Information and Software Technology 112 (2019) 18–34

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Selecting component sourcing options: A survey of software engineering's broader make-or-buy decisions

Markus Borg[a,*], Panagiota Chatzipetrou[b,c], Krzysztof Wnuk[b], Emil Alégroth[b], Tony Gorschek[b], Efi Papatheocharous[a], Syed Muhammad Ali Shah[d], Jakob Axelsson[a]

[a] RISE Research Institutes of Sweden AB, Scheelevägen 17, Lund, SE-223 70, Sweden
[b] Blekinge Institute of Technology, Valhallavägen 1, Karlskrona SE-371 41, Sweden
[c] Örebro University School of Business, Örebro SE-701 82, Sweden
[d] iZettle, Regeringsgatan 59, Stockholm SE-111 56, Sweden

ARTICLE INFO

Keywords:
Component-based software engineering
Sourcing
Software architecture
Decision making
Survey

ABSTRACT

Context: Component-based software engineering (CBSE) is a common approach to develop and evolve contemporary software systems. When evolving a system based on components, make-or-buy decisions are frequent, i.e., whether to develop components internally or to acquire them from external sources. In CBSE, several different sourcing options are available: (1) developing software in-house, (2) outsourcing development, (3) buying commercial-off-the-shelf software, and (4) integrating open source software components.
Objective: Unfortunately, there is little available research on how organizations select component sourcing options (CSO) in industry practice. In this work, we seek to contribute empirical evidence to CSO selection.
Method: We conduct a cross-domain survey on CSO selection in industry, implemented as an online questionnaire.
Results: Based on 188 responses, we find that most organizations consider multiple CSOs during software evolution, and that the CSO decisions in industry are dominated by expert judgment. When choosing between candidate components, functional suitability acts as an initial filter, then reliability is the most important quality.
Conclusion: We stress that future solution-oriented work on decision support has to account for the dominance of expert judgment in industry. Moreover, we identify considerable variation in CSO decision processes in industry. Finally, we encourage software development organizations to reflect on their decision processes when choosing whether to make or buy components, and we recommend using our survey for a first benchmarking.

1. Introduction

Component-based software engineering (CBSE) is an established approach to enable large-scale code reuse and rapid development. By turning systems into assemblies of components, CBSE supports software evolution by simplifying component replacement [1]. However, in contemporary software engineering, the best option might not be to internally develop the new component. For example, buying commodity software components off-the-shelf might enable a faster time to market [2]. Furthermore, outsourcing development of less critical components could save the most knowledgeable internal development resources for differentiating features [3]. Moreover, it is increasingly common that software components can be reused within software ecosystems [4]. In this work, our interpretation of the term component is inclusive, i.e., a component is any separable software part of a system, from the database to "traditional" components as usually considered in CBSE, e.g., a software package, a web service, a web resource, or a module that encapsulates a set of related functions.

A recurring strategic consideration for organizations evolving component-based systems is the make-or-buy decision, i.e., whether to develop the components internally or to acquire them from external sources. Numerous studies from decisions in the manufacturing sector exist, e.g., structuring the decision process by providing support for cost identification and break-even analysis [5]. However, research on strategic decision making conducted in "traditional" manufacturing contexts does not necessarily apply to R&D projects, e.g., development of software-intensive systems. Kurokawa points out two main reasons [6]: First, as opposed to manufacturing projects, not only costs calculations are required for R&D projects, but also benefit calculations, i.e., an R&D organization can use acquired knowledge to generate revenue later. Second, in comparison to manufacturing, analyses of R&D make-or-buy options are subject to higher degrees of uncertainty, i.e., decisions must be made with less accurate estimates of costs and benefits.

In software engineering, the make-or-buy decisions are more complex as both making and buying are represented by several sourcing

[*] Corresponding author.
E-mail address: markus.borg@ri.se (M. Borg).

https://doi.org/10.1016/j.infsof.2019.03.015
Received 14 July 2018; Received in revised form 13 December 2018; Accepted 28 March 2019
Available online 29 March 2019

options [7]. “Making” a software component can be interpreted as traditional in-house software development. However, making can also mean developing the component as part of an open source software (OSS) strategy, making the source code available from the start with the goal to establish a community. An alternative is to carefully specify requirements and to outsource the development of the source code to an external organization, i.e., an option between “make” and “buy”. Finally, strict “buying” means purchasing a commercial-off-the-shelf (COTS) software component [8]. However, an increasingly common alternative to buying a COTS component is to instead integrate an existing OSS component [9], e.g., in operating systems [10], mobile applications [11], and even in safety-critical development contexts [12].

Deciding which component sourcing option (CSO) to use when evolving a software-intensive system is difficult. Often several stakeholders are involved in the decision making, and they might represent conflicting viewpoints [13]. Several highly advanced decision support systems have been proposed in software engineering research, e.g., Bayesian networks [14], formalism through modeling languages [15], and process simulation models [16]. Unfortunately, there is little available research on how practitioners make software engineering decisions, and even less on how sourcing decisions are made [17]. To address this, we present an industrial survey on practitioners’ decision making in relation to choosing between CSOs when integrating components in evolving software-intensive systems. Analogous to the case survey reported by Petersen et al. [18], we simplify CSO decisions to selecting one of the four alternatives:

- In-house: The company develops the component internally. In line with the work by Badampudi et al. [17], in-house includes any distributed development (incl. offshoring) and internal development by external consultants.
- Outsource: The company acquires the component from an external development organization, e.g., after bilateral contract negotiation or procurement via a competitive bidding process. Often the source code is part of the deal.
- COTS: The company buys an existing component from a software vendor or publisher. Typically the source code is not included in the deal.
- OSS: The company integrates an existing component that has been developed as open source software, possibly by a community. The source code is publicly available and the company might have to adapt it to fit the rest of the system.

We obtained 188 responses from various roles, across different domains, confirming that the phenomenon under study indeed exists in industry, i.e., CSO selection is a recurring decision point in software engineering. Furthermore, we show that CSO decisions are dominated by expert judgment, both in the actual decision making and in the assessment of component qualities. Finally, regarding component selection, we identified that functional suitability acts as an initial filter among candidate components, then reliability is the most important quality. Our main recommendation for industry practitioners is to increase awareness of how decisions are made internally in their organizations. Hopefully, our survey can let organizations benchmark against the state-of-practice in CSO decisions – thus enabling identification of improvements in the internal decision making processes. Finally, to meet the needs of industry practice, we call for academic researchers to focus efforts on how to support decision making that is mainly driven by expert judgment, rather than developing decision support of esoteric nature with limited practical value.

The rest of the paper is structured as follows: Section 2 presents related work on decision making in software engineering. Section 3 describes how we conducted the industrial survey. In Section 4, we present our results in the light of previous work. Section 5 answers the research questions and reports from a more thorough analysis. In Section 6, we discuss the primary threats to validity. Finally, Section 7 concludes our paper and presents our plans for future work.

2. Related work

This section reviews related work on two types of decision making in software engineering: CSOs selection and component selection.

2.1. CSO selection

Badampudi et al. [17] conducted a systematic literature review on approaches to choose between architectural assets, i.e., how to make trade-offs between different sourcing options. The investigation covered decision criteria, methods for decision making, and evaluations of the decision result. Through snowballing and systematic literature search, three types of solutions were identified to support the selection: (1) usage of decision methods, e.g., simulation models, analysis of requirements dependencies, components clustering, and decision tables, (2) usage of alternative criteria such as quality criteria, and (3) usage of alternative CSOs. The review highlighted that no systematic reviews exist on the topic of CSO selection whereas the CSOs compared were mainly focused on In-house vs. COTS and COTS vs. OSS. Furthermore, Badampudi et al. [17] analyzed the factors that are used in CSO selection, but they did not discuss the decision process involved – motivated by the limited number of case studies identified in the literature. In contrast, our survey captures a broad picture of decision making and we explicitly target the decision process in one of the research questions (cf. RQ2 in Section 3.1).

As only a limited number of reported case studies exist, Petersen et al. recently presented a case survey studying 22 case studies of how practitioners choose between CSOs [18]. The CSOs identified were: (1) in-house, (2) outsource, (3) COTS, (4) OSS, and (5) services, i.e., making use of services that are pre-built and can be invoked over a network, e.g., web services. One of the conclusions was that the most frequent trade-offs are carried out between in-house vs. COTS, in-house vs. outsource, and COTS vs. OSS, partly confirming the result of the Badampudi et al. study [17], and bringing forward the in-house vs. outsource option. Based on the outcome of the decisions made in Petersen et al. [18], the CSO in-house was the favorable decision option, however, the evaluation of the decision showed that many of the decisions were perceived as suboptimal, indicating the need for optimizing the decision making process and outcomes. This survey has been designed to partly overlap Petersen et al.’s case survey. The two studies differ in scope and detail, and enable both method and data triangulation – a recommended basis for knowledge discovery in software engineering [19]. The case survey discusses 22 decision cases in detail, whereas this survey collects high-level empirical data from a broad variety of respondents. Still, the RQs are similar enough to allow direct comparisons, and generalization from the 22 cases.

Several primary studies explored in-house vs. COTS CSO decisions, e.g., Brownsworth et al. [20], discussed the changes resulting from introducing COTS into the development process and presented a new process framework. These changes occur through simultaneous definition and inevitable trade-offs considering the requirements, marketplace, as well as architecture and design. The changes require not just an engineering or technical change to the typical (in-house) development process of requirements, architecture, and implementation, but also a business, organizational, and cultural change. Many new activities need to be carried out, e.g., vendor relationships establishment, COTS cost estimation, and license negotiation to leverage the benefits of a COTS marketplace. Li et al. [21] empirically identified new COTS-specific activities and roles integrated to traditional development to reduce risks and provide CSO process improvement. Two CSO processes were found popular in practice: (1) familiarity-based selection, and (2) Internet-based search with hands-on trials. In Cortellessa et al. [22], a framework was presented to support the decision to buy components or build them in-house for software architects. The framework presented is based on a non-linear cost/quality optimization model. A set of quality constraints related to delivery time and product reliability are used to estimate the amount of unit testing to be performed to build components. The main limitation of the approach is the difficult instantiation of the general model to specific cases.

Li et al. [23] studied decisions made during integration of COTS vs. OSS and showed significant differences and commonalities. The main rationale was to obtain shorter time to market and reduced development effort. COTS was expected to have higher quality and vendor support than OSS, whereas the no acquisition cost needed for the source code was the main motivation for choosing OSS, as well as the open-access source code benefit. On the other hand, maintenance costs were higher for COTS, as well as the required selection effort. For OSS the level of support was found questionable.

Considering in-house vs. outsource, Daneshgar et al. [24] discussed the factors affecting the decision process for CSOs for both SMEs and large organizations: requirements fit, cost, scale and complexity, commoditization/flexibility, time, in-house experts, support structure, and operational factors. The study further distinguished the factors for SMEs (ubiquitous systems, availability of free download, and customizable to specific government/tax regulations) and large organizations (strategic role of the software, intellectual property concerns, and risk). However, the small sample of companies investigated in the study (8 companies), limits generalizability. Wider-scope studies are needed, including SMEs across various industries and countries. The survey presented in this paper aims to collect data from more practitioners and companies, i.e., direct data that are current rather than based on historical cases, to attempt confirmation of recent work by Badampudi et al. [17] and Petersen et al. [18] as well as previous studies by other researchers.

2.2. *Component selection*

Once a component sourcing strategy has been selected, the organization needs to concretize the particular component to use. If the strategy is to do new development (either in-house or outsourcing), this will be handled in the development process chosen. However, in the case of OSS or COTS, there could be several different candidate component to choose between. In practice, a particular component could fit more or less well into the overall system architecture, and hence there is also an element of architecture decision making involved. In this section, we cover first results about architecture decisions with relevance to component selection. Then, the two particular cases of choosing OSS and COTS will be detailed.

Architectural decision making contains many challenges, as discussed by Tofan et al. [25]. Based on a survey with architects in industry, they identified that dependencies between different decisions and the large business impact are major difficulties. Decisions are often unique, and the analysis requires a large effort. van Vliet and Tang [26] collected literature related to the actual decision process that architects use, and they put perspectives on the rationality of that decision making, contrasting it with naturalistic decisions that are more contextually embedded. They conclude that the strategy chosen depends on how well-structured the problem is. They also identified sources of bias in the decision making, and discussed the phases of architectural decision making, including problem framing, design exploration, and solution identification. Axelsson [27] described a case study from the automotive industry, where the evolution of the system architecture was investigated as a result of a number of change requests. It shows how two processes interact, namely the revolutionary architecting of a brand new solution for future product lines, and the evolutionary architecting that handles smaller adaptations. The inclusion of a component into an existing architecture would be an example of an evolutionary step, which has a strong focus on interface alignment.

Relating to component selection, Ayala et al. [28] and Gerea [29] found that common steps are identification, evaluation, learning and knowledge management, use of the component, and choosing. Gerea also found that the process of selection is impacted by the component size. Larger components were selected earlier in the development life-cycle. For OSS, identification is a challenge, since there is a multitude of different places to look. Kokkoras et al. [30] attacked this problem using a federated search engine that queries a number of existing open source search facilities and aggregates the result. Once an OSS candidate has been identified, one type of analysis is to look at the business value [31]. In this approach, the net present value of the component can be compared to the discounted costs, where the value is based on the assessment of a number of non-functional properties relevant to the situation. However, this does not take into account the uncertainties that result from the ecosystem nature of OSS development, and therefore the approach is extended with a real-options analysis. Hauge et al. [32] interviewed software companies about the integration of OSS into systems. They concluded that project specific factors are more decisive than general evaluation criteria, thereby emphasizing the relation to architecture described in the previous paragraph. Also, the decisions tend to be satisficing rather than optimizing.

For COTS, Ayala et al. [28] found a gap in the processes for component selection proposed in the literature versus what is used in practice. For example, component repositories are proposed, but not often used. The process used for selection is rarely formal and rather ad hoc in nature, which has been reported by multiple authors (cf. Ayala et al. [28]; Li et al. [33]; and Torchiano and Morisio [34]). For COTS selection Li et al. [33] found that companies use prototyping to learn about COTS. In line with Tofan et al. [25], our survey reports the major challenges practitioners face when making architectural decisions, and, similar to Ayala et al. [28] and Gerea [29], we also attempt to capture the nature of the decision process. However, our work is primarily targeting CSO selection rather than component selection.

Finally, Jadhav and Sonar conducted a systematic literature review on selection of software packages [35], largely overlapping with what we refer to as COTS components. They report that the analytic hierarchy process has frequently been proposed as a solution to tackle package selection in industry, but that the main obstacle has been the challenge of defining clear evaluation criteria – which we specifically address in our related tool PROMOpedia [36].

3. Research methodology

This section describes the research questions, the design of the survey, the instrument evaluation, the data collection, and the data analysis.

3.1. *Research questions*

The goal of our survey is to understand how CSOs and individual components are selected in industry. More specifically, we contribute knowledge to architectural decision making [25], by decomposing the goal into the three specific Research Questions (RQs) listed below. In the list, we also present the mapping between the RQs and the questionnaire Questions (Q) described in Section 3.2.

RQ1 **Which CSOs are typically considered in industry?** Q9–Q13 investigates the main CSOs considered in industry according to previous work [17,18].

RQ2 **What is the decision process when selecting CSOs and components?** Influenced by previous work, Q14 explores the roles involved in decision making [13,37] and Q17 + Q20–Q23 address the nature of the decision process [38].

RQ3 **What component qualities are the most important input to the decision process?** In Q18, we use the classification of software quality from the international standard ISO/IEC 25010 [39].

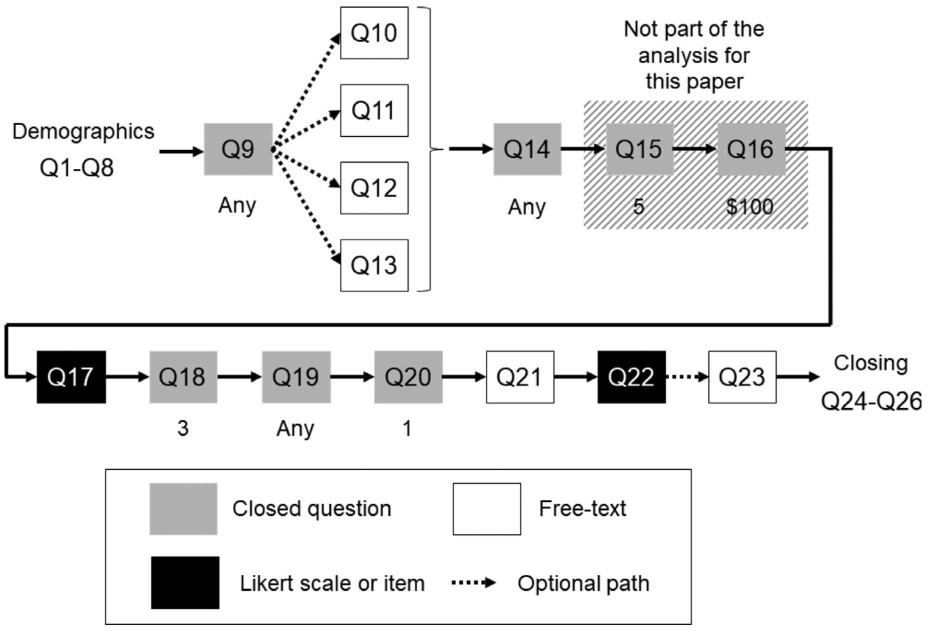

Fig. 1. Overview of the questionnaire. The numbers under the closed questions show whether one, three, or any number of options could be selected. The individual questions are listed in the appendix. Note that Q15–Q16 are not part of the analysis in this paper.

3.2. Survey design

We designed a structured cross-sectional web-based survey [40] and implemented it using the Querous Survey Platform[1]. A survey method allows for reaching a large number of respondents from geographically diverse locations [41] and enables both automation in data collection and flexibility in analysis [42]. We selected the Querous Survey Platform because it supports a more advanced question control flow compared to what is offered by simpler solutions such as Google Forms and SurveyMonkey.

The questionnaire consisted of a mix of closed-end and open-end (free-text) questions. The closed-end questions were of the following types: (1) select one option, (2) select multiple options (any number, up to three options, or up to five options), and (3) Likert scales. To distinguish between type (1) and (2) in Section 4, we present the former as vertical bar plots with percentages, and the latter as horizontal bar plots.

Definitions and clarifications were provided for those parts of the questionnaire for which there was a risk of misinterpretations. All questions included either an "Other" option with a free-text field or an "N/A" option. The final version of the questionnaire, containing 26 questions referred to as Q1–Q26, is available in the Appendix. Note that we designed the survey to allow also partial answers, i.e., any dropouts that at least answered Q9 contributed data to the subsequent data analysis phase.

Fig. 1 shows an overview of the questionnaire. Our target respondents were practitioners involved in CSO decision making in industry, including roles in strategic management (e.g., CTOs), product planning (e.g., product managers), operational management (e.g., project manager), and software architecture. As the target population is large and highly heterogeneous, we included a relatively large demographics section (Q1– Q8) to enable a detailed characterization of the respondents. We collected (1) the role, (2) working experience, and (3) level of education of the individual respondents, and (1) domain, (2) maturity and (3) size of the respondents' organizations, as well as characteristics of their software development processes, i.e., whether they use traditional plan-driven processes or rather adhere to agile practices.

The demographics section was followed by a pivotal question on which CSOs respondents consider (Q9), i.e., which of the four CSOs (In-house, outsource, COTS, and OSS). The subsequent questions Q10–Q13 only appeared to the respondent as a clarifying free-text question if the corresponding CSO was not selected, e.g., "What is the main reason for you not to consider the option OSS?" (cf. "optional path" in Fig. 1).

The next section of the questionnaire (Q14–Q22) collected the backbone data of the study. Q14 is a closed-end question for which any number of options could be selected. Q15 and Q16 have been analyzed in a separate publication [43], but for transparency and completeness the questions can be found in the appendix. Q17–Q19 are closed-end questions with up to three, any, and one possible selection, respectively. Q20 is a mandatory free-text question regarding the most important challenge involved in CSO decisions. Finally, Q21 is a single Likert item followed by Q22 as a free-text clarification if (and only if) "Strongly agree" or "Strongly disagree" is selected (i.e., an "optional path" in Fig. 1). Finally, the questionnaire concluded by a section of closing questions related to contact information and follow-up studies (Q23–Q25).

3.3. Survey instrument evaluation

We evaluated the questionnaire in two stages. In the first stage, the entire Orion research team[2] reviewed the questions. In addition, we invited an external senior software engineering researcher, a native English speaker, to particularly review the questions from a language perspective. We refined the survey instrument based on the feedback, covering wording, readability, understandability, and potential ambiguities. After the first stage, the questionnaire was implemented in the Querous survey platform.

In the second evaluation stage, we invited 15 colleagues from our partner networks to act as test pilots. We asked these pilot respondents, of which a handful had worked as senior product developers or managers in industry, to measure the time needed to complete the questionnaire, and to provide feedback on any unclear questions. The feedback from the pilot respondents led to the removal of 2 questions to ensure that 10–15 min would be sufficient to complete the survey. Moreover, some of the replies entered in "Other" categories by the pilot respondents were used to refine the answer options. The final version of the questionnaire consisted of the 26 questions presented in Fig. 1.

[1] http://www.querous.com/.

[2] http://orion-research.se/participants.html.

3.4. Data collection

We opted for an inclusive approach and used convenience sampling [44] to elicit as much information from industry practitioners as possible in relation to CSO and component selection. Previous empirical studies have suggested that both technical and management roles are involved in the decisions under study [13,18,37]. The roles identified in our previous work include: software management, software development, external support, software testing or quality control, customers, experts, legal, sales, software design and architecture, and subcontractors (component providers). The multitude of roles confirms that an inclusive approach is the most suitable for this survey, as our aim is to collect opinions from a broad spectrum of decision makers and industry representatives, i.e., the target population [45].

Data collection started on January 14th of 2016 and finished on August 31st of 2016. The majority of the responses was collected during January and February. The Orion research team was tasked to send direct invitations to industry partners, focusing on software architects and product managers, but we also asked those industry partners to circulate invitations within their organizations. Moreover, we advertised the survey on social media, e.g., Twitter and several LinkedIn and Facebook groups related to software engineering and in particular software architecture.

We kept track of the origin of the responses by sharing five separate invitation links, i.e., one per academic partner in the Orion project: Blekinge Institute of Technology, RISE, and Mälardalen University, one for the pilot responses, and one link for open invitations, e.g., LinkedIn, Twitter, and Facebook. The advantages of using LinkedIn in software engineering surveys have been discussed in the literature, e.g., Galster and Tofan [46], and include increased subject heterogeneity and the possibility to reach a population for which no centralized bodies of professionals exist. In total we collected 353 responses; 296 responses through direct invitations and 39 through open invitations, 15 pilot responses, and three undefined responses, i.e., responses that the Querous platform failed to track.

3.5. Data analysis

We started the analysis by filtering out invalid answers, i.e., nonsense or careless responses. All filtering steps were done by the first author and validated by the third author. In total we obtained 353 responses, of which 152 were complete (43%). As most of the responses from the test pilots were collected from respondents belonging to the target population, we agreed to keep all but two (collected from test pilots mainly inspecting the language). Regarding the partial responses, we decided to keep all that at least completed Q9, i.e., the question on which CSOs are considered, resulting in 188 remaining responses. The average completion time for respondents who completed the whole questionnaire was 20min and 2s (SD = 19 min 44 s) – 87% completed it within half an hour.

After the filtering, we analyzed all "Other" answers from closed-end questions, i.e., answers containing free-text, to investigate whether any answers should be consolidated with the existing possible options for the questions (i.e., Q1–Q4, Q9, Q14 and Q18–Q20). We decided to consolidate 13 answers for Q1 (respondents' roles), three answers for Q4 (respondents' domains), and two answers for Q18 (quality attributes), but this did not introduce any new answer options. As for the filtering steps, all merging operations were suggested by the first author and validated by the third author.

We conducted a number of statistical analyses within this study to answer the RQs. For the demographics section (Q1– Q8) contingency tables were used to explore frequency data [47]. All the results from the tables were depicted with bar charts. Chi-square of independence was performed to test the variety of the sizes of the different contingency tables, as well as more than one type of null or alternative hypotheses. The threshold value for p was 0.05 [48]. To examine the strength of associations we used Cramér's V test. Cramér's V is a measure of the strength of association of a nominal by nominal relationship, ranging in value from 0 to +1 representing no association and complete association, respectively. A value more than 0.5 indicates strong association, as suggested by Cohen's guidelines for interpreting Cramér's V [49].

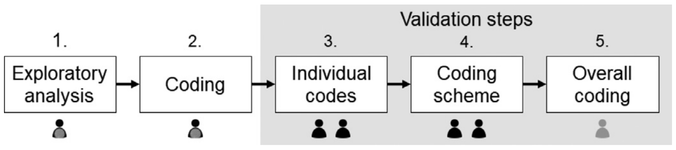

Fig. 2. The five steps of the qualitative analysis. The person icons reflect how four different researchers were involved.

For the free-text survey results (Q10–Q13, Q21, and Q23), coding analysis was performed, inspired by grounded theory [50] through the five step process depicted in Fig. 2. Step 1 was an exploratory analysis of collected quotes performed by one researcher, resulting in extraction of 125 relevant quotes. Step 2 was coding of the extracted quotes by one researcher, an incremental process with the goal summarize key concepts that resulted in 37 codes. During this process, instructions on how to interpret and apply the codes were captured in a coding scheme.

Steps 3–5 involved validation of the codes and the instructions for application. Step 3 was a validation of the codes, conducted by two authors other than the original creator of the codes, by analyzing a subset of quotes – resulting in the removal of one single code. Step 4 was a validation of how the codes should be applied according to the instructions, also conducted by the two authors involved in Step 3. Finally, Step 5 was a validation of the overall coding conducted by the researcher in Step 1. The final step was done by letting yet another author code a subset of 10 quotes, and we obtained an acceptable inter-rater reliability (Cohen's Kappa value of 0.62). All details of the qualitative analysis are reported in the accompanying technical report [51].

4. Results and discussion

This section presents the results from our survey and a discussion in the light of previous work.

4.1. Demographics

Fig. 3 shows the roles of the individual respondents. A third of the respondents primarily associate themselves as product developers, reflecting that the number of developers outnumbers other roles in industry. The second largest group of respondents are software architects (20.7%), which appears promising given our goal to better understand architectural decision making. Other roles represented by ten or more respondents are strategic management, product planning, quality assurance, and end-user perspective. Overall, the respondents represent a wide variety of roles involved in decision making.

Fig. 4 shows the respondents' working experience (a) and education level (b), respectively. A majority of the respondents reported 10 or more years of working experience (72.3%). Twenty-six respondents had more than 25 years of working experience and 10.6% of the respondents can be considered juniors with 0–4 years of working experience. Most of the respondents had received degrees from postgraduate education. We conclude that our survey covers the viewpoints of senior engineers in the software engineering industry.

Fig. 5 illustrates the wide variety of domains covered by the survey (note, however, that respondents could select any number of domains). The domain selected most frequently is by far "Computer [Software]". Other well-represented domains include telecommunications, engineering/architecture, automotive, and mobile applications.

The final part of the demographics section addressed the respondents' organizations. As a proxy for the maturity, Q6 asked for how

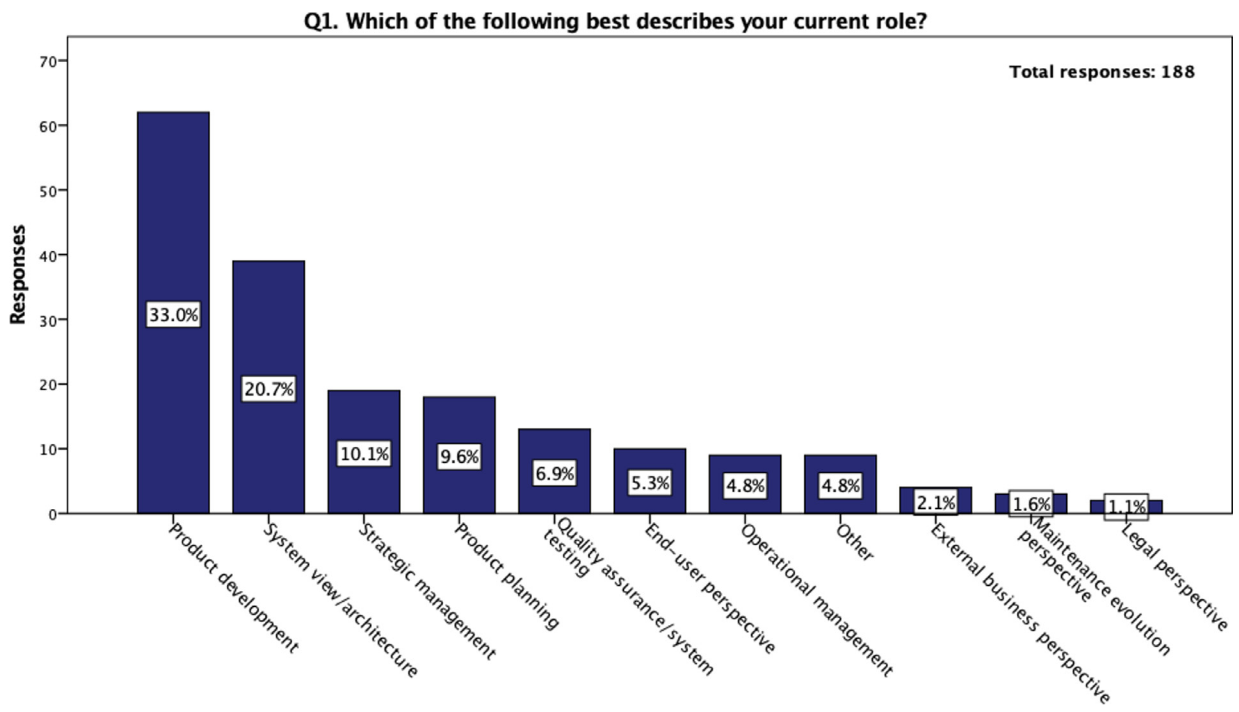

Fig. 3. Roles of the respondents.

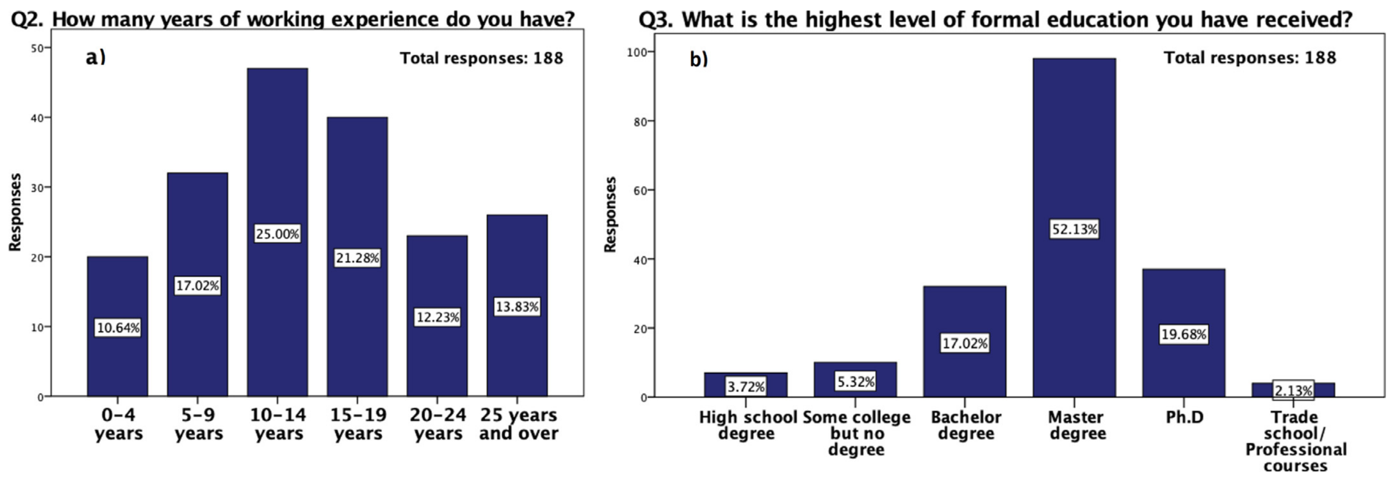

Fig. 4. Working experience (a) and education level (b) of the respondents.

many years the respondents' companies had offered products or services to the market. Fig. 6(a) shows that 31.9% of the respondents stated "25 years or over". On the other side of the scale, 23.4% of the respondents answered "0-4 years", representing companies new to the market. The median answer was "10–14 years". Fig. 7 depicts the size of the business units in which the respondents work. The most common size of the respondents' units is 5–19 co-workers (28.7%), but as many as 39 respondents (20.7%) work in companies that do not appear to break down to smaller business units, i.e., they have more than 500 co-workers.

Finally, our questionnaire gauged whether the respondents' development organizations adhere to an agile development methodology or rather traditional process models, e.g., waterfall development. As reported in Fig. 6(b), Q8 requested the respondents to select the level of agreement to the statement "My development organization is more agile than plan-driven". A majority of the respondents (58.0%) agreed or strongly agreed to the statement, while 26.6% disagreed or strongly disagreed. However, note that only 10 respondents strongly disagreed compared to 43 that strongly agreed. While our survey covers all levels of agility, we acknowledge that a larger fraction of the respondents adhere to agile practices.

4.2. Which CSOs are typically considered in industry? (RQ1)

Fig. 8 shows which CSOs are typically considered in industry, i.e., the answers to the branching question Q9[3]. A strong majority of the respondents (87.2%) consider in-house development when choosing between CSOs. The second and third most common CSOs are OSS (113 out of 188, 60.2%) and COTS (99 out of 188, 52.7%), respectively. We note that both OSS and COTS are considered in more than half of the responses. Outsourcing is the least commonly considered CSO, but still frequently considered as a viable option (68 out of 188, 36.2%).

[3] Note that more than one CSO could be selected, thus the percentages in this paragraph do not sum up to 100%.

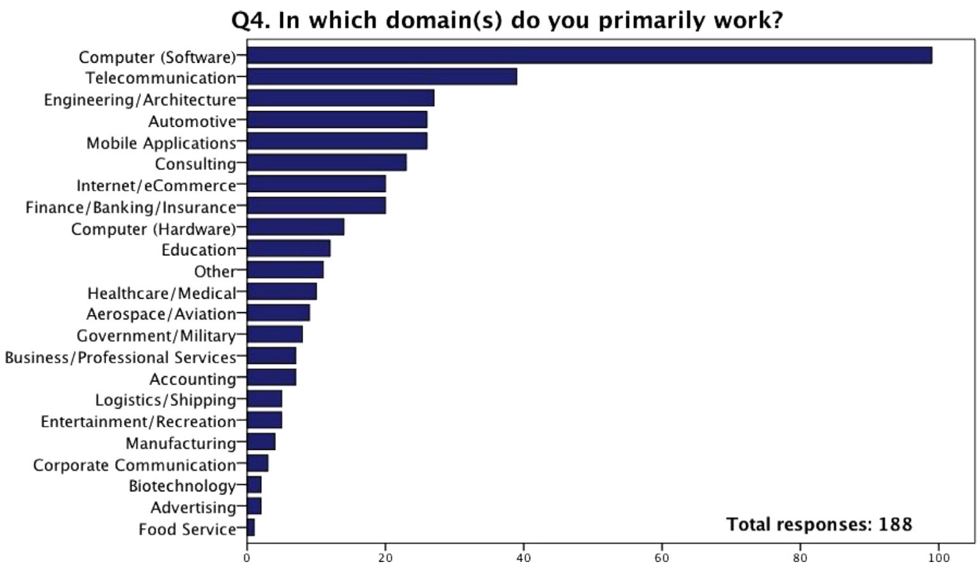

Fig. 5. Overview of the respondents' domains. Note that any number of domains could be selected.

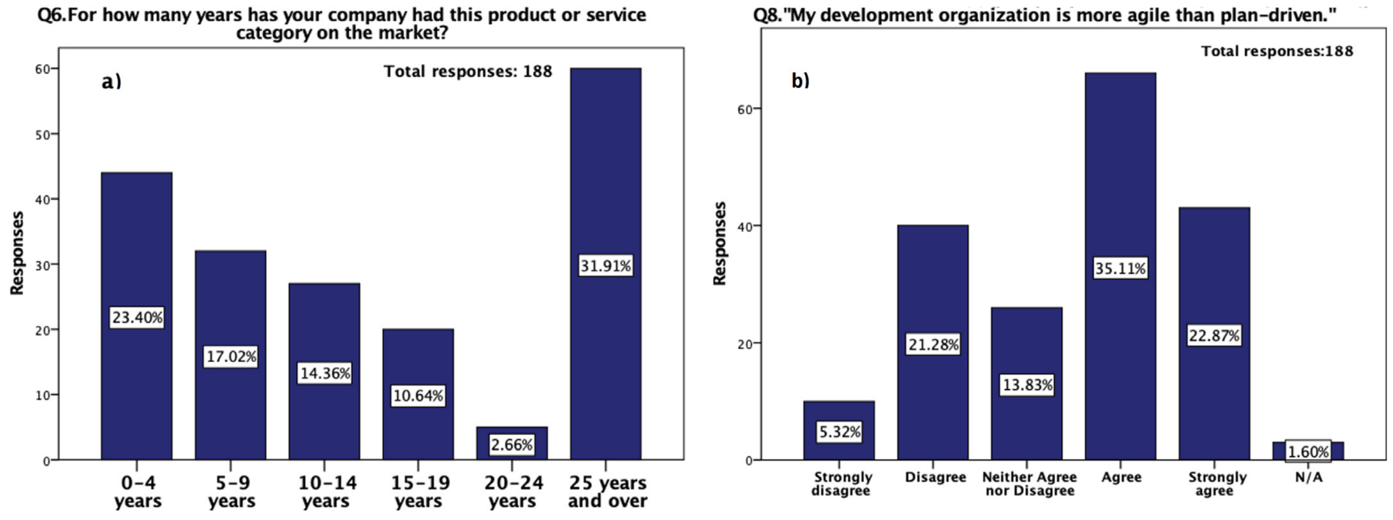

Fig. 6. Organizations' time on the market (a) and self-reported agility of the respondentsâ development organizations (b).

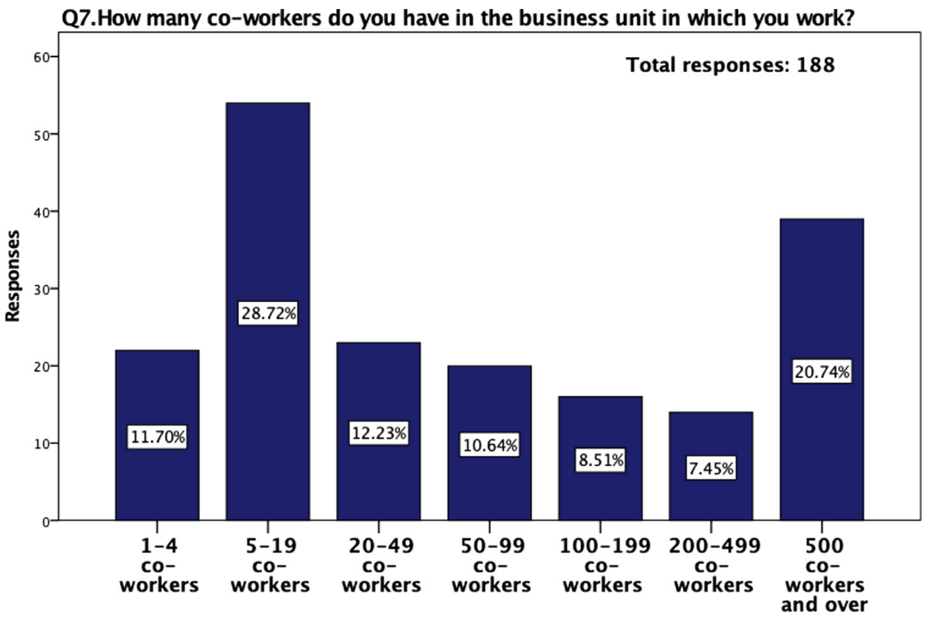

Fig. 7. Number of co-workers in the respondents' business unit.

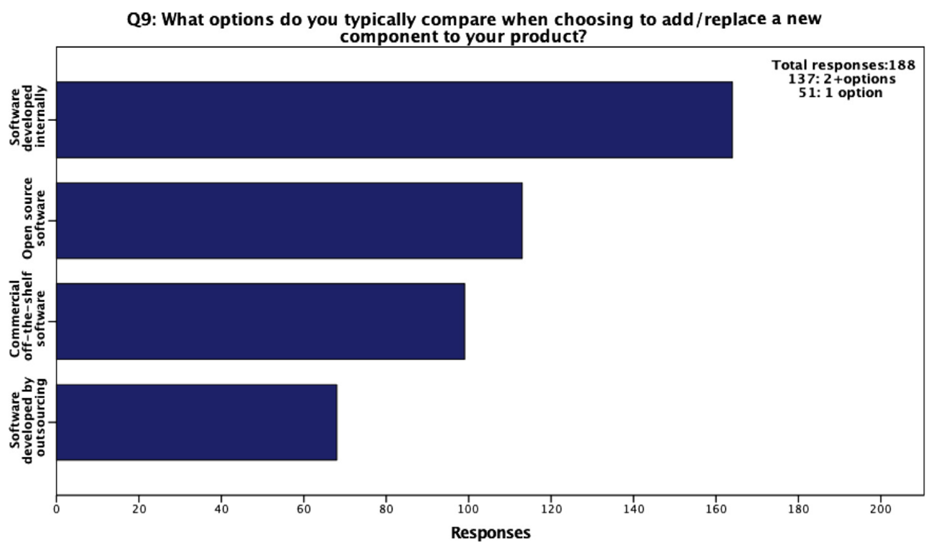

Fig. 8. The CSOs considered by the respondents. Number of CSOs selected by the respondents: One CSO = 51, Two CSOs = 51, Three CSOs = 53, and Four CSOs = 33.

In contrast to the case survey by Petersen et al. [18], our study suggests that OSS often is considered when practitioners compare CSOs when evolving a software-intensive system, i.e., we report 60.2% whereas Petersen et al. reported 11.3%. A possible explanation is that our sample has a larger representation from the domains mobile applications and Internet/e-commerce, which are known to frequently use OSS [52,53]. Another explanation, partly related, is that the previous case survey draws conclusions on older decision cases, whereas OSS has matured considerably in the last decade.

A majority of the respondents report that they typically consider two or more CSOs when adding new components (72.9%). The two CSOs that most frequently co-occur in decisions are in-house and OSS (97 times) and in-house and COTS (92 times), followed by in-house and outsource (59 times). Roughly a quarter of the respondents typically consider only one CSO, i.e., 51 respondents state that there is no decision between different CSOs in their organizations. Among these 51 respondents, 33 respond only in-house development, 12 respond only OSS components, five respond only outsourcing, and one respondent answered only COTS.

Each time a respondent did not select one of the CSOs, the questionnaire proceeded with a free-text question on why the CSO was not considered. For the four CSOs, we received 243 such motivations distributed as 14, 100, 70, and 59 corresponding to in-house, outsource, COTS, and OSS, respectively. In the next paragraphs, for each CSO, we explicitly enumerate the most common motivations. The minority who did not select the in-house option tend to refer to strategic decisions such as (1) only OSS should be used and (2) the organization does not employ any internal developers at all.

Several respondents shared strong negative opinions about outsourcing development. The most common arguments against outsourcing are related to low return on investment, i.e., (1) poor quality despite (2) high costs. Examples include "[our] experience of outsourcing doesn't mean you get better developers", "it takes more time to write a detailed specification than it takes to write the code yourself", and "outsourcing is expensive as it requires huge control". Two other important reasons are (3) the reluctance to decrease the low-level control of the development, and (4) the importance of keeping the knowledge of the source code in-house – supported by comments such as "we need direct control over the software and cannot compartmentalize it to an outsourceable task" and "previous outsourced modules /—/ created knowledge silos which hampered internal maintainability and extendability". Our findings are well in line with previous research listing challenges of outsourcing [3,54].

The main reasons why organizations do not consider COTS components appear to be that they are (1) costly, but still (2) do not fulfill all requirements. Several respondents express that their needs require tailored solutions, e.g., "COTS typically has a bad fit with our offerings" and "we need rather specialized parts" Another frequently reported issue with COTS is that due to the high costs involved, there is a (3) threat of vendor lock-in. Finally, analogous to arguments against outsourcing, several respondents highlight (4) the risks of future maintainability issues when the source code is not managed within the organization, i.e., a lack of low-level control.

It appears that (1) lack of OSS alternatives is the main reason for not selecting OSS components. Our study does not reveal to what extent the respondents have explored the OSS landscape before coming to this conclusion, but it motivates research on solutions that support practitioners to identify OSS components [30]. Several respondents explain that OSS is not an option for (2) regulatory and legal reasons, e.g., strict process requirements on security and safety, supported by statements such as "[a] strict development process must be demonstrated to authorities" and "legal aspects together with the lack of responsibility are challenges that need to be overcome." Furthermore, issues with (3) licensing incompatible with business models are mentioned, e.g., "the licenses around open source may prevent us from charging our customers", and (4) uncertainties in long-term maintainability of OSS components – the same reluctance to lose low-level control as is reported for COTS. In general, it appears that the reasons why organizations do not consider OSS remain the same as reported a decade ago [55]. Our findings thus contrast recent work by Ayala et al., who reported that licensing was not an issue for organizations considering OSS components, but rather the lack of available documentation [56].

4.3. What is the decision process when selecting CSOs and components? (RQ2)

Fig. 9 shows the roles (or perspectives) involved in the CSO decision process. The two roles most frequently involved are product development (62.8%) and system view/architecture (61.2%). Also product planning perspectives (48.4%) and maintenance/evolution perspectives (45.2%) are often involved in the decisions. Our results show that all of the roles presented as options to Q14 are relevant to CSO decisions, as even the least frequent answer (internal business perspective) is selected in 18.1% of the answers. Furthermore, our list appears to be rather comprehensive as the number of "Other" answers is low.

Q17 is a Likert scale consisting of five Likert items related to the character of the CSO decision process. Fig. 10 presents the answer distribution of the 164 remaining respondents, i.e., 24 respondents had dropped out. While the CSO decision processes in industry appear to vary, some general trends can be seen. Roughly the same amount of

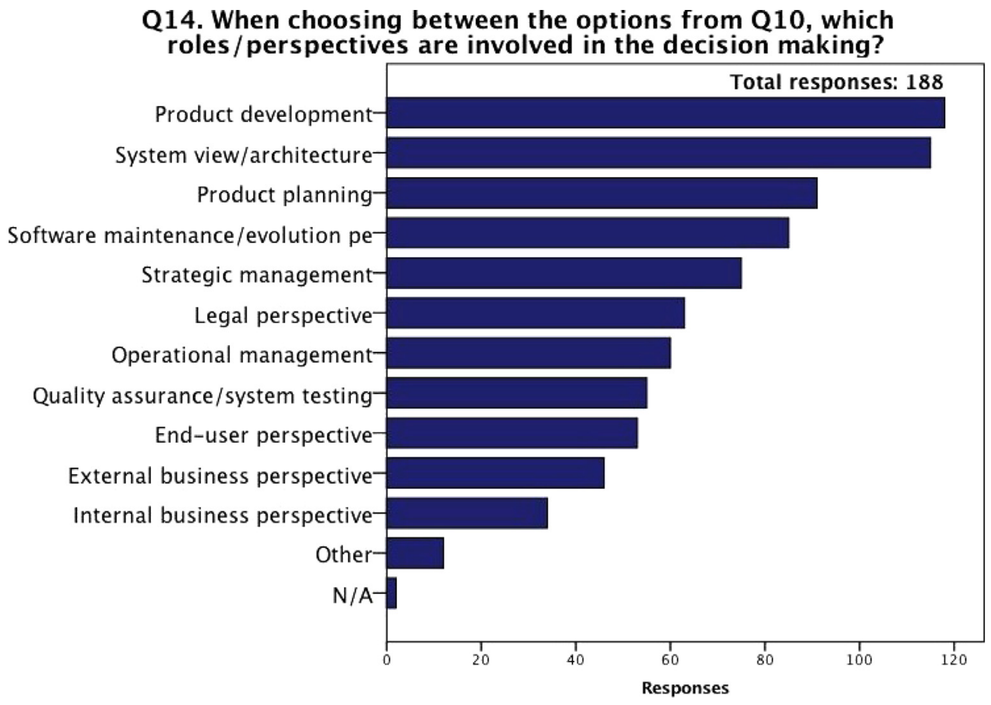

Fig. 9. Roles involved in the CSO decision process. Note that any number of roles could be selected.

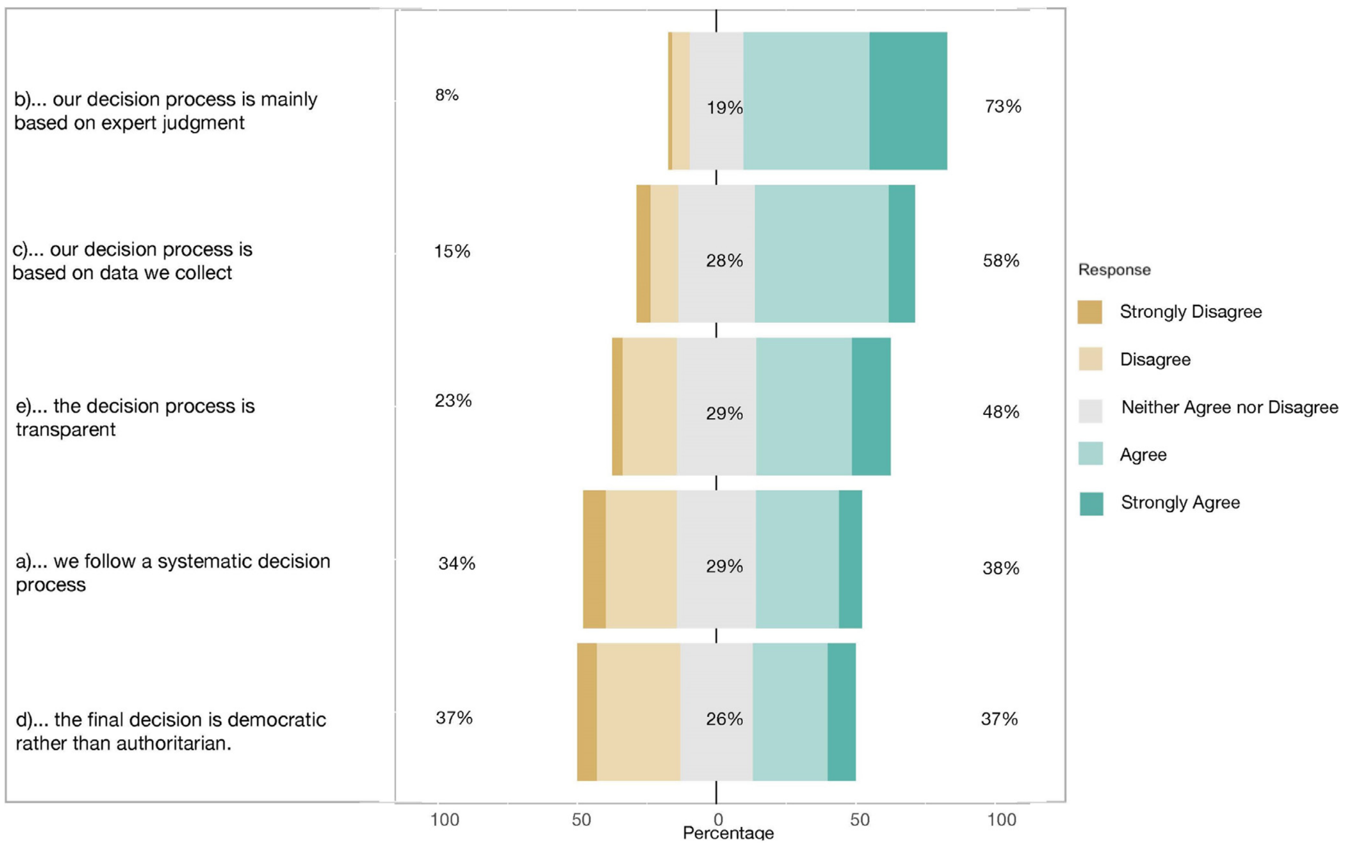

Fig. 10. Likert scale on the CSO decision process.

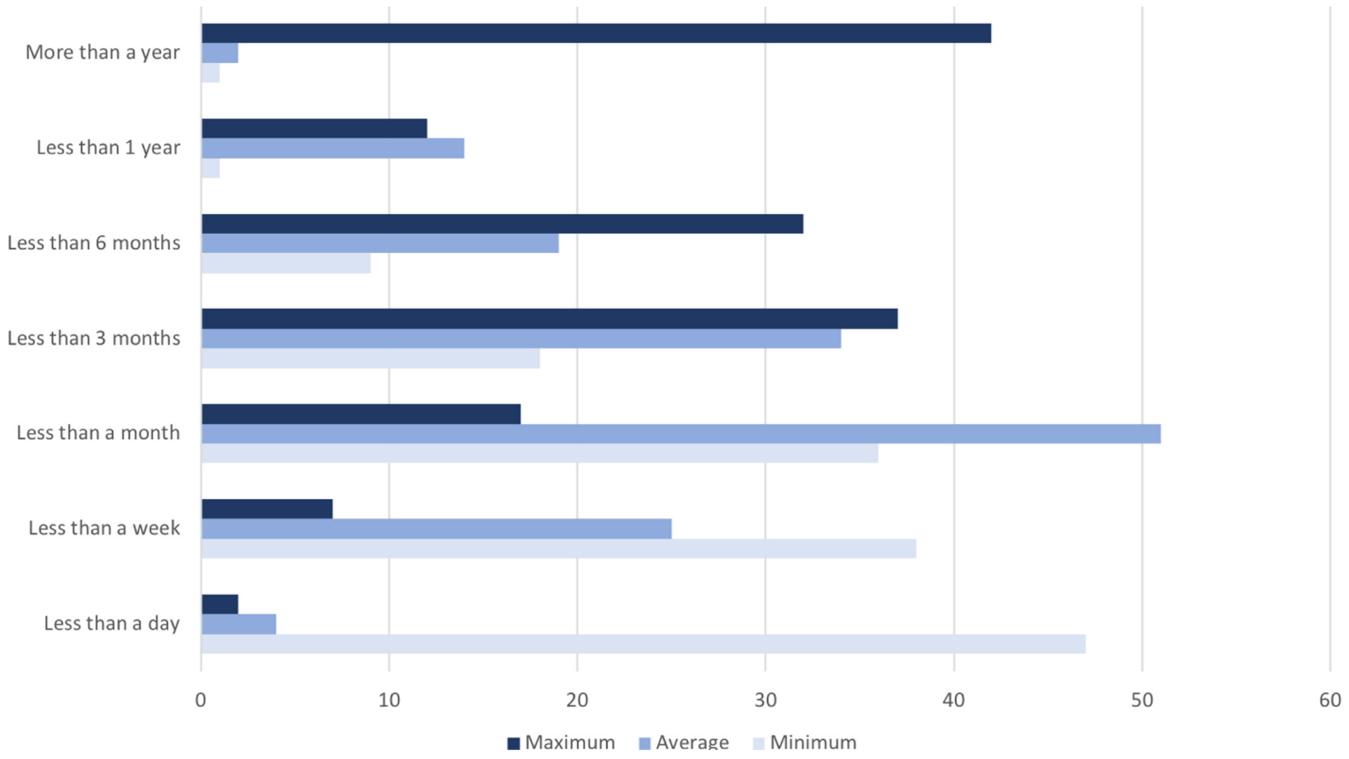

Fig. 11. Lead-time needed to reach CSO decisions.

respondents agrees (29.3%) and disagrees (25.0%) to whether the CSO decision process is systematic (cf. Fig. 10a). Only 26 respondents have strong opinions on the statement. Consequently, our results suggest that some organizations have a decision process in place while others do not, but few companies appear to have rigid processes established for CSO decisions, in line with previous studies on component selection [28,33,34].

Fig. 10b shows that CSO decision processes in industry are mainly based on expert judgment, as a clear majority agrees; 43.9% of the respondents agree and 26.8% strongly agree, but only 7.3% disagree or strongly disagree. In line with several other software engineering studies [18,57,58], expert judgment is the dominant approach. On the other hand, Fig. 10c shows a contrasting view: almost half of the respondents consider the decision process to be based on collected data – 47.0% of the respondents agree and 9.1% strongly agree. We interpret this as follows: expert judgment dominates CSO decisions in industry, but the experts inform themselves based on data collected within the organizations. Consequently, the typical CSO decision process in industry seems to be based on a data-driven approach to expert judgment.

Fig. 10d shows that CSO decision processes can be both democratic and authoritarian – the same number of respondents agreed (or strongly agreed) and disagreed (or strongly disagreed) to the statement. Regarding the transparency of the decision process, there is a positive tendency; Fig. 10e shows that 47.0% of the respondents agree (or strongly agree) that decisions are transparent, whereas 22.6% disagree (or strongly disagree). Clearly, the perception of group involvement and transparency in CSO decision processes is diverse.

Fig. 11 shows the lead-time needed to reach a CSO decision. The subplots represent answers from the 153 remaining respondents concerning the minimum time, the average time, and the maximum time, respectively. Fifty-one respondents (33.3%) report that the average lead-time for a CSO decision is less than one month. Both longer and shorter average times are common in industry though; 45.1% of the respondents report less than three months or longer lead-times, and 19.0% answered less than a week – or even less than one day.

Most respondents (55.6%) answer that the minimum lead-time is less than one week, 30.7% even say the minimum time is less than one day. Regarding the maximum lead-time, more than one year is the most frequent answer, reported by 27.5% of the respondents. A majority of the respondents (52.9%) selected alternatives corresponding to lead-times of the magnitude of months, i.e., less than three months, less than six months, or less than one year. Seventeen percent of the respondents claim that the maximum lead-time for CSO decisions is less than 1 month, possibly explained by less complex system development or leaner decision processes, as previous work report that increased product complexity lead to longer decision lead-times [59].

Q20 is free-text question about what makes CSO decisions challenging. We received 125 answers, and find that these challenges are aligned with the reasons why certain CSOs are not chosen, i.e., Q10–Q13 reported in Section 4.2. The results show a variety of reasons that could be divided into three key aspects: (1) management, (2) functional, and (3) quality-oriented, i.e., aspects that affect the feasibility of the candidate components, and in turn what CSOs were considered. Aspects associated with management include the cost of the component, the cost of its adoption and cost of component management, but also political factors, e.g., that OSS may not be allowed due to licensing issues. This conclusion is supported by statements like: “mostly legal issues regarding contracts, sourcing partners and SLA”, “We don’t know quality of complex open source. We don’t know how long open source will be maintained”, and “it can be hard to compare the time and costs in developing something of our own with the monetary costs in purchasing a finished product”.

Next, the functional aspects of a component seem to be particularly important for a decision maker to consider. This analysis varies from component to component and can prohibit the use of a certain CSO if the functionality is not good enough or if the component is not open source. This conclusion is supported by statements such as: “Our key factor is to correctly determine the strategic importance of the component, e.g., deciding where in the life-cycle it is and how quickly we believe the functionality will be commodity vs. differentiating” and

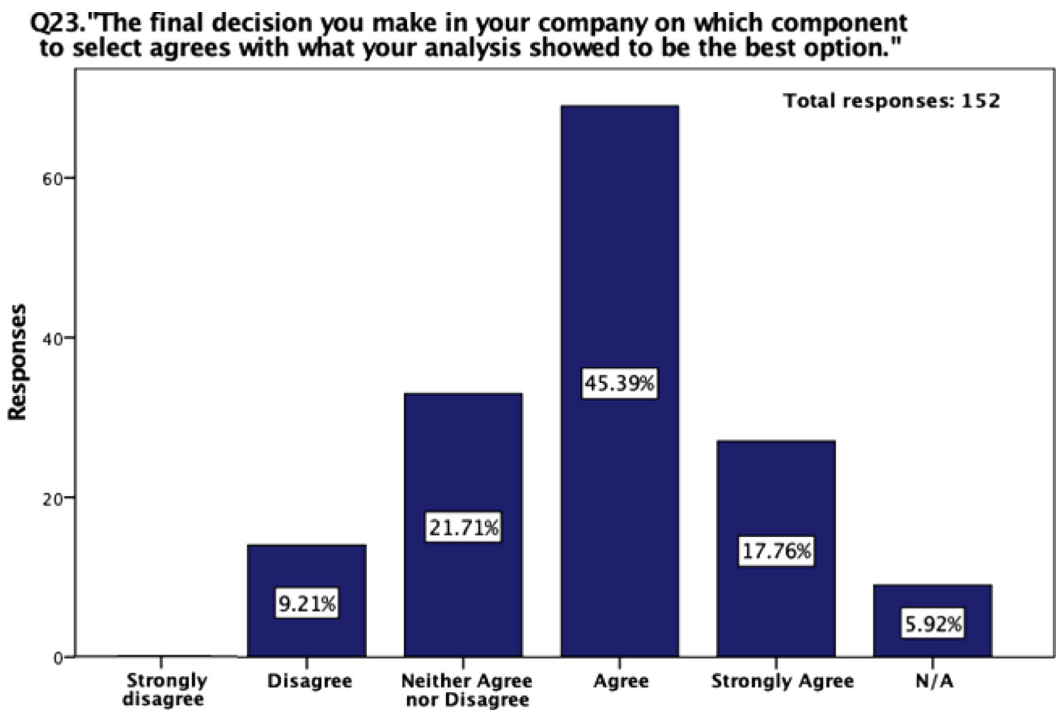

Fig. 12. Likert item on agreement between analysis and decision.

"acquiring technical knowledge needed to evaluate alternatives, and allocate resources to prototype concept proofs of concept or prototypes to test alternatives", which refers to the technical knowledge required to understand the viability of a certain asset.

Finally, the quality-oriented aspects are crucial since they are used to determine which component is chosen in the end, mentioned explicitly by 40 out of 188 respondents (21.3%). Hence, in the selection process, components of similar functionality are first identified and then the qualities of these components are compared to determine which one to select. As expected, the functionality is considered first to ensure that the component truly fulfills the needs of the decision maker. This conclusion is supported by statements like: "We test the product (asset) by integrating it with our product and calculate different function points and check performance. On the basis of the data collected from such tests a decision is made." and "Is it reliable and compatible with our product or not?"

Based on our findings, a general chain of decision making steps can be inferred, namely: (1) identification of components that follow the managerial and political guidelines of the organization, (2) identification of components of suitable functionality to fulfill the need of the organization, and, finally, (3) comparison of the quality aspects of the candidate components to acquire the one with the best fit for the organizational needs. Our results suggest that the component selection and the CSO selection are intertwined, i.e., any candidate component that is identified through the three steps can be selected, regardless of its CSO.

Our survey identifies many challenges, but they are diverse and vary across different types of companies and domains. Due to the diversity, it is difficult to determine any general challenges for certain contexts, which will instead require more detailed research in future work.

Fig. 12 shows 152 responses to the Likert item addressing the statement "The final decision you make in your company on which component to select agrees with what your analysis showed to be the best option." A majority of the respondents (63.2%) agree or strongly agree that the decision follows the analysis. While several respondents neither agree nor disagree to the statement, only 9.2% disagree with the statement. Also, not a single respondent strongly disagrees. Our results indicate that decisions indeed are made in line with what is recommended by analyses.

For respondents selecting "Strongly agree" to Q21, a free-text question appeared, requesting a motivation. We received 23 such motivations, and conclude that most companies have agreement because of their structured decision-process and often some type of democratic decision, supported by quotes such as "it has not been any big debates about it", "We prepared the choice through a formal process with a form that assists the decision. This form removes personal aspects and puts technical requirements that support in fact a complex decision", "unless someone can make a spectacular argument for an alternative, the team usually goes with what was democratic-ishly selected."

4.4. What qualities are the most important when selecting components? (RQ3)

Fig. 13 shows the most important quality attributes, as defined by ISO/IEC 25010 [39], when making CSO decisions and component selection. Functional suitability is of the highest importance, selected by 60.1% of the respondents, followed by reliability (42.0%) and maintainability (34.0%). On the contrary, portability was only selected by 7.4% of the respondents.

Fig. 14 presents how the ISO quality attributes are estimated prior to CSO decisions. Most organizations use internal expert judgment (68.1%), clearly the dominant approach used in industry. Five other common estimation methods appear to be equally common: previous experience (47.3%), perform measurements (44.1%), prototype component (43.1%), perform pre-study (42.6%), and read up on subject (42.0%). Considerably less frequent estimations methods are asking source providers, asking component users, and external expert judgment – all selected in less than 25% of the responses. As only three respondents selected "Other", the options provided by Q19 appear to be comprehensive. Moreover, our results show that all estimation methods are actually used in industry. Our study both corroborates and expands findings from Li et al. [33], i.e., prototyping is used in COTS selection, but also in component selection from other CSOs.

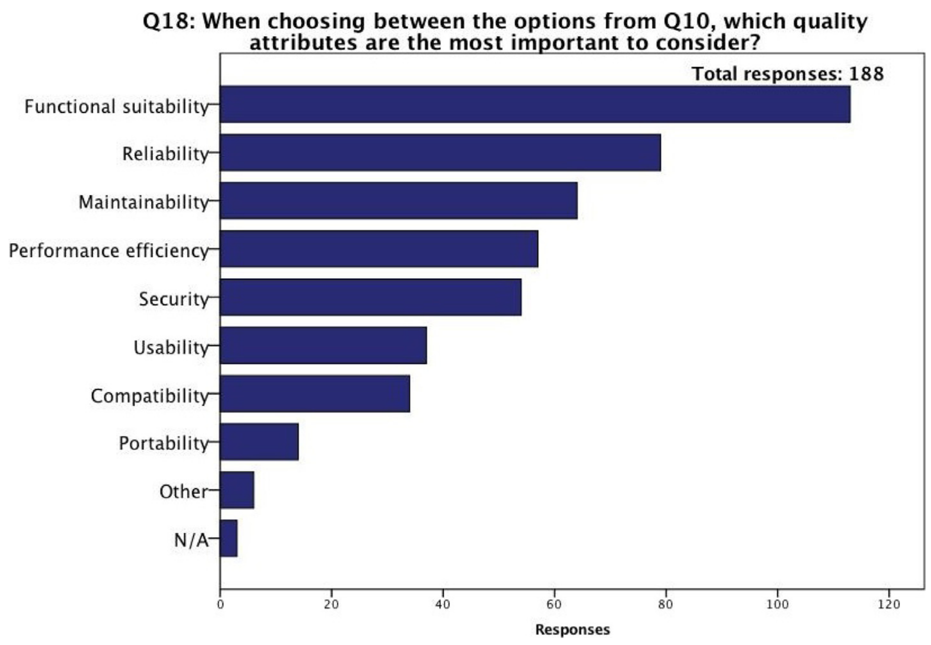

Fig. 13. The most important quality attributes in the CSO decision process. Note that up to three quality attributes could be selected.

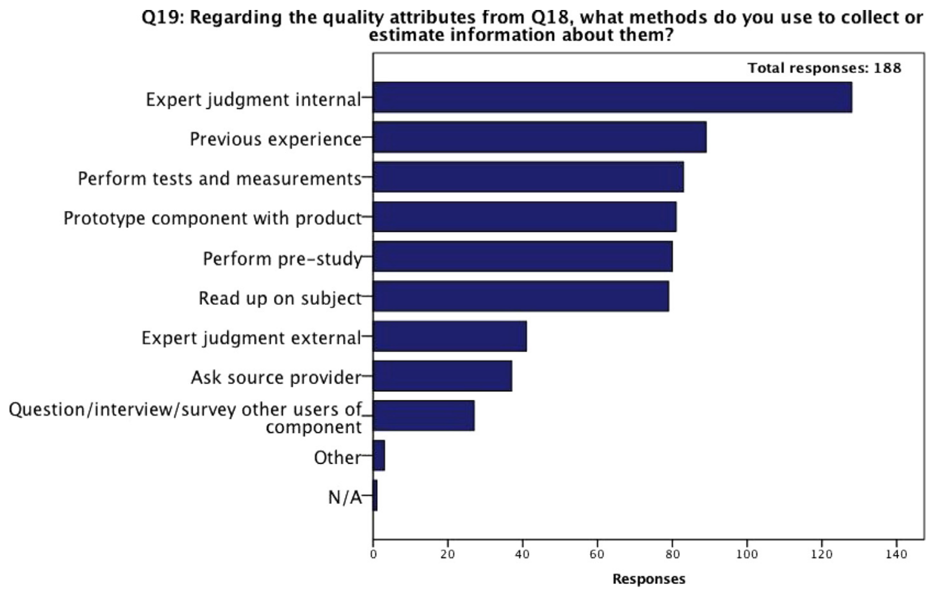

Fig. 14. How quality attributes are estimated prior to CSO decisions. Note that any number of estimation methods could be selected.

5. Statistical analysis and synthesis

This section presents additional statistical analyses of the results and synthesizes our findings to answer the RQs.

5.1. *Which CSOs are typically considered in industry? (RQ1)*

Most importantly, our results confirm the presence of the software engineering phenomenon under study: CSO selection, i.e., an extended make-or-buy decision. As reported in Section 4.1, most companies consider more than one CSO when evolving component-based systems – and if only one CSO is considered, the company typically develops all components in-house. Furthermore, our survey indicates that OSS and COTS are equally common CSOs in component-based software evolution.

Our findings confirm Badampudi et al.'s conclusion that the four CSOs in-house, outsource, OSS, and COTS are compared in practice [17]. However, while their work showed that the academic literature focuses on in-house vs. COTS and COTS vs. OSS, our work suggests that both in-house vs. COTS and in-house vs. OSS are frequently compared in industry. Also compared to Petersen et al. [18], our survey reveals a higher prevalence of OSS in industry.

On top of these findings, we identify a number of patterns among the respondents; all of the findings reported below are statistically significant, but we remind the reader that "agility" was self assessed by the respondents, as discussed in Section 6. First, agile companies are more likely to consider more than one CSO compared to plan-driven companies ($p < 0.01$, Cramér's V = 0.336). However, agile companies less frequently consider outsourcing ($p < 0.05$, Cramér's V = 0.250), concurring with the analysis provided by Turk et al. [60]: "agile development provides limited support for subcontracting". In contrast, Jalali and Wohlin [61] presented a more recent systematic literature study that shows that outsourcing indeed is used in agile development contexts. Our survey shows that outsourcing is instead a more common CSO alternative for companies with mature products and large business units – as one could expect, small organizations appear to rarely have the resources to orchestrate outsourcing initiatives.

On the other hand, agile companies are more inclined to consider OSS ($p < 0.05$, Cramér's V = 0.352). The tendency to consider OSS is not restricted to agile organizations though, as OSS is generally more popular in companies with product offerings less than a decade old. Note, however, that our survey also suggests that many mature companies consider OSS – 22 out of 60 respondents from companies with products on the market for 25+ years reported considering this CSO. We conclude that OSS has permeated software engineering contexts of various nature, attracting both young start-ups and mature companies.

5.2. What is the decision process when selecting CSOs and components? (RQ2)

Our results show that the process for CSO and component selection in industry varies considerably. Roughly the same proportion of respondents report an ad hoc decision process as a decision process of a systematic nature. On the same note, CSO decisions in industry are driven by expert judgment, corroborating findings from Petersen et al. [18]. Not only does expert judgment dominate the actual decision process, but also the assessment of the alternative options. On the other hand, a majority of the respondents also claim that the decision process is based on collected data, adhering to calls to make software engineering decisions based on empirical evidence [62]. Note, however, that our work does not organize decision making into phases as proposed by Badampudi et al. [38], i.e., decision preparation, decision initiation, and decision making. Thus, it is possible that certain phases of the decision making process are more data-oriented than others. Moreover, while the respondents' emphasis on both expert judgment and data at first might appear contradictory, we argue that it motivates efforts to complement expert judgment-based decision processes with historical data, e.g., by providing easy access to analogous cases stored in a knowledge repository [63]. Future work should take into consideration the nature of decision making in industry, and propose applicable decision support that can be integrated in contemporary ways of working – otherwise there will be no industry adoption.

We highlight a number of findings related to involvement in decision processes. First, in line with the distribution of answers for the ad hoc/systematic question, roughly the same number of respondents claim that the decisions are democratic as authoritarian. As one could expect, agile organizations with developers involved in the decision making more frequently perceive their process as democratic. On the other hand, developers' self-perception of involvement does not necessarily reflect reality. We observed that developers are more likely than other respondent roles to report that "developers are involved in the decision making" ($p < 0.05$, Cramér's V = 0.362). One explanation could be that the developers tend to overestimate their involvement in CSO decisions, i.e., that developers experience an illusion of democracy. Another possible explanation is that the developers feel involved as they are active in collecting information during the decision preparation, but other roles do not believe that activity qualifies as involvement in the actual decision making. A third explanation, partly related, could be that developers and the other roles are involved in different phases of the decision making, thus other roles are not fully aware of the developers' involvement. The last two explanations are in line with the different phases of decision making described by Badampudi et al. [38], but future work is needed to confirm whether developers' involvement tends to be restricted to early phases.

While not statistically significant, we notice a tendency that respondents from agile development organizations report longer CSO decision lead-times. As faster time-to-market is one of the expected benefits of agile development methods [64,65], the slower decision processes are somewhat counter-intuitive. On the other hand, researchers have previously highlighted that software architecture might deteriorate when agile development is introduced [66,67] – which could explain the slower architectural decision making identified in our survey.

Another finding we want to highlight in relation to RQ2 regards the agreement between the preparatory analysis and the final decision. We expected to find a discrepancy, i.e., many respondents reporting that the suggestions from analyses are not followed in the actual decision. Such a discrepancy would resonate with previous work stressing political factors in software engineering [68], e.g., in cost estimation [69] and complex systems engineering [70]. Instead, we found that most respondents agree that the final decisions follow the recommendations from analyses – again developers stand out by agreeing more than other roles ($p < 0.05$, Cramér's V = 0.341). Our conclusion is that CSO decisions are not held back by internal politics, it appears that organizations indeed make decisions in line with the results from the analyses.

5.3. What qualities are the most important input to the decision process? (RQ3)

We conclude that functional suitability is the single most important quality in CSO decisions, essentially acting as the deal-breaker. Components must serve their functional purpose. This finding was consistent across all respondents, i.e., no matter which CSOs are considered, regardless of organization size, development methods, and maturity of the company: functional suitability acts as an initial filter in the decision making.

After the initial screening phase based on functional suitability, reliability is the second most important quality in CSO decisions. Other qualities that typically are central to CSO decisions are maintainability, performance, and security. Our findings corroborate results from a literature study by Sentilles et al. [71], investigating the most important qualities in the automotive domain. The authors report that cost is the most important, followed by performance and reliability/safety. While we do not consider cost in our survey, as it is not part of the ISO 25010 quality model, our respondents report a similar ranking of qualities. Moreover, two recent surveys have ranked quality attributes in OSS. The 2017 GitHub Open Source Survey [72], collecting data from 5500 randomly sampled respondents representing 3800 GitHub repositories, reports that developers primarily value stability and security, followed by user experience, compatibility, and transparency. LibreOffice replicated the survey among their users, collecting 1330 answers, showing that the most important quality is stability, followed by compatibility, user experience, and security. We note that neither of the OSS surveys presented performance as an alternative option, that reliability was referred to as stability, and that maintainability was captured in other terms such as customizability and support. Taking all results into consideration, we conclude that reliability appears to be the universally most important quality of software regardless of domain. Furthermore, security awareness permeates software engineering in general, although it was not ranked as high by Sentilles et al. [71].

We show that component qualities are primarily estimated using expert judgment. The least common estimation methods both require human interaction: interviewing users of the component and directly asking the source provider. In general, software engineering recommendations on how to evaluate software components are technically oriented rather than human-oriented, i.e., more guidelines for experimentation and mathematical modelling such as Goševa-Popstojanova and Trivedi [73] have been published than high-level assessment frameworks such as Kontio [74]. Our survey shows that respondents working in large business units more frequently ask the source providers about the quality characteristics of a component ($p < 0.05$, Cramér's V= 0.332). This indicates that there is a higher degree of trust between large development organizations and their suppliers. As large business units typically exist in mature companies, it is likely that long-lasting business relations enable more direct and transparent communication. Furthermore, companies that recently released products or services to the market more often use prototyping to estimate component qualities ($p < 0.05$, Cramér's V= 0.278). One possible explanation is that young companies already are used to develop prototypes, thus prototyping also with candidate components is a natural approach.

Finally, we report that organizations that estimate components' (1) performance and/or (2) reliability on average have longer decision lead-times ($p < 0.05$, Cramér's V equals to 0.323 and 0.352 correspondingly). These are reasonable findings, as performance testing is known to be challenging [75] and the reliability of a component inevitably grows with increased testing times [76] – a high reliability target, necessitates prolonged time in the testing phase.

6. Limitations and threats to validity

The population under study, i.e., practitioners involved in architectural decision making in component-based software evolution, is large and highly heterogeneous. Our survey was not designed to make strong quantitative conclusions about the general population of practitioners involved in CSO decisions, but rather to identify larger trends. We relied on a non-probabilistic method referred to as accidental sampling [77], i.e., we recruited respondents based on convenience – in line with most software engineering surveys [78]. While we took normal precautions to collect answers from valid respondents, and filtered all answers as described in Section 3.5, we can never be certain that the respondents actually are knowledgeable about CSO decisions in industry.

We continue by discussing three types of threats to survey validity presented by Kitchenham and Pfleger [40]. For further details, we refer the reader to the technical report [51]. *Content validity* concerns how much a measure represents every single element of a construct. In our case, we needed to ensure that our questionnaire covered all aspects of CSO decisions in industry. However, there is a trade-off between the length of the questionnaire and the coverage. We used an instrument evaluation with pilot runs, as described in Section 3.1, to find a feasible balance. Some aspects related to CSO decisions were intentionally left out, as well as combinations of the four CSOs. Among the excluded aspects, we list three aspects whose influence on CSO decisions would be particularly interesting to study in future work: offshoring, product-line engineering, and software ecosystems.

Construct validity refers to how an operational definition of a variable actually reflects the true theoretical meaning of a concept. The major threat to our study is whether our inquiry about previously experienced CSO decisions truly reflects a phenomenon in industry. Our initial construct captured CSO decisions and component selection as two separate activities, but our construct evolved during the study. Our data analysis suggested that in many cases the two decisions are intertwined; the most appropriate component is selected regardless of its CSO. To mitigate this threat, we let RQ1 address CSO decisions and we opened up RQ2 and RQ3 to instead discuss component selection in more general terms. Another threat to construct validity is our simplified description of the OSS option, i.e., integrating an existing OSS component. It could be argued that this reflects a naïve and immature view on open source development, as it is increasingly common for organizations to develop new components in-house, but under an OSS license from the start [79] – such development would qualify as both in-house and OSS in our questionnaire. However, we believe that we captured all such examples through the mix of closed and open questions.

Finally, *criterion validity* deals with the ability of a measurement instrument to distinguish respondents belonging to different groups. Our questionnaire collected self-reported assessments and opinions, an approach that might introduce certain biases. In our questionnaire, we believe that the biggest threat is related to the self-assessment of agility, a phenomenon that is known to be hard to evaluate [80]. We use the statement in Q8 ("My organization is more agile than plan-driven") to distinguish respondents, but we did not triangulate this self-reported agility assessment beyond analyzing the respondents' open question replies.

7. Conclusion and future work

A recurring strategic consideration for organizations evolving component-based systems is the make-or-buy decision, i.e., whether to develop the components internally or to acquire them from external sources. In software engineering, make-or-buy decisions are more complex than in traditional manufacturing, as both the make and buy options are represented by several sourcing options and have long-term implications in terms of maintenance and evolvability. In this work, we focus on four component sourcing options (CSO): (1) in-house development, (2) outsourced development, (3) buying commercial-off-the-shelf (COTS) software, and (4) integrating existing open source software (OSS).

We present an industrial survey on practitioners' decision making in relation to choosing between CSOs for adding components in evolving software-intensive systems. We obtained 188 responses from various roles involved in development of software-intensive products and services across different domains. As there are few previous studies on CSO selection, we contribute novel input to an understudied software engineering challenge, manifested in the answers to the research questions below.

RQ1 Our survey confirms that CSO selection constitutes a recurring decision point in software engineering. All four CSOs are frequently considered in industry; in-house is the most common, in turn followed by OSS, COTS, and outsourcing. Furthermore, most companies consider more than one CSO when evolving component-based systems.

RQ2 We show that the processes for CSO and component selection in industry vary considerably. Ad hoc decision processes are about as common as systematic counterparts. Irrespective of systematism, component decisions in industry are driven by expert judgment. On the other hand, most decision processes are complemented by collected data. Moreover, authoritarian and democratic decision processes appear to be equally common in industry, but many different roles are involved in both cases.

RQ3 We conclude that functional suitability is the single most important quality in component decisions. Once a component's functional suitability has been determined, other qualities are estimated. Reliability is the second most important quality, followed by maintainability, performance, and security. Due to consistency across domains and contexts, our results suggest that functional suitability and reliability are universal qualities that make components more likely to be selected.

Our research has several implications for research and practice. First, we show that any solution-oriented work on decision support has to account for the dominance of expert judgment in industry – or end up as yet another academic construct collecting imaginary dust in a digital library. Note that we do not argue that researchers must incorporate expert judgment in their solutions, we rather ask researchers to acknowledge how state-of-practice CSO decisions are made and adapt any deployment plans accordingly. Academic solution proposals, e.g., objective decision support tools, might appear exotic to industry and thus need to be gradually introduced. Second, we observe a wide variety of decision processes – not at all surprising given the heterogeneous contexts experienced by the survey respondents. Future research should further explore contextual differences, as a single decision process will not rule them all. Instead, to really make an impact in industry, researchers should focus efforts on solutions tailored for specific contexts rather than global solutions. Third, our survey shows that decisions are based on data, but further research is needed to understand what types of data are used, how they are used, and how the data are translated into the actual decisions.

From the perspective of an industrial practitioner, our survey might encourage self-reflection. Hopefully, our study can motivate organizations to take a step back and consider their own decision processes. We believe there is a value in simply increasing awareness of how decisions are made, and our survey can enable a benchmarking against the overall community. Along the same lines, we recommend organizations to explicitly cover CSO and component decisions in retrospective meetings and post-mortem analyses. Only by bringing the decision processes to the surface, assessment and improvement activities can commence. Finally, also related to process improvement, development organizations might benefit from augmenting internal metrics initiatives to encompass also CSO and component decisions, incl. lead-times and decision rationales. While it by no means is trivial, software reuse will remain an engine for software engineering projects – thus CSO and component decisions will inevitably be key concepts.

In the Orion project, we will continue our research on efficient and effective decision making in component-based software engineering. First, we aim to help organizations improve their decisions-making processes by collecting data or evidence relevant to their architectural decisions with the GRADE taxonomy [37] and canvas [81], possibly through direct interaction with partner companies during agile retrospective meetings in the field, or through focus groups in more controlled settings. Second, we will continue developing COACH [82], with a particular focus on creating a lightweight decision support tool that can be easily integrated into a decision process dominated by expert judgment. Moreover, since we now know that decisions are also complemented by data (cf. Fig. 10c), we will continue our efforts to make previous decisions useful [63], either by case-based reasoning or more advanced pattern matching techniques. Finally, we have initiated detailed investigations of a handful of development contexts. At the moment, we study component selection in the automotive domain and sourcing strategies in public sector development at government agencies in Sweden [83].

Acknowledgement

The work is partially supported by a research grant for the ORION project (reference number 20140218) from The Stiftelsen för Kunskaps- och Kompetensutveckling in Sweden. The authors have no competing interests to declare.

Conflict of interest

None.

Appendix

Tables 1–3 show the questions of the web-based questionnaire. The questions were either closed, i.e., multiple choice, Likert scale (or single Likert item), distribute $100 or open-ended, i.e., with free-text answers.

Table 1
The demographics section of the questionnaire, Q1–Q8.

ID	Question	Type
Q1	Which of the following best describes your current role?	Multiple choice, select one
Q2	How many years of working experience do you have?	Multiple choice, select one
Q3	What is the highest level of formal education you have received?	Multiple choice, select one
Q4	In which domain(s) do you primarily work?	Multiple choice, select one or more
Q5	Please provide a brief description of the product or service that is the principal focus of your work.	Free-text
Q6	For how many years has your company had this product or service category on the market?	Multiple choice, select one
Q7	How many co-workers do you have in the business unit in which you work?	Multiple choice, select one
Q8	“My development organization is more agile than plan-driven.”	Likert item

Table 2
The main part of the questionnaire, Q9–Q24.

Q9	What options do you typically compare when choosing to add/replace a new component to your product? [Inhouse, Outsource, COTS, OSS]	Multiple choice, select one or more
Q10–Q13	What is the main reason for you not considering the option <from Q9>?	Free-text
Q14	When choosing between the options <from Q9>, which roles/perspectives are involved in the decision making?	Multiple choice, select one or more
Q15	When choosing between the options <from Q9>, what information is the most important input to the decision process?	Multiple choice, select one to five
Q16	Given the options you selected as the most important in the previous question, please indicate their relative importance by distributing 100 points across the alternatives.	Distribute $100
	a) “... we follow a systematic decision process.”	
	b) “... our decision process is mainly based on expert judgment.”	
Q17	c) “... our decision process is based on data we collect.”	Likert scale
	d) “... the final decision is democratic rather than authoritarian.”	
	e) “... the decision process is transparent.”	
Q18	When choosing between the options <from Q9>, which quality attributes are the most important to consider?	Multiple choice, select one to three
Q19	Regarding the quality attributes <from Q18>, what methods do you use to collect or estimate information about them?	Multiple choice, select one or more
	How long is the typical lead-time needed to reach a decision in selecting a component (i.e., choosing between <from Q9>) for your product?	
Q20	a) Minimum lead-time	Multiple choice, select one
	b) Typical lead-time	
	c) Maximum lead-time	
Q21	When choosing between the options <from Q9>, what are the main factors that make a decision challenging?	Free-text
Q22	“The final decision you make in your company on which component to select agrees with what your analysis showed to be the best option.”	Likert item
Q23	(If strongly agree/disagree to Q22) Please explain the reason for your answer to the previous question.	Free-text
Q24	If you would like to receive our final report at the end of the study, please provide your email address.	Free-text

Table 3
The concluding part of the questionnaire, Q25–Q26.

Q25	If you entered an email address in the previous question, would you be willing to answer follow-up questions related to the topic of this survey?	Yes/No
Q26	Finally, is there anything else you would like to add before submitting the survey?	Free-text

Note that Q15 and Q16 are presented for the sake of completeness, and the findings are reported in a separate publication [43].

References

[1] T. Vale, I. Crnkovic, E.S. de Almeida, P.A.d.M. Silveira Neto, Y.C. Cavalcanti, S.R.d.L. Meira, Twenty-eight years of component-based software engineering, J. Syst. Softw. 111 (2016) 128–148, doi:10.1016/j.jss.2015.09.019.
[2] P. Ulkuniemi, V. Seppanen, COTS component acquisition in an emerging market, IEEE Softw. 21 (6) (2004) 76–82, doi:10.1109/MS.2004.38.
[3] D. Šmite, C. Wohlin, T. Gorschek, R. Feldt, Empirical evidence in global software engineering: a systematic review, Empir. Softw. Eng. 15 (1) (2010) 91–118, doi:10.1007/s10664-009-9123-y.
[4] K. Manikas, K.M. Hansen, Software ecosystems - a systematic literature review, J. Syst. Softw. 86 (5) (2013) 1294–1306, doi:10.1016/j.jss.2012.12.026.
[5] P. Kraljic, Purchasing must become supply management, Harv. Bus. Rev. 61 (5) (1983) 109–117.

[6] S. Kurokawa, Make-or-buy decisions in R&D: small technology based firms in the United States and Japan, IEEE Trans. Eng. Manage. 44 (2) (1997) 124–134, doi:10.1109/17.584921.
[7] C. Wohlin, K. Wnuk, D. Šmite, U. Franke, D. Badampudi, A. Cicchetti, Supporting strategic decision-making for selection of software assets, in: Proc. of the International Conference of Software Business, in: Lecture Notes in Business Information Processing, Springer, Cham, 2016, pp. 1–15, doi:10.1007/978-3-319-40515-5_1.
[8] K. Wallnau, S.A. Hissam, R.C. Seacord, Building Systems from Commerical Components, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
[9] C. Ruffin, C. Ebert, Using open source software in product development: a primer, IEEE Softw. 21 (1) (2004) 82–86, doi:10.1109/MS.2004.1259227.
[10] B. Adams, R. Kavanagh, A. Hassan, D.M. German, An empirical study of integration activities in distributions of open source software, Empir. Softw. Eng. 21 (3) (2016) 960–1001, doi:10.1007/s10664-015-9371-y.
[11] A. Holzer, J. Ondrus, Trends in mobile application development, in: Proc. of the Mobile Wireless Middleware, Operating Systems, and Applications - Workshops, Springer, Berlin, 2009, pp. 55–64, doi:10.1007/978-3-642-03569-2_6.
[12] S.M. Sulaman, A. Orucevic-Alagic, M. Borg, K. Wnuk, M. Höst, J.L. de la Vara, Development of safety-critical software systems using open source software - a systematic map, in: Proc. of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications, 2014, pp. 17–24, doi:10.1109/SEAA.2014.25.
[13] K. Wnuk, Involving relevant stakeholders into the decision process about software components, in: Proc. of the IEEE International Conference on Software Architecture Workshops (ICSAW), 2017, pp. 129–132, doi:10.1109/ICSAW.2017.68.
[14] N.E. Fenton, M. Neil, Decision support software for probabilistic risk assessment using bayesian networks, IEEE Softw. 31 (2) (2014) 21–26. https://doi.org/10.1109/MS.2014.32.
[15] S.A. Busari, E. Letier, RADAR: a lightweight tool for requirements and architecture decision analysis, in: Proc. of the 39th International Conference on Software Engineering Companion, IEEE Press, Piscataway, NJ, USA, 2017, pp. 552–562, doi:10.1109/ICSE.2017.57.
[16] Z. Sahaf, V. Garousi, D. Pfahl, R. Irving, Y. Amannejad, When to automate software testing? Decision support based on system dynamics: an industrial case study, in: Proc. of the 2014 International Conference on Software and System Process, 2014, pp. 149–158, doi:10.1145/2600821.2600832.
[17] D. Badampudi, C. Wohlin, K. Petersen, Software component decision-making: in-house, OSS, COTS or outsourcing - a systematic literature review, J. Syst. Softw. 121 (2016) 105–124, doi:10.1016/j.jss.2016.07.027.
[18] K. Petersen, D. Badampudi, S.M.A. Shah, K. Wnuk, T. Gorschek, E. Papatheocharous, J. Axelsson, S. Sentilles, I. Crnkovic, A. Cicchetti, Choosing component origins for software intensive systems: in-house, COTS, OSS or outsourcing? - a case survey, IEEE Trans. Softw. Eng. 44 (3) (2018) 237–261, doi:10.1109/TSE.2017.2677909.
[19] J. Miller, Triangulation as a basis for knowledge discovery in software engineering, Empir. Softw. Eng. 13 (2) (2008) 223–228, doi:10.1007/s10664-008-9063-y.
[20] L. Brownsword, T. Oberndorf, C.A. Sledge, Developing new processes for COTS-based systems, IEEE Softw. 17 (4) (2000) 48–55, doi:10.1109/52.854068.
[21] J. Li, F.O. Bjørnson, R. Conradi, V.B. Kampenes, An empirical study of variations in COTS-based software development processes in the Norwegian IT industry, Empir. Softw. Eng. 11 (3) (2006) 433–461, doi:10.1007/s10664-006-9005-5.
[22] V. Cortellessa, F. Marinelli, P. Potena, An optimization framework for æbuild-or-buyg decisions in software architecture, Comput. Oper. Res. 35 (10) (2008) 3090–3106, doi:10.1016/j.cor.2007.01.011.
[23] J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, M. Torchiano, M. Morisio, An empirical study on decision making in off-the-shelf component-based development, in: Proc of the 28th International Conference on Software Engineering, ACM, New York, NY, USA, 2006, pp. 897–900, doi:10.1145/1134285.1134446.
[24] F. Daneshgar, G.C. Low, L. Worasinchai, An investigation of build vs. buy decision for software acquisition by small to medium enterprises, Inf. Softw. Technol. 55 (10) (2013) 1741–1750, doi:10.1016/j.infsof.2013.03.009.
[25] D. Tofan, M. Galster, P. Avgeriou, Difficulty of architectural decisions - a survey with professional architects, in: Proc. of the 5th European Conference on Software Architecture, in: Lecture Notes in Computer Science, Springer, Berlin, 2013, pp. 192–199, doi:10.1007/978-3-642-39031-9_17.
[26] H. van Vliet, A. Tang, Decision making in software architecture, J. Syst. Softw. 117 (2016) 638–644, doi:10.1016/j.jss.2016.01.017.
[27] J. Axelsson, Evolutionary architecting of embedded automotive product lines: an industrial case study, in: Proc. of the Joint Working IEEE/IFIP Conference on Software Architecture & 3rd European Conference on Software Architecture, 2009, pp. 101–110, doi:10.1109/WICSA.2009.5290796.
[28] C. Ayala, O. Hauge, R. Conradi, X. Franch, J. Li, Selection of third party software in Off-The-Shelf-based software development - a interview study with industrial practitioners, J. Syst. Softw. 84 (4) (2011) 620–637, doi:10.1016/j.jss.2010.10.019.
[29] M.M. Gerea, Selection of Open Source Components - A Qualitative Survey in Norwegian IT Industry, Technical Report, Norwegian University of Science and Technology, 2007.
[30] F. Kokkoras, K. Ntonas, A. Kritikos, G. Kakarontzas, I. Stamelos, Federated search for open source software reuse, in: Proc. of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications, 2012, pp. 200–203, doi:10.1109/SEAA.2012.55.
[31] A. Mavridis, I. Stamelos, Real options as tool enhancing rationale of OSS components selection, in: Proc. of the 3rd IEEE International Conference on Digital Ecosystems and Technologies, 2009, pp. 613–618, doi:10.1109/DEST.2009.5276768.
[32] O. Hauge, T. Osterlie, C.-F. Sorensen, M. Gerea, An empirical study on selection of open source software - preliminary results, in: Proc. of the 2nd ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, IEEE Computer Society, Washington, DC, USA, 2009, pp. 42–47, doi:10.1109/FLOSS.2009.5071359.
[33] J. Li, R. Conradi, C. Bunse, M. Torchiano, O.P.N. Slyngstad, M. Morisio, Development with off-the-shelf components: 10 facts, IEEE Softw. 26 (2) (2009) 80–87, doi:10.1109/MS.2009.33.
[34] M. Torchiano, M. Morisio, Overlooked aspects of COTS-based development, IEEE Softw. 21 (2) (2004) 88–93, doi:10.1109/MS.2004.1270770.
[35] A.S. Jadhav, R.M. Sonar, Evaluating and selecting software packages: a review, Inf. Softw. Technol. 51 (3) (2009) 555–563.
[36] S. Sentilles, F. Ciccozzi, E. Papatheocharous, Promopedia: a web-content management-based encyclopedia of software property models, in: Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, in: ICSE '18, ACM, New York, NY, USA, 2018, pp. 45–48, doi:10.1145/3183440.3183482.
[37] E. Papatheocharous, K. Wnuk, K. Petersen, S. Sentilles, A. Cicchetti, T. Gorschek, S.M.A. Shah, The GRADE taxonomy for supporting decision making asset selection in software-intensive system development, Inf. Softw. Technol. (2018), doi:10.1016/j.infsof.2018.02.007.
[38] D. Badampudi, K. Wnuk, C. Wohlin, U. Franke, D. Šmite, A. Cicchetti, A decision-making process-line for selection of software asset origins and components, J. Syst. Softw. 135 (2018) 88–104, doi:10.1016/j.jss.2017.09.033.
[39] International Organization for Standardization, Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SquaRE) - System and Software Quality Models, 2011.
[40] B. Kitchenham, S. Pfleeger, Personal opinion surveys, in: Guide to Advanced Empirical Software Engineering, Springer, London, 2008, pp. 63–92. 00168 doi:10.1007/978-1-84800-044-5_3.
[41] J. Singer, S. Sim, T. Lethbridge, Software engineering data collection for field studies, in: F. Shull, J. Singer, D. Sjoberg (Eds.), Guide to Advanced Empirical Software Engineering, Springer, 2008, pp. 9–34.
[42] S. Easterbrook, J. Singer, M. Storey, D. Damian, Selecting empirical methods for software engineering research, in: F. Shull, J. Singer, D. Sjoberg (Eds.), Guide to Advanced Empirical Software Engineering, Springer, London, 2008, pp. 285–311.
[43] P. Chatzipetrou, E. Alégroth, E. Papatheocharous, M. Borg, T. Gorschek, K. Wnuk, Component selection in software engineering-which attributes are the most important in the decision process? in: Proc. of the 44th Euromicro Conference on Software Engineering and Advanced Applications, IEEE, 2018, pp. 198–205.
[44] L.M. Rea, R.A. Parker, Designing and Conducting Survey Research: A Comprehensive Guide, fourth ed., Jossey-Bass, San Francisco, CA, United states, 2014.
[45] R.M. de Mello, G.H. Travassos, Surveys in software engineering: identifying representative samples, in: Proc. of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2016, pp. 55:1–55:6, doi:10.1145/2961111.2962632.
[46] M. Galster, D. Tofan, Exploring web advertising to attract industry professionals for software engineering surveys, in: Proc. of the 2nd International Workshop on Conducting Empirical Studies in Industry, 2014, pp. 5–8, doi:10.1145/2593690.2593695.
[47] A. Field, Discovering Statistics using SPSS, third ed., SAGE Publications, Thousand Oaks, CA, US, 2009.
[48] A. Agresti, Categorical Data Analysis, third ed., Wiley, 2013.
[49] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, second ed., Routledge, London, UK, 1988.
[50] A. Strauss, J. Corbin, Grounded theory methodology - an overview, in: N.K. Denzin, Y.S. Lincoln (Eds.), Handbook of Qualitative Research, SAGE Publications, Thousand Oaks, CA, US, 1994, pp. 273–285.
[51] M. Borg, P. Chatzipetrou, K. Wnuk, E. Alegroth, T. Gorschek, E. Papatheocharous, S. Muhammad Ali Shah, J. Axelsson, Selecting Software Component Sourcing Options - Detailed Survey Description and Analysis, Technical Report, RISE Report 2018:71, ISBN 978-91-88907-15-8 RISE Research Institutes of Sweden AB, 2018.
[52] I.J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, A.E. Hassan, A large-scale empirical study on software reuse in mobile apps, IEEE Softw. 31 (2) (2014) 78–86, doi:10.1109/MS.2013.142.
[53] D.A. Botwe, J.G. Davis, A comparative study of web development technologies using open source and proprietary software, Int. J. Comput. Sci. Mob. Comput. 4 (2) (2015) 154–165.
[54] N.B. Moe, D. Šmite, G.K. Hanssen, H. Barney, From offshore outsourcing to insourcing and partnerships: four failed outsourcing attempts, Empir. Softw. Eng. 19 (5) (2014) 1225–1258.
[55] S. Goode, Something for nothing: management rejection of open source software in Australias top firms, Inf. Manag. 42 (5) (2005) 669–681.
[56] C. Ayala, A. Nguyen-Duc, X. Franch, M. Höst, R. Conradi, D. Cruzes, M.A. Babar, System requirements-oss components: matching and mismatch resolution practices – an empirical study, Empir. Softw. Eng. (2018) 1–56.
[57] A. Aurum, C. Wohlin, The fundamental nature of requirements engineering activities as a decision-making process, Inf. Softw. Technol. 45 (14) (2003) 945–954, doi:10.1016/S0950-5849(03)00096-X.
[58] M. Jørgensen, Forecasting of software development work effort: evidence on expert judgement and formal models, Int. J. Forecas. 23 (3) (2007) 449–462, doi:10.1016/j.ijforecast.2007.05.008.
[59] K. Wnuk, T. Gorschek, D. Callele, E.A. Karlsson, E. Ahlin, B. Regnell, Supporting scope tracking and visualization for very large-scale requirements engineering-utilizing FSC+, decision patterns, and atomic decision visualizations, IEEE Trans. Softw. Eng. 42 (1) (2016) 47–74, doi:10.1109/TSE.2015.2445347.
[60] D. Turk, R. France, B. Rumpe, Assumptions underlying agile software development processes, J. Database Manag. 16 (4) (2005) 62–87.

[61] S. Jalali, C. Wohlin, Global software engineering and agile practices: a systematic review, J. Softw. 24 (6) (2011) 643–659, doi:10.1002/smr.561.
[62] T. Dybå, B.A. Kitchenham, M. Jørgensen, Evidence-based software engineering for practitioners, IEEE Softw. 22 (1) (2005) 58–65, doi:10.1109/MS.2005.6.
[63] A. Cicchetti, M. Borg, S. Sentilles, K. Wnuk, J. Carlsson, E. Papatheocharous, Towards software assets origin selection supported by a knowledge repository, in: Proc. of the 1st International Workshop on Decision Making in Software Architecture, 2016.
[64] N. Dzamashvili Fogelström, T. Gorschek, M. Svahnberg, P. Olsson, The impact of agile principles on market? Driven software product development, J. Softw. Mainten. Evol. 22 (1) (2010) 53–80, doi:10.1002/spip.420.
[65] H. Holmström Olsson, H. Alahyari, J. Bosch, Climbing the "Stairway to Heaven" - a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software, in: Proc. of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications, 2012, pp. 392–399, doi:10.1109/SEAA.2012.54.
[66] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (9) (2008) 833–859, doi:10.1016/j.infsof.2008.01.006.
[67] P. Abrahamsson, M. Babar, P. Kruchten, Agility and architecture: can they coexist? IEEE Softw. 27 (2) (2010) 16–22, doi:10.1109/MS.2010.36.
[68] M. Lavallee, P.N. Robillard, Why good developers write bad code: an observational case study of the impacts of organizational factors on software quality, in: Proc. of the 37th IEEE International Conference on Software Engineering, 2015, pp. 677–687, doi:10.1109/ICSE.2015.83.
[69] A. Magazinius, S. Börjesson, R. Feldt, Investigating intentional distortions in software cost estimation - an exploratory study, J. Syst. Softw. 85 (8) (2012) 1770–1781, doi:10.1016/j.jss.2012.03.026.
[70] P.F. Katina, C.B. Keating, R.M. Jaradat, System requirements engineering in complex situations, Require. Eng. 19 (1) (2014) 45–62, doi:10.1007/s00766-012-0157-0.
[71] S. Sentilles, E. Papatheocharous, F. Ciccozzi, K. Petersen, A property model ontology, in: Proc. of the 42th EUROMICRO Conference on Software Engineering and Advanced Applications, 2016, pp. 165–172.
[72] R.S. Geiger, Summary analysis of the 2017 GitHub open source survey, arXiv:1706.02777 [cs] (2017). https://github.com/github/open-source-survey.
[73] K. Goseva-Popstojanova, K.S. Trivedi, Architecture-based approach to reliability assessment of software systems, Perform. Eval. 45 (2) (2001) 179–204, doi:10.1016/S0166-5316(01)00034-7.
[74] J. Kontio, A case study in applying a systematic method for COTS selection, in: Proc. of the 18th International Conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, 1996, pp. 201–209.
[75] M. Woodside, G. Franks, D.C. Petriu, The future of software performance engineering, in: Proc. of the Future of Software Engineering, IEEE Computer Society, 2007, pp. 171–187, doi:10.1109/FOSE.2007.32.
[76] M.R. Lyu, Software reliability engineering: a roadmap, in: Proc. of the 1st Future of Software Engineering, IEEE Computer Society, Washington, DC, USA, 2007, pp. 153–170, doi:10.1109/FOSE.2007.24.
[77] J. Linåker, S.M. Sulaman, R.M. de Mello, M. Höst, Guidelines for Conducting Surveys in Software Engineering, Technical Report, Lund University, 2015.
[78] R.M. de Mello, P.C. da Silva, G.H. Travassos, Investigating probabilistic sampling approaches for large-scale surveys in software engineering, J. Softw. Eng. Res.Dev. 3 (1) (2015) 1–26. 00017 doi: 10.1186/s40411-015-0023-0.
[79] O. Alexy, J. Henkel, M.W. Wallin, From closed to open: job role changes, individual predispositions, and the adoption of commercial open source software development, Res. Policy 42 (8) (2013) 1325–1340, doi:10.1016/j.respol.2013.04.007.
[80] S. Jalali, C. Wohlin, L. Angelis, Investigating the applicability of agility assessment surveys: a case study, J. Syst. Softw. 98 (2014) 172–190, doi:10.1016/j.jss.2014.08.067.
[81] E. Papatheocharous, K. Petersen, J. Axelsson, C. Wohlin, J. Carlson, F. Ciccozzi, S. Sentilles, A. Cicchetti, The GRADE Decision Canvas for Classification and Reflection on Architecture Decisions, 2017, pp. 187–194.
[82] J. Axelsson, U. Franke, J. Carlson, S. Sentilles, A. Cicchetti, Towards the architecture of a decision support ecosystem for system component selection, in: Proc. of the 11th Annual IEEE International Systems Conference, 2017, pp. 1–7, doi:10.1109/SYSCON.2017.7934757.
[83] M. Borg, T. Olsson, U. Franke, S. Assar, Digitalization of swedish government agencies: aperspective through the lens of a software development census, in: Proc. of the 40th International Conference on Software Engineering: Software Engineering in Society, in: ICSE-SEIS '18, ACM, New York, NY, USA, 2018, pp. 37–46, doi:10.1145/3183428.3183434.

Markus Borg is a senior researcher with the Software and Systems Engineering Laboratory, RISE Research Institutes of Sweden AB and an adjunct senior lecturer at Lund University, Sweden. Dr. Borg received a PhD degree in software engineering in 2015 from Lund University, Sweden. His research interests include software testing, safety-critical systems, and machine learning.

Panagiota Chatzipetrou is an assistant professor at the department of Informatics at Örebro University School of Business in Örebro Sweden where she belongs to the *Centre for empirical research on information systems (CERIS)*. She is also part of the *Software Research Engineering Lab (SERL)* at Blekinge Institute of Technology in Karlskrona, Sweden. She received her BSc, MSc and PhD in Informatics from the Department of Informatics, Aristotle University of Thessaloniki (AUTh), Greece. As a researcher, she mainly focuses on empirical studies under the different perspectives of software development. Her research interests include applications of statistical methods to quality problems in software engineering and especially to requirements engineering and the exploitation of human factor and the different views that ultimately determine the quality of a software product and the product development. Also, she has been working with decision support systems for the development of software-intensive systems, large-scale agile (and global) software development, and behavioural software engineering.

Krzysztof Wnuk is an associate professor at 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.

Emil Alégroth finished his PhD in 2015 at Chalmers University of Technology, Sweden, and has since then worked as a researcher at Blekinge Institute of Technology, Sweden. His research interests include software decision making, human-factors in software engineering, and automated software verification and validation. Emil also has several years of industrial experience as the CEO of a small service and product development company.

Tony Gorschek is a professor of software engineering at Blekinge Institute of Technology, Sweden – where he works as a research scientist in close collaboration with industrial partners. Dr. Gorschek has over fifteen years industrial experience as a CTO, senior executive consultant, and engineer. In addition, he is a serial entrepreneur – with five startups in fields ranging from logistics to Internet-based services and database register optimization. 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^{TM} and lean product development and evolution.

Efi Papatheocharous is a senior researcher at RISE Research Institutes of Sweden AB. She received a BSc degree in computer science from the Department of Computer Science, University of Cyprus, in 2004, a MSc degree in advanced computer science with ICT management from the University of Manchester, in 2005, and a PhD degree in computer science from the University of Cyprus, in 2012. Her primary research interests include software and systems architecture decision making and agile methods in software engineering.

Syed Muhammad Ali Shah is a software quality specialist at iZettle AB. He received his PhD in software engineering from Politecnico di Torino, Italy. He was a senior researcher in the Software and Systems Engineering (SSE) Laboratory at the 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.

Jakob Axelsson received an MSc in computer science in 1993, and a PhD in computer systems in 1997, both from Linköping University, Sweden. He was with Volvo and Volvo Cars in Göteborg from 1997 to 2010. In 2004, he became a part-time professor at Mälardalen University, Västerås. Since 2010, he has also been with the Swedish Institute of Computer Science (SICS) in Kista, where he founded the Software and Systems Engineering Laboratory. He is the author of around 100 research publications. His current research interests are focused on system-of-systems engineering.