software + Apps	Agile SCRUM variant
Case 3	40	32	Human resource management	Software	Hybrid Plan driven, Agile
Case 4	21	20	Trade, Security, Consumer	Software, Hardware, apps	Hybrid Plan driven, Agile
Case 5	500	6	Financial	Software + Hardware (servers)	Hybrid Plan driven, Agile
Case 6	17.000	1.300	Surveillance/Security	Software + Hardware	Plan driven, iterative
Case 7	NA	NA	Telecommunication	Embedded system	NA
Case 8	NA	NA	Automotive	Embedded control systems	Iterative, informal process
Case 9	100.000	50	Automotive	Interactive tool for calibration	N/A
Case 10	100.000	4.000	Automotive	Embedded software architecture	Iterative development, time-boxed deliveries, incremental
Case 11	100.00	4.000	Automotive	Embedded system architecture	Waterfall
Case 12	100.00	4.000	Automotive	Embedded system architecture	Waterfall
Case 13	20	4	Defense	Search engine	Agile
Case 14	20	3	Marketing & Advertising	Analytics tool	Waterfall
Case 15	14.000	180	Defense	Mission critical	Streamlined development, Hybrid process using the most appropriate development methods and techniques.
Case 16	140 000	300	Process automationand roboticsfor several industries	Embedded control system	Waterfall with safety certification and extensive changeimpact analysis process.
Case 17	100.000	30	Automotive	Embedded control system	Each of the department uses its own development methodology but there is a global process (V model). The components used SCRUM
Case 18	10.000	10	Automotive	Control-dominant software	Agile
Case 19	N.A. (large scale)	1.000	Telecom	Information system	Hybrid Plan driven, Agile
Case 20	100.000	100	Telecom	Charging system	Scrum
Case 21	100.000	5	Telecom	Embedded system	Scrum
Case 22	3	3	Telecom	NA	NA

The size refers to the overall (including different development organizations), while the size of the development unit refers to the organization where the selection of the component took place. The development units were located in Sweden. Furthermore, we characterized the domain, the application type, and the development methodology used in the development unit.

Ayala et al. [2] also highlighted this, in particular sourcing/component decision practices in industry are not widely investigated. Hence, at the exploratory stage we do not know these characteristics to the degree to make a reliable power analysis. Miller [55] points towards qualitative direction as an alternative to statistical significance testing. Thus, in our results we utilize the statistical analysis from odds ratio to triangulate the findings with the responses from the subjects of this study (see Section 4.2.3).

4 Results

4.1 Overview of Cases

Table 6 provides an overview of the 22 cases included in the case survey. The cases were characterized by the size of the company, the development unit where the component should be used. Furthermore, the domain, application type, and development methodology were specified. Half of the cases were in the context of the automotive domain (11 of 22), while other contexts have been considered as well. Given the proportionally high number of automotive cases, the most frequently reported application type was embedded systems. A variety of software development methodologies has been used, including agile and plan-driven processes. Furthermore, multiple cases reported hybrid processes combining agile and plan-driven concepts. With regard to sizes a range of different company sizes as well as sizes of the development units concerned have been reported. Overall, cases with varying sizes, domains, and development methodologies have been obtained.

4.2 RQ1: How Are CSOs Chosen?

In the context of RQ1 we investigated the CSOs considered, the involved stakeholders, the decision criteria, and the decision model used.

4.2.1 RQ1.1: Which CSOs (a) Were Considered and (b) Which CSOs Among Those Considered Were Chosen?

The decision making problem constitutes the choice of CSOs for a software intensive system, namely in-house, COTS, OSS and Outsource. In addition the practitioners considered services as a separate option in their decision making. Looking at the definition of services they were considered in the category of COTS by Ayala et al. [2]. More specifically, they are defined as a way of delivering functionality: "*Software-intensive services, often delivered as cloud or internet services, can also be*

TABLE 7
Decision Making Options Considered: The Table Shows the Frequency of Options Considered

CSO	Frequency of consideration	%
In-house	17	32.08
COTS	14	26.42
Outsource	11	20.75
OSS	6	11.32
Services	5	9.43
Total	53	100.00

products from all industries like financial, insurance, gaming, social software, or personal services based on software" (cf. [56]). In this paper our aim was to present the decision making problem as it was formulated by the practitioners, hence services has been presented as a separate option. It is interesting to observe that what is conceptually incorrect (i.e., including services as a CSO) was not a factor for the practitioners to exclude them as an option in their decision making.

Table 7 shows the most frequently considered CSOs. As is evident from the table the most frequently considered CSOs were In-house, COTS and Outsource. This information is important to, for example, decide which options have to be well supported by evidence with regard to benefits and limitations of one alternative over another (e.g., COTS in comparison to In-house). Only in five cases the practitioners considered services as an option.

To understand which CSOs are traded off against each other we analyzed the frequency of combinations for comparisons between the alternatives, which is shown in Table 8. The most frequent trade-offs were between In-house versus COTS, In-house versus Outsource, and COTS versus OSS. It is no surprise that the most frequent comparisons comprise of the CSOs being most frequently considered (see Table 7).

The number of options considered has an implication on decision support methods, i.e., one needs to determine whether solutions can support a trade-off between two or more alternatives. The number of options considered in the cases are thus illustrated in Fig. 3. The figure shows that it is common to consider two options, while there are still a number of cases that involve three options and more.

Fig. 4 shows the options considered for each case as well as the options chosen. The elements in the figure are to be interpreted as follows:

- CSOs that were considered, but not chosen, are represented as black circles. For example, in case 12, In-house has been considered, but it was not chosen.

TABLE 8
Decision Making Option Trade-Offs: The Table Shows the Frequency of the Considerations of Pairs for the CSO Options in the Cases

	In-house	COTS	OSS	Services	Outsource
In-house		**9**	1	4	**11**
COTS			**6**	3	4
OSS				1	1
Services					2
Outsource					

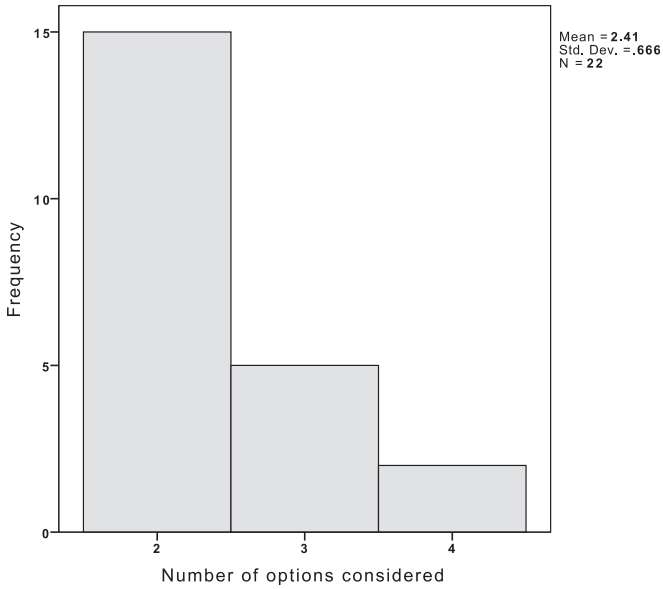

Fig. 3. *Number of options considered:* The figure shows a histogram of the number of options considered for the set of cases.

- Decisions that deviated from the recommended choice based on the investigation during the decision preparation are represented as black diamonds. For

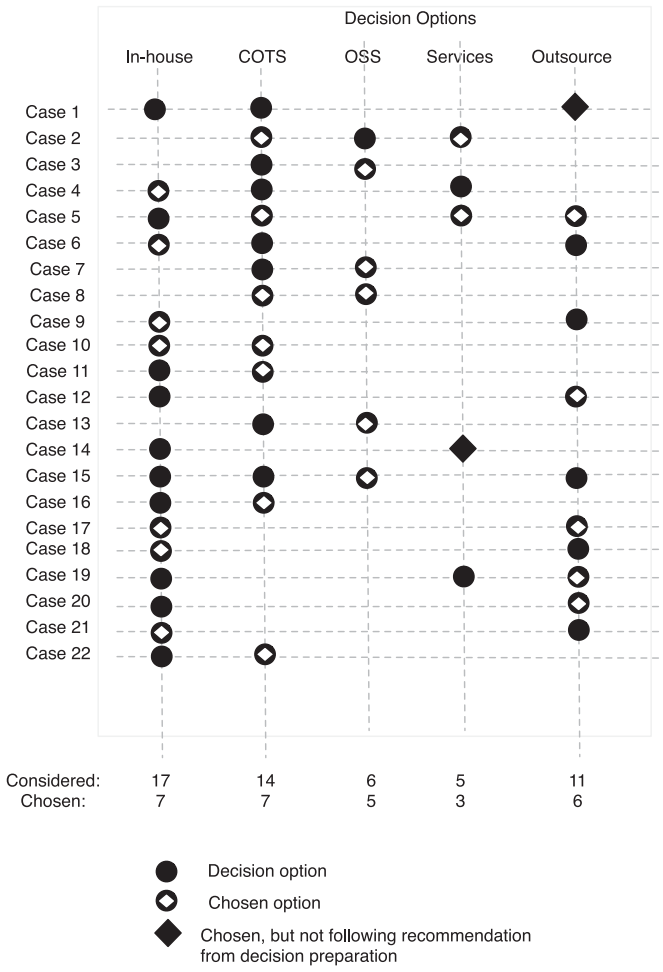

Fig. 4. *Decision outcomes:* The figure shows the decision options considered and chosen for each case. In addition, we illustrate how often each option has been considered, and how often it has been chosen.

TABLE 9
Decision Making Roles: The Table Shows the Different Decision Making Roles Involved

Roles	Initiation		Preparation		Deciding	
	Abs	%	Abs	%	Abs	%
Software management	15	62.5	32	43.24	21	80.77
Software construction/dev.	7	29.17	9	12.16	4	15.38
External support	1	4.167	8	10.81	0	0.00
Software test/quality control	1	4.167	2	2.70	0	0.00
Customers	0	0.00	3	4.05	0	0.00
Expert group (group of roles, unspecified)	0	0.00	5	6.75	0	0.00
Legal	0	0.00	2	2.70	1	3.84
Sales (business/customer relations)	0	0.00	4	5.40	0	0.00
Software architecture/ design	0	0.00	6	8.10	0	0.00
Sub-contractors (providers of components)	0	0.00	3	4.0554	0	0.00
Total	24	100.00	74	100.00	26	100.00

It distinguishes between decision initiation, preparation, and decision making. The absolute values and percentages are stated.

example, in Case 1 the recommendation was to go with COTS based on the decision preparation, but then the choice made was Outsource, which was also a CSO considered from the beginning in the decision making process.

- CSOs that were considered, chosen, and did not deviate from the recommended choice, are illustrated as circles with a white diamond inside.

In 7 of 22 cases in-house development has been chosen among the alternatives. Also in 7 cases components off-the-shelf was the CSO chosen. In-house development has been considered in 17 out of 22 cases. Thus, the reflection of developing internally, or obtaining a component externally from different sources seems to be a key decision in practice. In a few cases only, the recommended alternative has not been chosen, which was true for Cases 1 and 14.

For case 1, the decision was prepared on a technical level considering the wide range of factors (performance, reliability, etc.). The technical people in preparation then passed their recommendation to the decision maker on business level, who decided to ignore the findings and make a political decision. In conclusion, for this case politics and the management of relationships overruled technical considerations.

For case 14, a social media analytics platform required a component for sentiment analysis from Facebook and Twitter. In-house development was recommended and a similar library was already developed that could have been used, but a web-service has been obtained also. In the end, quality issues were observed for the web service (textAnalytics) while also an alternative service (Mashape) was considered. Overall, this led the company to initiate a new decision, namely replace the textAlytics service with Mashape, or to utilize the in-house developed component.

As mentioned earlier decisions are often combined [31]. In the case survey, the cases where more than one CSO (see Fig. 4) has been chosen represent combined decisions. For example, in Case 5 a system for access control, back-end trade, and reporting for a large financial system was developed with high demands on accuracy and performance. This required three components, which were all handled as one decision problem to solve account, trading, and reporting. When we consider this as a decision problem, the company considered build or service for the first component, build or outsource for the second component, and build or get a COTS for the third component.

4.2.2 RQ1.2: Which Stakeholders Were Involved in the Decision Process?

We distinguish three groups of stakeholders in the decision making. According to Strydom [57] the key stakeholders in decision making are:

(1) *Decision initiation*. The decision initiator who is raising the need to make a decision to solve a specific problem (in this case the selection of CSOs).
(2) *Decision preparation*. People in the decision preparation are concerned with the study of documentation, the presentation and production of internal reports to share the findings of investigations of CSOs, meetings in the form of workshops and seminars, as well as informal discussions. Furthermore, they document a rationale. Only a few cases (four cases) the rationale for the decision was not documented, but rather stated in discussions. In the remaining cases the documentation took place in the form of reports.
(3) *Decision makers:* The decision makers (taking the decision and thus *"making the final choice between alternatives" (cf. [57])*). The following ways of making a decision were found: A leader takes a decision and hence no consensus was required; The stakeholders agreed and hence no negotiation needed to take place and consensus was reached; A demonstrator/ simulator was used to illustrate the effects of the solution to facilitate the decision.

Table 9 provides an overview of the roles involved in the decision making, grouped by initiation, preparation, and decision making. Table 10 shows the distribution for the management roles involved.

Decision Initiation. With regard to decision initiators software management is the most frequent initiator for the decision making cases. Non-managerial roles have initiated the decision making process in three cases (software architecture and design/construction).

In three cases (16, 17 and 19) multiple roles initiated the decision. For software management the roles could be further refined. There it is noteworthy that in six cases

TABLE 10
Management Roles: The Table Shows the Different Types of Management Roles Involved

Roles	Initiation Abs	%	Preparation Abs	%	Deciding Abs	%
Executive management (CEO/CTO)	6	30.00	11	31.43	8	32.00
Management (type unspecified)	7	35.00	9	25.71	9	36.00
Product management	1	5.00	7	20.00	2	8.00
Project management	4	20.00	7	20.00	4	16.00
Line management	2	10.00	1	2.86	2	8.00
Total	20	100.00	35	100.00	25	100.00

It distinguishes between decision initiation, preparation, and decision making. The absolute values and percentages are stated.

executive management (CTO/CEO) have initiated the decision making. Other roles initiating the decision making were project leaders/manager, line manager, and product manager. When multiple roles were involved, it was a combination of management and technical roles.

Decision Preparation. Compared with the decision initiation, more distinct roles were involved as stakeholders in the process of supporting the decision with discussion and expert input. Table 9 shows that the most common roles in decision preparation were software management and software architecture and design/construction. It was evident that expert decision support was obtained in twelve cases from either researchers or consultants. Additionally, customer relations/sales and software architects and architects/ designers were involved frequently. Only in a few cases software test (Cases 6 and 7), the actual customers (Cases 2, 4 and 9) and sub-contractors (Cases 5, 9 and 16) were involved.

Decision Making. The decision makers were primarily managers. In comparison, only a few decision makers were in the category of software construction. In two cases (4 and 8) a consensus decision between management and software design/construction was made.

Number of Roles per Decision Making Process. Fig. 5 shows the number of roles involved in the decision making process related to initiation, preparation, and decision making. Understanding the number of roles involved has important implications on how to support the decision making process. For example, as soon as multiple roles are involved there is a need to support consensus building and incorporating multiple points of view during the process. Only a few roles (primarily management) are involved in the initiation (one individual role in 19 cases) and making the decision, while a larger set of roles is involved in the preparation (in average three roles, see Fig. 5).

4.2.3 RQ1.3: Which Criteria (a) Were Considered for Making the Decision and (b) Which Criteria Initially Considered Ended Up as Significant for the Final Decision?

(a) Criteria Considered. The criteria based on which the decision options are evaluated are shown in Table 11. The table indicates the number of cases that have considered the criteria and also the percentage of the total number of cases that consider the criteria.

Frequently considered criteria related to the product are quality criteria, such as performance, reliability, maintainability, compatibility and security. Further product-related characteristics were considered frequently—the most frequently mentioned ones being certification, level of openness and access/control. Furthermore, cost was an important criterion. A sub-set of cases further specifies the type of cost (e.g. to buy/rent, licensing, etc.).

As seen in Table 11, some of the criteria are considered more frequently. We investigated the criteria that are considered together among the most frequently considered criteria (frequencies greater than 10) as shown in Fig. 6. The percentages are calculated by dividing the number of times the criterion is considered together with another criterion by the total number of times a criterion is considered. For example performance is considered 17 times in total and out of the 17 times, it is considered 13 times together with reliability. Therefore the percentage of performance

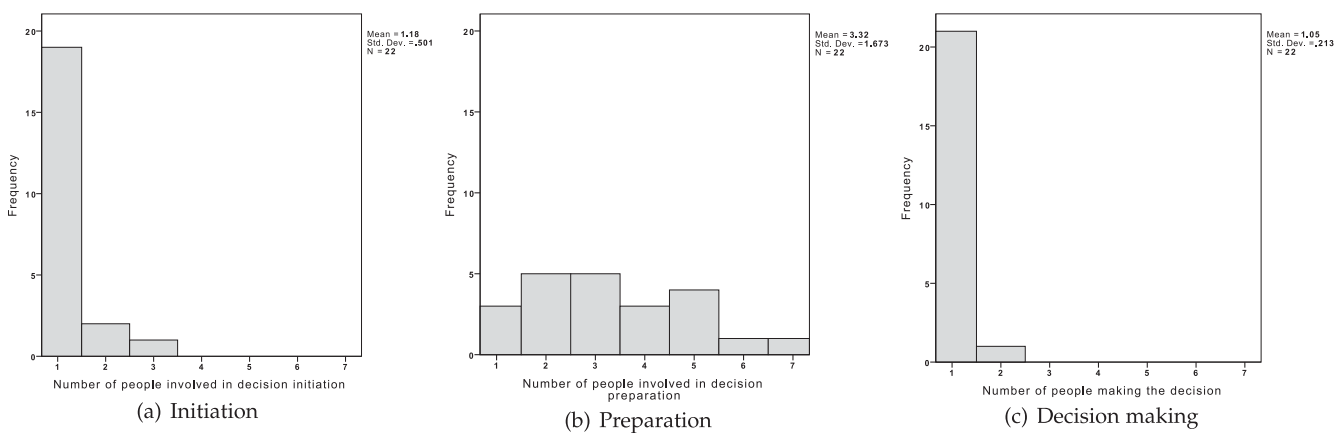

Fig. 5. *Number of roles:* The figure shows a histogram of the number of roles involved in the decisions presented in the cases.

TABLE 11
Decision Criteria: The Table Shows the Frequency with Which the Individual Criteria Were Considered

Group	Criterion	Cases	%
Product	Performance	17	77.27
	Reliability	13	59.09
	Maintainability	13	59.09
	Certification	12	54.55
	Level of openness and access/control	11	50.00
	Security	9	4.91
	Compatibility	9	4,91
	Functionality	7	31.82
	Architectural dependencies	4	18.18
	Compliance to standards and regulations	3	13.64
	Licensing rules	3	13.64
	Portability	2	9.09
	Quality (general)	3	13.64
	Safety	1	4.55
	Stability	2	9.09
	Ease of integration	1	4.55
	Extandability	2	9.09
	Longevity of asset	2	9.09
	Scalability	2	9.09
	User experience	1	4.55
	Availability	1	4.55
	Fitness for purpose	1	4.55
	Number of users	1	4.55
	Robustness	1	4.55
Financial	Cost - general (time, effort, resources)	17	86.36
	Cost - buy/rent	3	1.62
	Cost - acquisition	1	4.55
	Cost - Adaptation	1	4.55
	Cost - Licensing	1	4.55
	Cost - risks incurred	1	4.55
	Cost - total cost of ownership	1	4.55
	Cost - maintenance	1	4.55
Project	Level of support	5	22,73
	Familiarity with technology	1	4.55
Business	Ecosystems	1	4.55
	Market trend	1	4.55
	time to market	1	4.55
Total		158	

and reliability considered together and in this order is (13/17)*100 = 76.47%. For reliability and performance the value is 100 percent as shown in Fig. 6. This means that every time the reliability is considered, it is always considered together with performance. Higher percentages indicate that possible trade-offs between the criteria might need to to be considered in the decision. For example, improving reliability might be done under performance constraints.

The number of criteria considered are shown in Fig. 7. The number of criteria taken into consideration impacts the

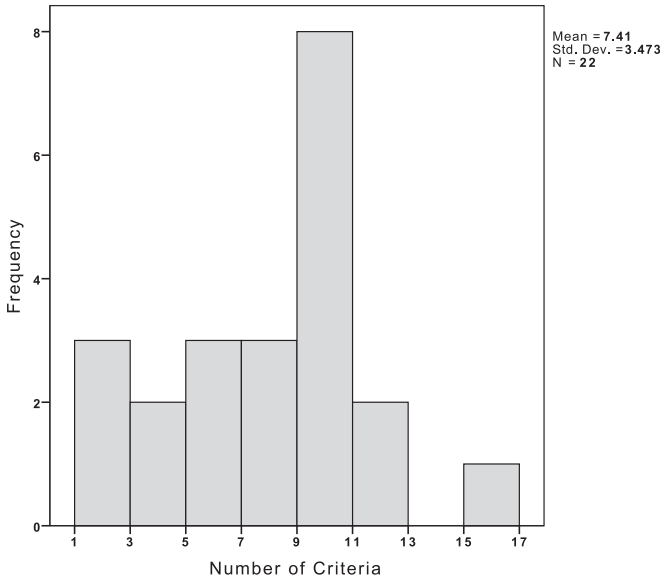

Fig. 7. *Number of criteria:* The figure shows a histogram of the number of criteria considered.

requirements on the solution, e.g., with respect to optimization this means to choose an approach for multi-objective optimization. When conducting an analysis of trade-offs between criteria (i.e., how they positively or negatively influence each other) the complexity of the analysis increases.

(b) Criteria Considered Significant in the Final Decision. To assess which criterion has a significant impact on the final decisions two approaches are used. *First*, we captured which criteria the subjects of the study mentioned as considered important in the final decision (Table 12). The criteria that were important in the final decision are divided into two groups: (1) Criteria that were considered in the preparation and are a sub-set chosen as important by the decision maker, (2) New criteria not considered in decision preparation, but in decision making. *Second*, we statistically explored the association between a criterion and a CSO in order to determine if the decision options are more favorable when a particular criterion is considered. To evaluate the association we considered the odds ratio (OR) between the criterion and decision option (Table 13). Using both statistical and qualitative approaches provides a means for triangulation and qualitative reflection.

We first present reflections for the data as explained by the subjects shown in Table 12. Several interesting observations can be made from the Table 12. In several cases only a small sub-set of criteria was considered in decision making compared to the preparation (see e.g., Cases 4, 5, 8, 17, and 19). That is, if the relevant decision criteria could have been identified earlier, this has a potential to save investigation

	Performance	Reliability	Maintianabiliity	Cost	Certification	Level of openness
Performance	X	76,47	58,82	70,59	58,82	58,82
Reliability	100	X	61,54	92,31	69,23	46,15
Maintianabiliity	76,92	61,54	X	92,31	53,85	46,15
Cost	73,68	63,16	63,16	X	57,89	47,37
Certification	91,67	83,33	58,33	91,67	X	50
Level of openness	90,91	81,82	54,55	63,64	54,55	X

Fig. 6. *Criteria Trade-offs:* The table shows the percentages of how often a criterion is considered together with another criterion in relation to the total number of times a criterion is considered.

TABLE 12
Overview of the Criteria Considered: The First Column of the Table Shows the Criteria Considered in Preparation

Case	Criteria considered in preparation	Criteria considered important in decision making (subset of preparation)	Criteria considered important in decision making (new)
In-house			
Case 4	Performance, reliability, security, cost, user experience, compatibility, level of support, level of openness and access/control, stability, and certification	Security	
Case 6	Performance, reliability, security, cost, user experience, compatibility, level of support, level of openness and access/control, stability, and certification		Tradition of developing in-house
Case 18	Performance, maintainability, reliability, cost, and certification.		Competence
Case 21	Performance, reliability, and certification	Performance, reliability	
COTS			
Case 11	Performance, maintainability, reliability, security, cost, safety, robustness, compatibility, and certification	Cost	Ease to provide solutions to customers
Case 13	Performance, cost, level of support, market trend, compatibility, architectural dependencies, level of openness/access and control, component history, certification	Component history, market trend, performance, (most important), compatibility, cost, level of support, architectural dependencies, level of openness and access/control	
Case 16	Maintainability, cost, fitness for purpose	Cost, fitness for purpose	
Case 22	Cost, functionality	Cost, functionality	
OSS			
Case 3	Performance, reliability, security, cost, scalability, level of openness/access and control	Scalability, functionality	
Case 7	Maintainability, cost	Maintainability, cost	
Case 15	Performance, maintainability, reliability, security, cost, architectural dependencies, level of openness and access/control	Architectural dependencies, reliability	
Outsource			
Case 1	Performance, reliability, cost (general), acquisition cost, adaptation cost, maintenance cost, functionality, quality (general), familiarity with technology, ecosystem, architectural dependencies, longevity of the component, extendibility, level of openness and access/control, compatibility		Relationship with supplier company (maintain)
Case 12	Performance, maintainability, reliability, cost, functionality, compatibility, certification	Performance, cost, maintainability, functionality, compatibility, reliability	
Case 19	Performance, maintenance cost, reliability, security, cost (general), quality (general), time to market, extendibility	Maintenance cost	Adaptability, level of openness and access/control
Case 20	Cost, certification	Cost	
Service			
Case 14	Performance, cost, functionality, availability, level of support, level of openness and access/control	Cost, functionality, performance, availability, level of support	
Combinations			
Case 2	Performance, maintainability, reliability, security, cost, risks incurred, licensing rules, level of support, portability, compatibility, certification	N.A.	
Case 5	Performance, reliability, security, cost, compliance to standards and regulations, cost for buying/renting, scalability, component history, certification	Performance, reliability	
Case 8	Performance, portability, cost, licensing, functionality, portability, licensing rules, level of openness and access/control, ease of integration, component history	Level of openness and access/control, component history	
Case 10	Performance, maintainability, reliability, security, cost, certification, level of openness and access/control, compatibility, stability, number of users	Cost, maintainability, reliability, certification, level of openness and access/control, performance, security	
Case 17	Performance, maintainability, cost, quality (general), compliance to standards and regulations, functionality, level of openness and access/control, compatibility	Cost, functionality, compliance tostandards and regulations	

The second column shows the subset of criteria that were considered in preparation, and were considered as significant in the final decision. The third column shows new criteria that were not considered in the decision preparation, but were considered significant for the final decision.

TABLE 13
Decision Outcomes: The Odds Ratio Values of the Criteria in Relation to the CSOs Is Shown

Criteria	In-house		COTS		OSS		Services		Outsourcing	
	OR	Conf. Int.	OR	Conf. Int.	OR	Conf. Int.	OR	Conf. Int.	OR	Conf. Int
Performance	2,18	0,20 24,21	0,82	0,11 6,34	1,17	0,09 14,52	0,00	0,00	0,81	0,07 9,52
Reliability	2,19	0,32 15,04	0,56	0,10 3,25	0,64	0,07 5,61	0,00	0,00	1,05	0,14 8,02
Maintianabiliity	2,19	0,32 15,04	1,25	0,21 7,41	2,40	0,21 27,72	0,67	0,04 12,27	1,05	0,14 8,02
Cost	0.399	0,04 3,52	0.16	0,00 0,00	0,39	0,00	0,56	0,00	0,64	0,13 26.32
Certification	6,86	0,66 71,72	1,25	0,21 7,41	0,17	0,01 1,96	0.18	0,00	0,78	0.09 5.69
Level of openness	3,75	0,54 26,04	0,45	0,08 2,67	3,75	0,32 43,31	0,00	0,00	0,18	0,02 1,92

The first column represents the frequently considered criteria. Columns two to six represent the odds ratio (OR) and confidence intervals (conf. int.) between the frequently considered criteria and the CSO. For example, column two consists of all the odds ratio between in-house and each criteria. Similarly, row two consists of all the odds ratio for performance with each CSOs. The odds ratios are computed using Equation (1) and the confidence intervals are computed using Equations (2) and (3) (see Section 3.2.4).

effort in the preparation phase (such as pre-studies). An undesirable case is where the input from the preparation is not considered in the final decision, and new criteria become important. That is, the effort spent in preparation is spent on investigating criteria that were not important (see, for example, Cases 1, 6, 11, 18, and 19). For example, in Case 1 a large car manufacturer performed a pre-study to investigate pros and cons as a basis for opting between In-house, COTS, and OSS considering 15 criteria, namely: *performance, reliability, cost (general), acquisition cost, adaptation cost, maintenance cost, functionality, quality (general), familiarity with technology, ecosystem, architectural dependencies, longevity of the component, extensibility, level of openness and access/control, compatibility*. However, the best option with respect to the criteria considered in preparation has not been chosen. Rather a political decision was made to maintain a good relationship with a supplier company.

We now triangulate the findings of the statistical analysis (Table 13) with the information provided by the subjects (see Table 12).

Overall, in-house has higher odds ratio values. In-house seems to be a favorable decision option when all the frequently considered criteria (excluding general cost and maintainability) are considered. In particular, certification has the highest odds ratio (OR = 6,86) when in-house is chosen as the outcome. We can also see in Table 12, certification is considered in all the cases (Case 4, 6, 18 and 21) where in-house is chosen. Though, certification was considered in all cases, it was not the criterion that turned out to be the most significant. As some of the decisions were taken based on new criteria (tradition and competence) as in Table 12. However, since all the cases that had chosen in-house considered certification as a criterion and the higher values of odds ratio in Table 13 indicate that certification is important criterion for choosing in-house.

As seen from Table 13, performance (OR = 2, 18) and reliability (OR = 2, 19) also have the highest odds ratio values when in-house is chosen. This supports the qualitative data in Table 12 as performance and reliability is considered in all the cases where in-house is chosen and also turned out to be important criteria in one of the case (Case 21). Level of openness has higher odds ratio however, it was only considered in half of the cases when in-house was chosen.

Maintainability (OR = 2, 40) and level of openness (OR = 3, 75) have highest odds ratio when OSS is chosen according to Table 13. This is also supported by qualitative data in Table 12 as both maintainability and level of openness is considered in two out of three cases where OSS was chosen. In addition, maintainability ended up being important criterion in one of the case (Case 7).

According to Table 13, cost is associated with lower odds of any CSOs being chosen as all values of odds ratio are below one. However, according to the qualitative data in Table 12, cost has been considered important in choosing COTS, outsourcing and services.

4.2.4 RQ1.4: Which Decision Making Approach/Model Was Used?

Table 14 shows the approaches used in decision making. Expert opinion/judgment has been utilized in all cases. Expert judgment was supported by a number of approaches. Prioritizing and ranking the alternatives with respect to weighted criteria (four cases), listing the Pros and Cons (five cases), and Pugh analysis (four cases) were identified. More formal approaches for estimating (e.g., COCOMO) or the use of models for decision making (e.g., optimization) was not observed. It should be noted that we explicitly considered more structured decision making techniques such as the Analytical Hierarchical Process (AHP), elicitation of weights under decision model [58] (Item 28

TABLE 14
Approaches Used for Decision Making (Frequency)

Decision making approaches used	Cases
Expert opinion/expert judgment	22
Pros and Cons	5
Prioritization, ranking, weighted criteria	4
Pugh analysis (Decision-matrix method)	4
Total	22

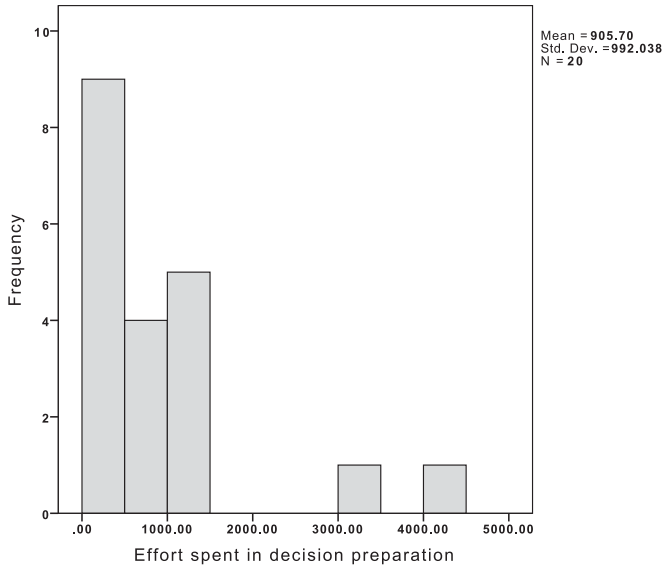

Fig. 8. *Effort in decision preparation:* The figure shows a histogram for the effort spent (persons/hour) on decision preparation. The intervals are illustrated in intervals of 500 person hours.

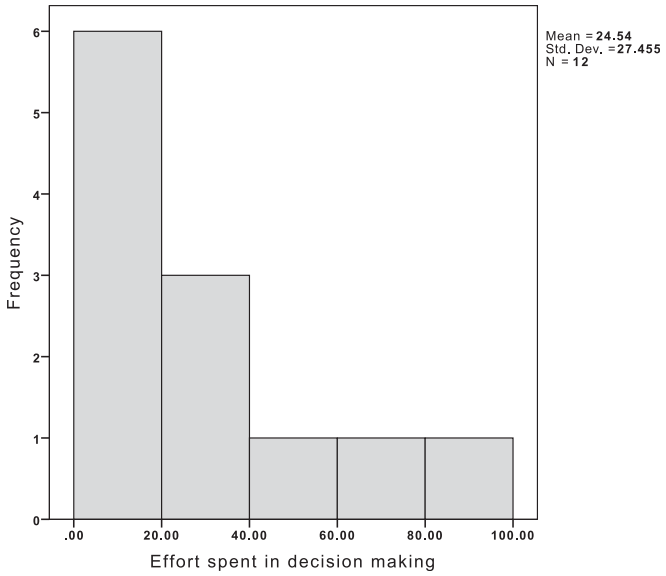

Fig. 9. *Effort in decision making:* The figure shows a histogram for the effort spent (persons/hour) on decision preparation.

in Table 5). However, they were not found in the cases as the decision making was discussion based and ad-hoc.

4.3 RQ2: What Was the Result of the Decision Making Process?

The second research question focuses on the description of the decision outcomes while following the decision making processes characterized in Section 4.2.

4.3.1 RQ2.1: What Was the Effort Invested in the Decision Making Process?

The effort spent on preparing the decision for the decision maker is shown in Fig. 8, the x-axis shows effort spent in person hours. The effort data was available in 16 of 22 cases. The average time spent on preparation was around 780 person hours.

In comparison Fig. 9 shows the effort spent on decision making, but it is available for 16 cases, only. This indicates only a small fraction of the effort spent in preparation (on average 30 person hours) is spent on decision making.

4.3.2 RQ2.2: Were the Chosen CSOs Considered the "Right" Choice Retrospectively?

Looking at how the decisions were evaluated it is visible that a high number of decisions were perceived as sub-optimal, which applied to a total of nine cases (see Table 15). In only seven cases the decision was perceived as clearly positive. This highlights the need for support in the decision making processes to improve the decision making outcomes. Considering the decision year it is visible that in the cases the majority of cases at least two years are in-between the decision year and the year the data was collected (2016). Hence, this allowed for a reflection by the participants as there was sufficient time allowing for an assessment.

Table 15 also provides an insight on why one option was preferred over another. For example, it is visible that in-house was preferred over COTS and services for security reasons (Case 4), in-house over COTS and outsource for performance and stability reasons (Case 21), etc. We briefly present the rationale stated by the subjects for either evaluating the decision positively (see evaluations with the $\surd$-symbol in Table 15) or negatively (see evaluations with the †-symbol).

Positive ($\surd$). When choosing in-house the practitioners perceived the best decision was taken in comparison to outsourcing. Furthermore, another reason to assess the choice of in-house development over outsourcing was the improvement in the reliability of the product. When COTS has been chosen, the absence of issues were raised as the reason for the positive assessment when being compared to in-house development. OSS has been found to save costs, in Case 15 a factor of cost savings by the factor of 10 has been mentioned. When choosing a combination of options Cases 10 and 17 reported positive results.

Negative (†). For the cases choosing in-house development two reasons for negative assessments have been given, namely the decrease in product quality with respect to security and issues with the decision making process itself. When choosing COTS, in one case product quality issues arose with regard to performance. When outsourcing was chosen only negative assessments were found in the cases, all of them being related to product quality issues. For the choice of combinations of CSOs issues arose with regard to the lack of ability to foresee problems and to conduct estimations.

5 Discussion

The discussion is structured along the research questions. In particular, the results of the systematic review [9] and the case survey are compared as both studies had a similar focus.

5.1 Reflections with Respect to the Research Questions

The first research question explored how CSOs are chosen in practice, four research questions were formulated and are discussed in the following. Three of the four questions are

TABLE 15
Decision Outcomes and Their Evaluation: The Table Shows the Outcome for Those Decision Cases Where a Decision Has Been Reached at the Point of Time of Data Collection

Case	Decision year	Outcome of the decision	Evaluation of the final decision √ = positive, o = indifferent, † = negative)
In-house			
Case 4	2012	In-house over COTS and Services	†: **Product - Security:** Company needed to build themselves, as security requirements could not be met by other options, i.e., other options would have been preferred if possible
Case 6	2011	In-house over COTS and outsource	†: **Decision making process:** Huge effort invested in the investigation of vendors, and in the end it was done in-house anyways
Case 18	N.A.	In-house over outsource	√: **General:** Perception that the best decision was made (no rationale given)
Case 21	N.A.	In-house over outsource	√: **Product - Reliability:** Reliability was improved
COTS			
Case 11	2006	COTS over in-house	√: **Absence of issues:** No issues later on, decision was considered a success
Case 13	2013	COTS over OSS	†: **Product - Performance:** Issues arose in terms of computing performance that was not seen in advance, a lager pre-study could have helped
Case 16	2009	COTS over in-house	o
Case 22	N.A.	COTS over in-house	o
OSS			
Case 3	2004	OSS over COTS	√: **Financial - Cost:** Cost of the chosen solution was low
Case 7	2006	OSS over COTS	o
Case 15	2013	OSS over COTS and in-house	√: **Financial - Cost:** Very successful with regard to cost reduction by a factor of 10
Outsource			
Case 1	2008	Outsource over COTS and in-house	† : **Product, financial, as well as project criteria:** Sub-optimal solution with respect to criteria considered in preparation
Case 12	2005	Outsource over in-house	†: **Product - Performance:** Underestimated computing resources needed (performance) of the chosen solution
Case 19	2015	Outsource over in-house and services	o
Case 20	N.A.	Outsource over in-house	†: **Product - Quality (general):** Quality issues arose
Service			
Case 14	2012	Services over in-house	†: **Product - Quality (general):** Solution not as successful as hoped for, quality issues cost time to market, service needs to be replaced.
Combinations			
Case 2	2012	COTS and Services over OSS	o: **Project - Level of Support, Financial - Cost:** Trade-off between level of support (safer solution), but with a higher cost
Case 5	2014	COTS, Services and Outsource over In-house	†: **Decision making process:** Challenging to estimate and assess the impact and decision required long lead-time
Case 8	2014	COTS over OSS (first iteration) and OSS over COTS (second iteration)	†: **Decision making process:** Problems in details could not be foreseen (old version of programming language became an issue for chosen mature component)
Case 10	2007	In-house and COTS over an i ndividual option	√: **General:** Perceived as the right decision
Case 17	2013	In-house and Outsource over an individual option	√: **Project - Familiarity with technology:** Perceived as positive decision with regard to a combination of in-house and outsource in terms of available competence and responsibility

The case number is shown. The year of decision shows the year in which the decision was taken. The evaluation of the outcome was based on perception. We show the assessment of the subjects in terms of whether the decision was positive, they were indifferent, or negative. Furthermore, the rationales for the assessment provided by the subjects have been stated.

shared with the literature review by Badampudi et al. [9], namely options considered (RQ1.1), criteria considered (RQ1.3), and decision making method (RQ1.4). Stakeholder roles were not explicitly discussed in the primary studies included by Badampudi et. al., and hence are new results provided by the case survey in the context of choosing CSOs.

Options Considered (RQ1.1). The contributions of existing literature and case survey are mapped as shown in Table 16. The primary studies include empirical comparisons between CSOs and solution proposals of how to choose among them. Even though there are comparisons, no decision making solutions assist the COTS versus OSS decision.

TABLE 16
Mapping of Existing Literature and Case Study Contributions with Respect to CSOs: The Table Shows the CSOs Considered, the Number of Studies, as Well as the Number of Cases

CSOs considered	Number of primary studies		Number of cases
	Comparisons	Solutions	
COTS versus OSS	6	0	7
In-house versus COTS	2	6	9
In-house versus Outsource	0	2	13
In-house versus OSS	1	0	2
COTS versus Outsource	0	0	5
OSS versus Outsource	0	0	2

Comparisons refer to studies investigating the properties of the CSOs in comparison to each other. Solutions refer to proposed approaches to decide between CSOs. The number of cases show how many times the CSOs were considered in comparison to each other in our case survey.

The case survey reported six cases that consider COTS and OSS in the decision.

While there are no primary studies comparing in-house and outsourcing, two papers propose decision making solutions for the selection of CSOs. Overall the number of primary studies (two papers) discussing in-house and outsource is low despite being the most frequent decision as revealed by our case survey (11 of 22 cases).

In-house and COTS have also been compared in primary studies identified in our systematic literature review [9], and decision making solutions are proposed to make in-house versus COTS decisions. The frequencies of the primary study and case survey are consistent for the in-house versus COTS options.

COTS versus outsource and OSS versus outsource are neither compared nor any decision making solutions are proposed. Therefore, based on the existing literature, it is unknown whether such decisions are even considered by decision makers. However, the case survey identify cases where COTS versus outsource and OSS versus outsource decisions are considered. There is thus a need for better support in practice.

The number of times options were chosen in comparison to the times they were considered indicates that all options are viable choices for the software-intensive systems considered. In-house has been selected in 41 percent of all cases it was considered (7/17), COTS in 50 percent (7/14), Outsource in 55 percent (6/11), OSS in 83 percent (5/6) and Services in 40 percent (2/5).

Stakeholders (RQ1.2). In most cases management roles initiated the decision, while the decision making preparation usually involved several roles from management, software construction and development, but also software design. Also, consultants were used to support the preparation of the decision, which indicates that additional competencies compared to those in the organization are needed to make the decision.

Criteria (RQ1.3). We first look at trade-offs between criteria, and compare them to the trade-offs discussed in our systematic literature review [9].

Trade-Off 1. Trade-offs between market-trend, technical support and maintainability are observed in the literature (cf. [9]). Following market trend indicates frequent updates which involves additional maintenance effort. At the same time, the need for a high pace in releasing new features is required, which may result in technical debt as shortcuts may be taken. Also, if the component is not upgraded to the latest available version, then the support offered from supplier/vendors might not be extended. Hence, a trade-off needs to be made between following market trends, maintaining the system stability and retaining the technical support. In the case survey 14 cases consider maintainability, five cases consider technical support, and one case considers market-trend. This means that maintainability, technical support, and market-trend are not considered together in the decisions.

Trade-Off 2. Another trade-off identified in the literature is between source code availability, technical support, and license [9]. The availability of source code might be a criteria for selecting a decision option so that the code can be changed. However, some licenses require changes in the code to be open. Also, technical support might not be extended for the modified code. Although source code availability (12) has high frequencies of cases, the frequencies of license (3) and technical support (5) are comparatively lower, which implies that the trade-off is not considered.

Trade-Off 3. Development effort and integration effort trade-off is identified in the literature [9]. Development effort can be saved if the development is not done in-house. However, if the saved development effort is less than the additional integration effort, then the decision is not optimal. No such trade-off is considered in any of the cases.

Overall the trade-offs observed in the cases and literature are not consistent, i.e., the existing studies do not support the processes observed in industry. This indicates a potential gap between industry practice and the focus of research with regard to sourcing decisions when looking at the trade-offs made. Overall, this merits the investigation of trade-off practices in the industry to support researchers in the selection of future research topics and questions with regard to trade-offs.

The need for prioritizing the criteria and identifying the important ones also became evident when analyzing the final decisions taken, and the criteria ending up as important among those considered. The decision problem may be simplified if criteria are identified and removed early. Much can be learned from the requirements community in that regard, in particular requirements prioritization techniques may be of value [59]. Barney et al.'s [60] study gives an overview of approaches for trade-offs between quality attributes, which may also be useful to not only trade-off between quality attributes, but by considering other factors as well.

Decision Making Methods (RQ1.4). The methods for decision making devoted to in-house versus COTS and in-house versus outsource exist in the literature [9], both trade-offs being frequently considered in the case survey. All the in-house versus COTS decision making solutions proposed in the literature consider technical factors, notably time, cost, and reliability. These factors are easy to calculate; probably this is the reason why the methods have considered them. Most of the cases in this case survey considered cost and

TABLE 17
Correlation with Effort: The Table Shows the Correlation with Effort in Relation to the People Involved as Well as the Complexity of the Decision Problem

Criteria	Spearman Correlation to Prep. Effort
Number of CSOs	0.559
Number of Decision Criteria	0.350
Number of People in Preparation	0.075

The complexity of the decision problem is indicated by two variables, the number of CSOs and the number of criteria considered.

reliability in their decisions. Time has not been considered that often. However, the case survey identified many other criteria that are considered in the decisions (which also include non-technical criteria), which are not included in the methods proposed in literature.

The methods for choosing between in-house versus outsource proposed in the literature consider requirement dependencies [9]. However, this is not identified as a criterion in any of the cases in the case survey.

All the decision making methods proposed in literature specifically for CSOs are automated and mathematical, i.e., they are highly formalized. However, the cases indicate that the most popular techniques used in making decisions are expert based, which involves subjective opinions. This indicates that the methods proposed in the CSO literature are not consistent with the practice in industry. This also suggests that practitioners are looking for decision making methods that aid in decision-making and not the solutions that give the decision/outcome. However, the solutions proposed for GDM in the context of architectures seem more appropriate (see Section 2.2.1).

Further, according to the case survey, the management takes most decisions. The decision making methods proposed in the literature are quite complex. Due to the complexity and learning curve involved, the managers might not accept or use the solutions proposed in the literature.

Effort Invested (RQ2.1). The effort invested in preparing the decision is quite significant, with the mean effort being 780 person hours, while in several cases the effort was over 1,000 hours. In order to understand the factors we investigated the correlation between the effort invested and the complexity of the decision problem (number of criteria and number of options considered). In addition we calculated the correlation between the number of people involved in the preparation and the effort. Table 17 shows the results of the correlation (using the non-parametric method proposed by Spearman). Rubin [61] defines boundaries for the strength of correlations. A strong positive correlation is observed for the number of CSOs, and a moderate positive correlation for the number of decision criteria. A lower correlation was observed with regard to the number of people involved. Given that the number of CSOs as well as the number of criteria seem to be related to effort in preparation we suggest a staged process for choosing among options to not invest preparation effort on more obvious exclusions. Similar reflections were presented in the field of requirements engineering, where it was found that more complex decisions take more time [62]. As a consequence requirements triage has been introduced that removes the most obvious options first to avoid investigative effort [63]. Thus, similar ideas may be relevant for choosing among CSOs. Even though the correlation between effort and the number of people was lower in comparison to the other measures (see Table 17) communication overhead during the preparation should still be considered as factor besides the number of CSOs and decision criteria as the overhead increases with the number of people involved.

Retrospective Reflection on CSO Choice (RQ2.2). In seven cases only, the result of the decision was perceived as positive. In particular, if high investments have been made in preparing the decision, and the final decision is not perceived as successful, then the preparation effort can be considered wasted. The methods used for decision making were mostly experience based, and no decision support systems or methods have been used (such as estimations, existing evidence from research); thus the results indicate that there is an industrial challenge relevant for the research to address. In particular, decision support systems aiding the experts may be of interest to design for this particular decision problem. Early attempts have been made and preliminary results are available in that regard (cf. Wohlin et al. [64]). Furthermore, the related work on architectural decision making provides solutions to record rationales and drive reflections in architectural decision making in Section 2.2.1 (e.g., [32], [33], [34]).

5.2 Characterization of Decisions

According to Larsson [11] case surveys focus on identifying patterns (similarities and differences) between cases. With regard to the research questions each individual question investigated a specific aspect of the decision making case. We used hierarchical cluster analysis, in particular clustering of binary data (presence and absence of variables or attributes in cases) for this purpose. The method to calculate the clusters was Squared Euclidean Distance. The clustering is used as a reflective tool. It takes the values obtained across research questions into account to determine the similarity and differences between the cases. We only included variables related to the decision case (CSOs, stakeholders, and decision criteria) to find commonalities of how decision making has been made. Fig. 10 shows the Dendrogram. It is evident that four cases are closely related, namely 18, 20, 21, and 22. Furthermore, cases 10 and 11 are very closely related. Overall, the Dendrogram shows that no clear patterns could be obtained as the distances between clusters were high. Using two-step clustering as an alternative approach showed that the shapes of clusters are not clearly identifiable, which is a sign that, besides the groups above, no clear patterns across cases are observable.

It has already been established that the cases share decision making characteristics, and a number of cases (more than two) is concerned, hence it is of interest to take a closer look at them. Hence, their contexts and outcomes are of interest to compare, which may explain why they are in the same cluster. The cluster of cases 18, 20, 21, and 22 is interesting because three of the four cases were perceived as positive. Despite of being in different domains (cases 18 and 22 are in the automotive domain and cases 20 and 21 are in the telecom domain) both of them are similar. The cases

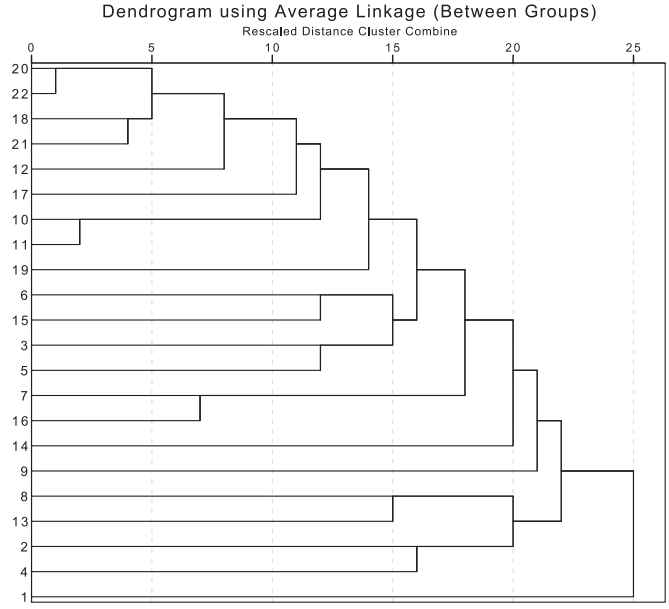

Fig. 10. *Hierarchical clustering of cases:* The Dendrogram shows the similarity and dissimilarity between cases to explore whether interesting or shared patterns emerged in the data. The more similar two cases, the closer the vertical lines are to the left-hand side of the figure. For example, cases 20 and 22 are more similar than cases 8 and 13. Only four cases clearly appear together in a cluster, namely 20, 22, 18, and 21.

characteristics, the context, and the outcome for the four cases in the cluster are briefly summarized in Table 18.

5.3 Comparison with the General Traits of Decision Making from Related Work

In the earlier sections we compared the findings of the case survey with the literature specific to each research question. The more general characteristics of decision making for architecture, CSO, and OTS decision making are presented in this section. We summarized eight characteristics of decision making in Table 19, and stated whether they are found in this case survey and provide reflections for the applicability of the findings. The table shows that two findings are not well supported, in particular L3, L6, L10, and L11. These show potential ways of improving industrial decision making (e.g., more explicitly considering risks and the experience and familiarity with the technology), while also may influence research solutions (e.g., providing solutions for CSO decision making that facilitate non-deterministic decisions). Solutions are present for architecture decision making and may be tailored to suit CSO decision making (see Section 2.2.1).

5.4 Validity Threats

Larsson [11] highlights a number of limitations related to the case survey method. Furthermore, threats described in Petersen and Gencel [65] are highlighted where applicable.

Generalizability. First, Larsson points out that a limited number of cases could be studied, given that a case survey extracts more detailed information than a survey. Furthermore, the available cases are limited and not easily accessible. In this study, a total of 22 cases have been obtained. As it can be seen from Table 6 different domains and application types were studied. The automotive and telecommunication domains are most frequently represented, which introduces a bias in the dataset. Ayala et al. [2] observed that there is an increasing adoption of OSS components over proprietary (in-house) solutions. In Ayala et al.'s study most of the experiences were reported from software consultancies. In our case survey study all sourcing options were considered, and also frequently chosen for all CSOs (as shown in Fig. 4). The difference in findings between the two studies indicates the need for further investigations.

Descriptive Validity—Factual Accuracy. There is a validity threat in that the coding is misunderstood and that data are not available/missing. Given that all researchers reviewed and iterated the instrument (see extraction scheme in Table 5) the risk of wrong interpretations of the scheme was reduced. Interviews were conducted over the phone, which allowed to clarify and ask follow-up questions to receive accurate answers. Another threat to factual accuracy is missing data. Effort data has been elicited for 16 out of 22 cases. Not all respondents were confident in being able to provide an

TABLE 18
Case Cluster: The Table Shows the Four Cases that Were Identified as Similar Based Through the Cluster Analysis

Attribute	Case 18	Case 20	Case 21	Case 22
Context				
Domain	Automotive	Telecom	Telecom	Telecom
Company size	10.000	100.000	100.000	3
Development unit size	10	100	5	3
Development method	Agile	Agile (Scrum)	Agile (Scrum)	N.A.
Decision case characteristics				
CSOs considered	In-house, Outsource	In-house, Outsource	In-house, Outsource	In-house, COTS
Stakeholders (initiation)	Software construction	Management	Management	Management
Stakeholders (preparation)	Management, Expert Group	Management	Management, Sales	Management
Stakeholders (decision making)	Management	Management	Management	Management
Criteria considered	Performance, maintainability, reliability, cost, and certification	Cost, certification	Performance, reliability, and certification	Functionality, Cost
Method	Expert judgment	Expert judgment	Expert judgment	Expert judgment
Outcome				
Decision impact	Positive	Negative	Positive	Indifferent
Effort	2400	640	480	N.A.

TABLE 19
Related Work Compared with the Case Survey: The Table Shows the Main Findings for the General Characteristics of Decision Making for Architecture, CSO Decision Making and OTS Decision Making

ID	Related work findings	Supported by case survey?	Reflections
		Architecture decision making	
L1	Architectural decisions types are structural, property and executive decisions.	Yes	In the case survey structural decisions were taken
L2	Architecture decisions are made in a group	Yes	Prepared in the group, reflections and rationales identified, but the actual decisions (final word) is made by individuals (mostly management)
L3	Tool support for group decision making, and the possibility to review decisions after they have been made	No	Not present and thus a source of improvement. Could have facilitated to collect more information in the case survey in comparison of what could be obtained
L4	Architecture decisions are not taken in isolation	Yes	Bundled related decisions using the same criteria and treated them as a single decision problem
L5	Decision making is non-deterministic and argumentative	Yes	No prescribed/ systematic decision making method used, rather discussion
		CSO decision making	
L6	Optimization solutions for CSO selection have been proposed assuming deterministic decision making	No	Decisions were based on discussions and without structured approaches (i.e., ad-hoc)
		OTS decision making	
L7	The solutions in the literature do not match the practice in companies	Yes	Non-deterministic approaches for CSO selection were used in the cases, while CSO literature (L6) assumes deterministic decision making.
L8	Decision making approaches are not formal, but rather ad-hoc	Yes	No well structured decision making method used (the most structured was Pugh-analysis, while otherwise the listing of pros and cons and less structured discussions took place)
L9	In OSS decision imitative is mostly taken by developers, while leaders (such as CEOs) have the final word	Partially	The initiation of the decision was mostly driven by management, the final word was mostly given by management (CEOs/CTOs)
L10	Risks are considered during the decision	No	Only the risk of cost increase was considered in one case
L11	Experience with the component is an important factor	No	Only in one case familiarity with the technology has been raised as an important criterion.
L12	Developers move towards OSS development	Partially	In 5 of 6 cases where OSS was considered it was chosen. Though it was not the default choice by companies, and they invested substantially to assess alternative options.

It also indicates whether the finding is supported by the case survey, distinguishing between fully, partially, and not supported. A reflection for the degree of support is also stated.

accurate number, thus six cases did not include the effort. We aimed at including only reliable data where the subjects felt confident in the information they were providing.

Theoretical Validity—Confounding Factors and Controllability. It would be desirable to make an inference from the characteristics of the decision making process to the success of the decision. Reflections related to the relationship between the two (process and success) are prone to confounding factors. Of particular interest are the context characteristics of the cases (see Table 6) where we captured size of the development unit developing the system, the domain, the application type, and the development methodology used in projects. In the case survey we did not identify additional context factors that may be of relevance. Though, looking at the literature a variety of factors may play a role (cf. Carlsson et al. [66])), such as organizational and team complexity, organizational models, developer experience with respect to the project, and individual development practices used (such as pair programming). Given that the practitioners only considered a subset of the contextual factors and criteria presented by Carlsson et al. the need for providing a more systematic approach of integrating context information into decision making is highlighted.

Interpretive Validity—Objectivity of the Researcher. Based on the findings from the survey conclusions and recommendations to practitioners are provided. The recommendations follow the data, and there is a risk that an individual researcher draws biased conclusions. The risk is reduced due to the number of authors involved in the study who provided their input to the reflections and key findings.

Interpretation and Coding of the Data. Distakes and potential bias are probable when interpreting and coding a large dataset. The coding activity is similar to what would be conducted in a systematic literature review when coding

extracted data from papers. Kitchenham et al. [67] recommends to peer-review the coding. Consequently, the second and third authors of the paper reviewed the coding done by the first author.

Repeatability. A data extraction form and the GRADE taxonomy [52] support the data extraction increases its objectivity, though a threat to the repeatability of the results remains due to the characteristics of the research.

6 Conclusion

In this paper we investigated how CSOs are chosen by conducting a case survey supported by Larsson's guidelines (cf. [11]). We identified and described 22 decision cases for choosing CSOs in the contexts of software-intensive systems. The CSOs were based on the experience of experts participating in a research project for choosing CSOs (ORION), and interviews with industry practitioners. 11 cases were based on experiences of the research project members, and 11 based on interviews.

All options considered were viable and have been selected by the practitioners, which includes In-house, COTS, OSS, Outsourcing or combinations of them. We found a mismatch between industry and practice. In particular, CSO related solutions for decision making were mostly deterministic, while decisions in practice were non-deterministic. Thus, solutions proposed for supporting decisions in the context of architecture in general are of interest for practitioners, as those take non-deterministic decision making into account. One example is the repository grid technique. Furthermore, solutions to record past decisions in a systematic way are of interest to gain deeper insights of what decisions can be reused in similar contexts. Such learning is important as in many cases the decision made was not perceived as successful. We found that recommendations in the literature were not followed, such as considering risks and experiences and the familiarity with the technology as explicit criteria. These are potential avenues for improvements. Also, decision making approaches were mostly ad-hoc and not well structured.

A common reason for the decision to be perceived as negative were issues with the quality, such as performance and security. Means for an early assessment and estimation of these properties are of use. A small set of cases used simulators.

Future Work. In summary, future work needs to focus on the following avenues:

- Conduct more in-depth empirical studies of how CSOs are chosen in industrial practice. In particular complementary case studies and large-scale surveys are of interest.
- Provide support for group decision making and consensus building as in the final decision aspects of the investigation (such as criteria and recommendations) were not followed, and the final word for the decision lied with individuals (mostly management). In particular, the tailoring of existing solutions for software architecture GDM is interesting.
- Provide tools to systematically capture and thus identify reusable decisions and create an evidence base for CSO decision making.

Acknowledgments

The work is partially supported by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden.

References

[1] P. Kruchten, "An ontology of architectural design decisions in software intensive systems," in *Proc. 2nd Groningen Workshop Softw. Variability*, 2004, pp. 54–61.
[2] C. Ayala, Ø. Hauge, R. Conradi, X. Franch, and J. Li, "Selection of third party software in off-the-shelf-based software development—An interview study with industrial practitioners," *J. Syst. Softw.*, vol. 84, no. 4, pp. 620–637, 2011.
[3] M. M. Gerea, "Selection of open source components-a qualitative survey in norwegian it industry," M.Sc. thesis, Dept. of Computer and Information Science, Norwegian University of Science and Technology (NTNU), Trondheim, Jun. 2007.
[4] J. Li, R. Conradi, C. Bunse, M. Torchiano, O. P. N. Slyngstad, and M. Morisio, "Development with off-the-shelf components: 10 facts," *IEEE Softw.*, vol. 26, no. 2, pp. 80–87, Mar./Apr. 2009.
[5] M. Torchiano and M. Morisio, "Overlooked aspects of COTS-based development," *IEEE Softw.*, vol. 21, no. 2, pp. 88–93, Mar./Apr. 2004.
[6] H. Holmström, E. Ó. Conchúir, P. J. Ågerfalk, and B. Fitzgerald, "Global software development challenges: A case study on temporal, geographical and socio-cultural distance," in *Proc. 1st IEEE Int. Conf. Global Softw. Eng.*, 2006, pp. 3–11.
[7] B. Ulziit, Z. A. Warraich, C. Gencel, and K. Petersen, "A conceptual framework of challenges and solutions for managing global software maintenance," *J. Softw.: Evol. Process*, vol. 27, no. 10, pp. 763–792, 2015.
[8] R. Britto, D. Smite, and L. Damm, "Software architects in large-scale distributed projects: An Ericsson case study," *IEEE Softw.*, vol. 33, no. 6, pp. 48–55, Nov./Dec. 2016.
[9] D. Badampudi, C. Wohlin, and K. Petersen, "Software component decision-making: In-house, OSS, COTS or outsourcing—A systematic literature review," *J. Syst. Softw.*, vol. 121, pp. 105–124, 2016.
[10] I. Groher and R. Weinreich, "Collecting requirements and ideas for architectural group decision-making based on four approaches," in *Proc. Eur. Conf. Softw. Archit.*, 2015, pp. 181–192.
[11] R. Larsson, "Case survey methodology: Quantitative analysis of patterns across case studies," *Academy Manage. J.*, vol. 36, no. 6, pp. 1515–1546, 1993.
[12] T. Gorschek and C. Wohlin, "Requirements abstraction model," *Requirements Eng.*, vol. 11, no. 1, pp. 79–101, 2006.
[13] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. d. M. S. Neto, Y. C. Cavalcanti, and S. R. de Lemos Meira, "Twenty-eight years of component-based software engineering," *J. Syst. Softw.*, vol. 111, pp. 128–148, 2016.
[14] J. Li, R. Conradi, O. P. Slyngstad, M. Torchiano, M. Morisio, and C. Bunse, "A state-of-the-practice survey of risk management in development with off-the-shelf software components," *IEEE Trans. Softw. Eng.*, vol. 34, no. 2, pp. 271–286, Mar./Apr. 2008.
[15] T. Helokunnas, "The dimensions of embedded COTS and OSS software component integration," in *Product Focused Software Process Improvement*. Berlin, Germany: Springer, 2002, pp. 509–518.
[16] J. Li, R. Conradi, O. P. N. Slyngstad, C. Bunse, M. Torchiano, and M. Morisio, "An empirical study on decision making in off-the-shelf component-based development," in *Proc. 28th Int. Conf. Softw. Eng.*, 2006, pp. 897–900.
[17] K. J. Stewart, A. P. Ammeter, and L. M. Maruping, "A preliminary analysis of the influences of licensing and organizational sponsorship on success in open source projects," in *Proc. 38th Annu. Hawaii Int. Conf. Syst. Sci.*, 2005, pp. 197c–197c.
[18] W. Chen, J. Li, J. Ma, R. Conradi, J. Ji, and C. Liu, "An empirical study on software development with open source components in the chinese software industry," *Softw. Process: Improvement Practice*, vol. 13, no. 1, pp. 89–100, 2008.
[19] P. Di Giacomo, "COTS and open source software components: Are they really different on the battlefield?" in *COTS-Based Software Systems*. Berlin, Germany: Springer, 2005, pp. 301–310.
[20] J. S. Norris, "Mission-critical development with open source software: Lessons learned," *IEEE Softw.*, vol. 21, no. 1, pp. 42–49, Jan./Feb. 2004.

[21] L. Brownsword, T. Oberndorf, and C. A. Sledge, "Developing new processes for COTS-based systems," *IEEE Softw.*, vol. 17, no. 4, pp. 48–55, Jul./Aug. 2000.
[22] T. Helokunnas and M. Nyby, "Collaboration between a COTS integrator and vendors," in *Software Quality*. Berlin, Germany: Springer, 2002, pp. 267–273.
[23] J. Li, F. O. Bjørnson, R. Conradi, and V. B. Kampenes, "An empirical study of variations in COTS-based software development processes in the Norwegian IT industry," *Empirical Softw. Eng.*, vol. 11, no. 3, pp. 433–461, 2006.
[24] B. Mishra, A. Prasad, and S. Raghunathan, "Quality and profits under open source versus closed source," in *Proc. Int. Conf. Inf. Syst.*, 2002, Art. no. 32.
[25] S. M. Syed-Mohamad and T. McBride, "A comparison of the reliability growth of open source and in-house software," in *Proc. 15th Asia-Pacific Softw. Eng. Conf.*, 2008, pp. 229–236.
[26] R. Torres, "Developer-led adoption of open source software libraries: A conceptual model," in *Proc. Eighteenth Americas Conf. Info. Syst. Proc. (AMCIS 2012)*, 2012, p. 34.
[27] S. A. Hissam and C. B. Weinstock, "Open source software: The other commercial software," in *Proc. 1st Workshop Open Source Softw. ICSE*, 2001, pp. 1–2.
[28] J. Li, et al., "Validation of new theses on off-the-shelf component based development," in *Proc. 11th IEEE Int. Symp. Softw. Metrics*, 2005, pp. 26–26.
[29] S. A. Hissam, D. Plakosh, and C. Weinstock, "Trust and vulnerability in open source software," *IEE Proc.-Softw.*, vol. 149, no. 1, pp. 47–51, Feb. 2002.
[30] J. Rudzki, K. Kiviluoma, T. Poikonen, and I. Hammouda, "Evaluating quality of open source components for reuse-intensive commercial solutions," in *Proc. 35th Euromicro Conf. Softw. Eng. Adv. Appl.*, 2009, pp. 11–19.
[31] U. van Heesch, P. Avgeriou, and R. Hilliard, "A documentation framework for architecture decisions," *J. Syst. Softw.*, vol. 85, no. 4, pp. 795–820, 2012.
[32] I. Malavolta, H. Muccini, and S. Rekha, "Enhancing architecture design decisions evolution with group decision making principles," in *Proc. Int. Workshop Softw. Eng. Resilient Syst.*, 2014, pp. 9–23.
[33] M. Nowak and C. Pautasso, "Team situational awareness and architectural decision making with the software architecture warehouse," in *Proc. Eur. Conf. Softw. Archit.*, 2013, pp. 146–161.
[34] O. Zimmermann, T. Gschwind, J. Küster, F. Leymann, and N. Schuster, "Reusable architectural decision models for enterprise application development," in *Proc. Int. Conf. Qual. Softw. Archit.*, 2007, pp. 15–32.
[35] R. Capilla, F. Nava, and C. Carrillo, "Effort estimation in capturing architectural knowledge," in *Proc. 23rd IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2008, pp. 208–217.
[36] M. Razavian, A. Tang, R. Capilla, and P. Lago, "In two minds: How reflections influence software design thinking," *J. Softw.: Evol. Process*, vol. 28, no. 6, pp. 394–426, 2016.
[37] D. Tofan, et al., "Empirical evaluation of a process to increase consensus in group architectural decision making," *Inf. Softw. Technol.*, vol. 72, pp. 31–47, 2016.
[38] M. Castellani, "Cognitive tools for group decision making: The repertory grid approach revisited," *Technol. Supporting Reasoning Communities Collaborative Decision Making: Cooperative Approaches: Cooperative Approaches*, 2010, pp. 172–192.
[39] S. Rekha and H. Muccini, "Suitability of software architecture decision making methods for group decisions," in *Proc. Eur. Conf. Softw. Archit.*, 2014, pp. 17–32.
[40] V. Cortellessa, R. Mirandola, and P. Potena, "Managing the evolution of a software architecture at minimal cost under performance and reliability constraints," *Sci. Comput. Program.*, vol. 98, pp. 439–463, 2015.
[41] V. Cortellessa, F. Marinelli, and P. Potena, "Automated selection of software components based on cost/reliability tradeoff," in *Software Architecture*. Berlin, Germany: Springer, 2006, pp. 66–81.
[42] P. Potena, "Composition and tradeoff of non-functional attributes in software systems: Research directions," in *Proc. 6th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.: Companion Papers*, 2007, pp. 583–586.
[43] V. Cortellessa, F. Marinelli, and P. Potena, "An optimization framework for build-or-buy decisions in software architecture," *Comput. Operations Res.*, vol. 35, no. 10, pp. 3090–3106, 2008.
[44] T. Kramer and M. Eschweiler, "Outsourcing location selection with soda: A requirements based decision support methodology and tool," in *Advanced Information Systems Engineering*. Berlin, Germany: Springer, 2013, pp. 530–545.
[45] T. Kramer, A. Heinzl, and K. Spohrer, "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*. Berlin, Germany: Springer, 2011, pp. 115–132.
[46] S. U. Khan, M. Niazi, and R. Ahmad, "Factors influencing clients in the selection of offshore software outsourcing vendors: An exploratory study using a systematic literature review," *J. Syst. Softw.*, vol. 84, no. 4, pp. 686–699, 2011.
[47] J. Xu, Y. Gao, S. Christley, and G. Madey, "A topological analysis of the open souce software development community," in *Proc. 38th Annu. Hawaii Int. Conf. Syst. Sci.*, 2005, pp. 198a–198a.
[48] M. Bergquist and J. Ljungberg, "The power of gifts: Organizing social relationships in open source communities," *Inf. Syst. J.*, vol. 11, no. 4, pp. 305–320, 2001.
[49] M. Morandini, A. Siena, and A. Susi, "Risk awareness in open source component selection," in *Business Information Systems*. Berlin, Germany: Springer, 2014, pp. 241–252.
[50] R. Land, L. Blankers, M. Chaudron, and I. Crnković, "COTS selection best practices in literature and in industry," in *High Confidence Software Reuse in Large Systems*. Berlin, Germany: Springer, 2008, pp. 100–111.
[51] T. Wanyama and B. Far, "An empirical study to compare three methods for selecting COTS software components," *Int. J. Comput. ICT Res.*, vol. 2, no. 1, pp. 34–46, 2008.
[52] E. Papatheocharous, K. Petersen, A. Cicchetti, S. Sentilles, S. M. A. Shah, and T. Gorschek, "Decision support for choosing architectural assets in the development of software-intensive systems: The GRADE taxonomy," in *Proc. Eur. Conf. Softw. Archit. Workshops*, 2015, pp. 48:1–48:7.
[53] R. K. Yin and K. A. Heald, "Using the case survey method to analyze policy studies," *Administ. Sci. Quart.*, vol. 20, pp. 371–381, 1975.
[54] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, vol. 398. Hoboken, NJ, USA: Wiley, 2013.
[55] J. Miller, "Statistical significance testing—a panacea for software technology experiments?" *J. Syst. Softw.*, vol. 73, no. 2, pp. 183–192, 2004.
[56] (2016). International software product management association (ISPMA), "SPM syllabus foundation level v.1.3," [Online]. Available: http://community.ispma.org/wp-content/uploads/2014/12/ISPMA-SPM-FL-Syllabus-V.1.3.pdf
[57] J. Strydom, *Introduction to Marketing*. Claremont, South Africa: Juta and Company Ltd, 2005.
[58] D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making techniques for software architecture design: A comparative survey," *ACM Comput. Surveys*, vol. 43, no. 4, 2011, Art. no. 33.
[59] P. Berander and A. Andrews, "Requirements prioritization," in *Engineering and Managing Software Requirements*. Berlin, Germany: Springer, 2005, pp. 69–94.
[60] S. Barney, K. Petersen, M. Svahnberg, A. Aurum, and H. T. Barney, "Software quality trade-offs: A systematic map," *Inf. Softw. Technol.*, vol. 54, no. 7, pp. 651–662, 2012.
[61] A. Rubin, *Statistics for Evidence-Based Practice and Evaluation*. Boston, MA, USA: Cengage Learning, 2012.
[62] K. Wnuk, J. Kabbedijk, S. Brinkkemper, B. Regnell, and D. Callele, "Exploring factors affecting decision outcome and lead time in large-scale requirements engineering," *J. Softw.: Evol. Process*, vol. 27, no. 9, pp. 647–673, Mar. 2015.
[63] A. M. Davis, "The art of requirements triage," *IEEE Comput.*, vol. 36, no. 3, pp. 42–49, 2003.
[64] C. Wohlin, K. Wnuk, D. Smite, U. Franke, D. Badampudi, and A. Cicchetti, "Supporting strategic decision-making for selection of software assets," in *Proc. 7th Int. Conf. Softw. Business*, 2016, pp. 1–15.
[65] C. Gencel and K. Petersen, "Worldviews, research methods, and their relationship to validity in empirical software engineering research," in *Proc. Int. Workshop Softw. Meas.*, 2013, pp. 81–89.
[66] J. Carlson, E. Papatheocharous, and K. Petersen, "A context model for architectural decision support," in *Proc. 1st Int. Workshop Decision Making Softw. ARCHit.*, 2016, pp. 9–15.
[67] B. Kitchenham, "Procedures for performing systematic reviews," Keele Univ., Keele, U.K., Tech. Rep. TR/SE-0401, vol. 33, no. 2004, pp. 1–26, 2004.

Kai Petersen received the PhD degree from BTH, in 2010. He is a professor in the Blekinge Institute of Technology (BTH), Sweden. His research focuses on software processes, software metrics, Lean and agile software development, quality assurance, and software security in close collaboration with industry partners. He has authored more than 70 research works in international journals and conferences.

Deepika Badampudi is working toward the PhD degree in the Blekinge Institute of Technology. Her research interests include component-based software engineering, agile software development, and evidence-based software engineering.

Syed Muhammad Ali Shah received the MSc degree in software engineering from the Blekinge Institute of Technology, Sweden, and the doctorate degree (PhD) in software engineering from Politecnico di Torino, Italy. He is a research consultant with Software Quality Systems AB, Sweden. He was senior researcher in the Software and Systems Engineering (SSE) Laboratory, Swedish Institute of Computer Science (SICS). Prior to this, he was an ERCIM (Marie-Curie) post-doctoral fellow with SICS. His research focus is on empirical software engineering, software testing and quality. He was research assistant in software engineering lab for approximately three years at Politecnico di Torino, Italy.

Krzysztof Wnuk is an assistant professor in the Software Engineering Research Group (SERL), Blekinge Institute of Technology, Sweden. His research interests include market-driven software development, requirements engineering, software product management, decision making in requirements engineering, large-scale software, system and requirements engineering and management and empirical research methods. He is interested in software business, open innovation, and open source software. He works as an expert consultant in software engineering for the Swedish Software Industry.

Tony Gorschek is a professor of software engineering in the Blekinge Institute of Technology-where he works as a research scientist in close collaboration with industrial partners. He has more than 15 years industrial experience as a CTO, senior executive consultant and engineer, but also as chief architect and product manager. In addition, he is a serial entrepreneur with five startups in fields ranging from logistics to internet based services and algorithmic stock trading. At present he works as a research leader and in several research projects developing scalable, efficient and effective solutions in the areas of Requirements Engineering, Product Management, Value based product development, and Real Agile and Lean product development and evolution. www.gorschek.com tony.gorschek@bth.se

Efi Papatheocharous received the BSc degree in computer science from the Department of Computer Science, University of Cyprus, in 2004, the MSc degree in advanced computer science with ICT management from the University of Manchester, in 2005, and the PhD degree in computer science from the University of Cyprus, in 2012. She is a senior researcher in the Swedish Institute of Computer Science (SICS). Prior to this, she was an ERCIM post-doctoral research fellow with SICS and a lecturer in the Department of Computer Science, University of Cyprus. She is currently carrying out core processes at the Software and Systems Engineering (SSE) Laboratory, SICS, and her research focus is empirical studies for systems and software engineering. Her primary research interests include software cost estimation, computational intelligence, agile software engineering, systems-of-systems, software ecosystems, and human factors in software engineering. Since 2006, she has been collaborating in a number of research projects at a National and European level and contributing to the research community with many papers published in peer reviewed books, international conference proceedings journals.

Jakob Axelsson (M'01-SM'05) received the MSc degree in computer science, in 1993, followed by the PhD degree in computer systems in 1997, both from Linking University, Sweden. He was with Volvo Technological Development, Göteborg, Sweden, from 1997-2001, and then moved to Volvo Car Corporation in the same city, where he was responsible for research and advanced engineering of electrical and electronic systems between 2001-2010, and also became a six Sigma Black Belt. In 2004, he became a part-time adjunct professor with Mälardalen University, Västerås, Sweden, and in 2011 a full professor at the same university. Since 2010, he has also been in the Swedish Institute of Computer Science (SICS), Kista, Sweden, where he is founder and director of the Software and Systems Engineering Laboratory. He is the author of more than 90 research publications. His research interests are focused on system architecture for embedded and cyber-physical systems, and system-of-systems engineering. He is a member of INCOSE, and has served as chairman of the Swedish section. He received best paper awards at the IEEE International Conference on Engineering of Computer-Based Systems in 2008, and at the Euromicro Conference on Software Engineering and Advanced Applications in 2014. He is a senior member of the IEEE.

Séverine Sentilles received the MSc degree in computer science with specialization in software engineering from the University of Pau and Pays de l'Adour, France, in 2006, and the licentiate and PhD degrees in computer science and engineering from Mälardalen University, Sweden, in 2009 and 2012, respectively. She is currently assistant professor in the School of Innovation, Design and Engineering, Mälardalen University. Her main research interests include component-based software engineering, model-driven engineering, and software quality (extra-functional properties).

Ivica Crnkovic received the MSc degree in electrical engneering, the MSc degree in theoretical physics, and the PhD in computer science, in 1991, all from the University of Zagreb, Croatia. He is a professor of software engineering with Chalmers University, Gothenburg, and Mälardalen University, Västerås, and a guest professor with the University of Osijek, Croatia. He is the director of ICT Area of Advance with Chalmers University. His research interests include component-based software engineering, software architecture, software development processes, and software engineering for large complex systems. He is the author of more than 200 refereed publications on software engineering topics, and guest editor of a number of special issues in different journals and magazines, such as the *IEEE Software*, and Elsevier the *Search Results Journal of Systems and Software*. He has been general chair of several top-level software engineering conferences (such as ICSE 2018, ECSA 2015, ASE 2014, Comparch & WICSA 2011, ESEC/FSE 2007,) and PC chair (COMPSAC 2015, ECSA 2012, Euromicro SEAA 2006, etc.). His teaching activities cover several courses in the area of Software Engineering undergraduate and graduate courses. From 1985 to 1998, Ivica Crnkovic worked with ABB, Sweden, where he was responsible for software development environments and tools. More information is available on http://www.ivica-crnkovic.net.

Antonio Cicchetti received the PhD degree in computer science from the University of L'Aquila with the thesis entitled "Difference Representation and Conflict Management in Model-Driven Engineering", in 2008. He is an associate professor in the IDT Department, Mälardalen University, Sweden. His research interests include the interplay of model-driven and component-based engineering techniques and their application in the development of industrial systems. Moreover, he investigates the general problems related to the design of modelling languages, mutiview systems, and model transformations, in the context of both academic research and industrial application. Further, he is interested in the concerns related to the management of evolution of both language and models. He can be reached at antonio.cicchetti@mdh.se. For more information see also http://www.es.mdh.se/~acicchetti/.