Software Quality Journal (2020) 28:567–593
https://doi.org/10.1007/s11219-019-09465-2

Component attributes and their importance in decisions and component selection

Panagiota Chatzipetrou[1] · Efi Papatheocharous[2] · Krzysztof Wnuk[3] · Markus Borg[2] · Emil Alégroth[3] · Tony Gorschek[3]

Published online: 7 September 2019

Abstract
Component-based software engineering is a common approach in the development and evolution of contemporary software systems. Different component sourcing options are available, such as: (1) *Software developed internally (in-house)*, (2) *Software developed outsourced*, (3) *Commercial off-the-shelf software*, and (4) *Open-Source Software*. However, there is little available research on what attributes of a component are the most important ones when selecting new components. The objective of this study is to investigate what matters the most to industry practitioners when they decide to select a component. We conducted a cross-domain anonymous survey with industry practitioners involved in component selection. First, the practitioners selected the most important attributes from a list. Next, they prioritized their selection using the Hundred-Dollar ($100) test. We analyzed the results using compositional data analysis. The results of this exploratory analysis showed that *cost* was clearly considered to be the most important attribute for component selection. Other important attributes for the practitioners were: *support of the component*, *longevity prediction*, and *level of off-the-shelf fit to product*. Moreover, several practitioners still consider in-house software development to be the sole option when adding or replacing a component. On the other hand, there is a trend to complement it with other component sourcing options and, apart from cost, different attributes factor into their decision. Furthermore, in our analysis, nonparametric tests and biplots were used to further investigate the practitioners' inherent characteristics. It seems that smaller and larger organizations have different views on what attributes are the most important, and the most surprising finding is their contrasting views on the cost attribute: larger organizations with mature products are considerably more cost aware.

Keywords Component-based software engineering · Component sourcing options · Decision making · Compositional data analysis · Cumulative voting

✉ Panagiota Chatzipetrou
panagiota.chatzipetrou@oru.se

Extended author information available on the last page of the article

1 Introduction

Component-based software engineering (CBSE) is a common approach in the development and evolution of contemporary software systems. However, in CBSE, developing a new component internally (in-house) is not necessarily the best option (Wohlin et al., 2016). Thus, practitioners are very often asked to choose between different component sourcing options (CSO). But what are the factors that affect a practitioners' decision to choose one CSO over another? In other words, how do the practitioners prioritize the attributes of a component when they have to decide on "buying" or "making" a new component?

Prioritization is a procedure of critical importance in decision making. In software engineering, it is encountered in cases where multiple attributes have to be considered in order to make a decision. However, human subjectivity accounts for variation when different people try to independently prioritize a certain number of attributes. These factors have led to the adoption of voting schemes where stakeholders express their relative preferences for certain attributes in a systematic and controlled manner. We used cumulative voting (CV) or the 100-point method or hundred-dollar ($100) test, described by Leffingwell and Widrig (Leffingwell & Widrig, 2003), to gather practitioners' preferences. The methodology utilized for the analysis of the results was compositional data analysis (CoDA).

There is little available research on which attributes of a component are of primary importance when multiple attributes are considered to decide which component sourcing option is more appropriate in different cases. Thus, making a CSO decision is crucial and the reasons behind it deserve further investigation. Understanding the source of variation between decision makers among different CSOs in CBSE may optimize the decision process and consolidate opinions with respect to prioritization. In the present work, we focused on the attributes that practitioners typically compare when they are choosing to add or replace a new component for their products. The products concern software-intensive systems and thus entail component complexity. Therefore, an industrial cross-domain anonymous survey regarding the practitioners' decision making in relation to choosing between CSOs was conducted (Borg et al., 2019). The questionnaire was web-based and consisted of both open-ended and closed-ended questions. The practitioners were asked to choose between different CSOs; however, they were also able to choose more than one CSO.

The present work is an extension of our previous work (Chatzipetrou et al., 2018). In this study, we aimed to further investigate the different views of practitioners towards the prioritization of different attributes. The results of the hundred-dollar ($100) test are coded as variables, and they are statistically analyzed in order to find differences or agreements in views and correlations with other inherent characteristics of the practitioners, i.e., the role, amount of working experience, level of education, maturity of product they work with, and size of their organization. This information was collected via the same survey (Borg et al., 2019). The selection of the inherent characteristics was based on the available data. For instance, the education type was included in the analysis but its domain was not included since the answers we got from the survey were too many and the results were sparse. Thus, we included in the analysis the inherent characteristics where are at the same time meaningful and we have data availability from our sample. A further investigation using the inherent characteristics of the practitioners was undertaken in the present work. The focus of this work is twofold. First, to investigate if there are any patterns behind the views on CSO decisions for a particular type of practitioner, product, or organization, and secondly to understand if the different views of particular types of practitioners, products developed, or organizations

change when these are of a specific level in terms of experience, education, product maturity, or company size.

The paper is structured as follows: Sect. 2 presents an outline of the related work. Section 3 provides research methodology and discusses the basic principles of CoDA along with various challenges related to its application. Section 4 presents results from descriptive statistics, the application of non-parametric tests and the CoDA framework on survey data. Finally, in Sect. 5, conclusions and directions for future work are provided.

2 Related work

Component-based software systems require decisions on component origins for acquiring components. A component origin is an alternative option of where to acquire a component. A recent systematic literature review about CSO selection (Badampudi et al., 2016) also investigated decision criteria, methods for decision making, and evaluations of the decision results. The paper highlighted that the CSO comparison was mainly focused on in-house vs. COTS and COTS vs. OSS. In a recent case survey (Petersen et al., 2017), 22 case studies of how practitioners choose between CSOs were investigated. One of the conclusions was that the most frequent trade-offs are carried out between in-house vs. COTS, in-house vs. outsourced, and COTS vs. OSS. In-house was the most favored decision option; however, the evaluation of the decision showed that many of the decisions were perceived as suboptimal, indicating a need for optimizing the decision-making process and outcomes.

In the survey of Borg et al. (Borg et al., 2019), the most important challenges related to CSO selection were identified and fell within the following three types: managerial, functional, and non-functional (quality-related). There have not yet been any attempts to identify the significance of these challenges to the best of our knowledge.

Several primary studies discussing in-house vs. COTS CSO decisions exist, i.e., (Brownsword et al., 2000; Li et al., 2006a). In (Cortellessa et al., 2008), a framework was presented to support the decision to buy components or build them in-house. The authors in (Li et al., 2006b) studied decisions made during the integration of COTS vs. OSS and showed significant differences and commonalities.

Cumulative voting (CV) is known as a prioritization technique, used for decision making in various areas. CV has been used also in various areas of Software Engineering, such as requirements engineering, impact analysis, or process improvement (Regnell et al., 2001; Berander & Wohlin, 2004). Prioritization is performed by stakeholders (users, developers, consultants, marketing representatives, or customers), under different perspectives or positions, who respond in questionnaires appropriately designed for the purposes of prioritization. CV has been proposed as an alternative to the Analytical Hierarchy Process (AHP) and its use is continuously expanding to areas such as requirements prioritization and the prioritization of process improvements (Leffingwell & Widrig, 2003; Firesmith, 2004).

In (Regnell et al., 2001), CV is used in an industrial case study where a distributed prioritization process is proposed, observed and evaluated. The stakeholders prioritized 58 requirements with $100,000 to distribute among the requirements (the large amount of "money" was chosen to cope with the large number of requirements). In (Staron & Wohlin, 2006), CV was used for an industrial case study on the choice between language customization mechanisms. In (Hatton, 2008), CV is one of the four prioritization methods examined, evaluated and recommended for certain stages of a software project. In (Chatzipetrou et al.,

2010), 18 interviewees were asked to prioritize 25 aspects using CV by distributing 1000 imaginary points to the aspects. Each interviewee prioritized the 25 aspects twice: Under the organizational perspective and under the self-perspective. The data were collected during an empirical study on the role of impact analysis (IA) in the change management process at Ericsson AB in Sweden. CoDA had also been used in the software effort phase distribution analysis (Chatzipetrou et al., 2015; Chatzipetrou et al., 2012).

In the previous work of the authors (Chatzipetrou et al., 2018), CoDA was used for the visualization of the inherent characteristics of the practitioners. The work is extended in this paper, with a deeper investigation into which attributes are the most important for the decision process related to CSOs. Box plots provided us with insights about the most important attributes in each different selection case. For this purpose, an exploratory study was rigorously conducted with regard to CSOs selection and the inherent characteristics of the participants, which led us to the visualization of relevant results.

3 Research methodology

3.1 Research questions

In this paper, we used the experience from our former studies on CV and on the selection of different CSOs in order to investigate why practitioners chose one CSO over another and why they chose specific combinations of CSOs. A thorough study was conducted based on their choices. Moreover, we aimed to discover if there were any trends among the practitioners based on their inherent characteristics, e.g., their current role or the number of years they have been active in the industry. Therefore, in the present study, we investigate the reasoning behind decision making in component selection based on the practitioners' inherent characteristics. The main contribution is to understand and explain the decision-making process of practitioners in CSO selection. The methodology is applied to real survey data, in order to draw interesting and useful results regarding the practitioners' decision processes.

Our work was driven by the following research questions (RQs):

- *RQ1: What matters the most to industry practitioners when selecting CSOs?*

We wanted to explore what matters the most to industry practitioners when selecting CSOs for their existing or new projects. In order to investigate which information is the most important input for their decision, we used a set of attributes (defined and presented in Table 1) and we asked the practitioner to prioritize those attributes by using the CV technique.

- *RQ2: Is the decision process regarding the CSO affected by the practitioners' known characteristics (i.e., role, working experience and education)?*

We wanted to explore if the decision process regarding the CSO was affected by the practitioners' characteristics. The available characteristics from this study are defined and presented in Sect. 4.3. We mainly investigated and explored trends and peculiarities among our population by using a powerful descriptive tool specially designed for this type of data: the biplot.

The above RQs are the starting point in our investigation and will help us drive our research and gain further understanding into the decision-making process in CSOs.

Table 1 Attributes used for prioritization

	Attributes	Description
1	Size	Size of the component, e.g., lines of code, memory footprint
2	Longevity prediction	Evolution of the component
3	Cost	Development, license, and maintenance cost
4	Level of off-the-shelf fit to product	Functional fitness, i.e., how much component customization is needed
5	Complexity	Code complexity
6	API adequacy	Maturity of external APIs
7	Programming language performance	Computational performance
8	Access to relevant documentation	Access to documentation
9	Code quality	Availability of automated tests, code review practices
10	Support of the component	Formal support, channels, active development community
11	Adherence to standards	Follow the rules
12	Other	Licensing issues, vendor relations

3.2 Description of the data set

An anonymous, cross-domain, industrial survey was conducted that aimed to identify the relationship between practitioners' decision making and which CSOs were chosen. The survey questionnaire was web-based and consisted of a number of both open-ended and closed-ended questions. The practitioners were asked to choose between four different CSO. Moreover, the practitioners were free to choose more than one CSO if they believed that to be necessary.

The CSO decisions can be summarized in the following four alternatives (Petersen et al., 2017; Borg et al., 2019):

- *Software developed internally (in-house)*: This is the case where a company develops a component internally. In addition, development is still considered in-house when the development is distributed in different locations, as long as it takes place within the company. The source code is developed and remains inside the same company.
- *Software developed outsourced*: Another company develops the component on behalf of the company that wants to obtain the component. Usually, the source code is delivered as part of the contract agreed upon between the two companies.
- *Commercial of the shelf software*: The company buys an existing component from a software vendor (pre-built). The source code is not available for the buyer.
- *Open-source software*: The company integrates a pre-built, existing component that has been developed by an open source community as an open-source software. The source code is publicly accessible.

Practitioners were asked to choose between the above-mentioned four CSOs and indicate which information was the most important input for their decision process. The attributes were chosen after a joint research effort was conducted with several senior researchers within the project research team (the Orion research team). In addition, an external senior software engineering researcher was invited to review the chosen attributes. Moreover, a native English speaker reviewed the attributes from a language perspective. The attributes' names and descriptions were refined to avoid potential ambiguities. An effort was put towards the common understandability of the attributes between the practitioners. In particular, a detailed description was included for each one of the attributes in order to avoid misunderstandings. At

the next evaluation stage, a pilot run was conducted. We invited 15 independent researchers to act as test pilots and evaluate the entire survey. We used their feedback in order to refine the attributes description. For instance, an "Other" category was added.

Finally, 12 attributes were chosen (Table 1 presents the attributes in the same order they appeared in the survey). The practitioners were asked to prioritize 12 attributes using CV by distributing 100 imaginary points. The number of the respondents involved was 157. The complete description and design of the survey is available in (Borg et al., 2019).

3.3 Data analysis

a) *Descriptive statistics and box plots*

Descriptive statistics involve the computation of simple summary statistics like minimum and maximum values, the mean, standard deviation, and the median of the data. Descriptive statistics are computed separately for the whole set of the data and separately for each combination of CSO selections. The statistics are accompanied by graphical representations like radial bar charts and box plots.

b) *Cumulative voting*

CV or the 100-point method or hundred-dollar ($100) test, described by Leffingwell and Widrig (Leffingwell & Widrig, 2003), is a simple, straightforward, and intuitively appealing voting scheme where each stakeholder is given a constant amount (e.g., 100, 1000, or 10,000) of imaginary units (or imaginary currency) that he or she can use to vote in favor of the most important attributes. In this way, the amount of money assigned to an attribute represents the respondent's relative preference (and therefore prioritization) in relation to other attributes. The points can be distributed in any way the stakeholder desires. Each stakeholder is free to put the entire amount given to them on only one attribute of critical importance. It is also possible for a stakeholder to equally distribute the amount to many, or even all, of the attributes.

However, since the results from the hundred-dollar ($100) test sum up to 1, we cannot treat them as independent variables and since they are restricted to the [0,1] interval, normality assumptions are invalid. A methodology that is suitable for the analysis of proportions is CoDA. This methodology has been widely used in the analysis of material composition in various scientific fields like chemistry, geology and archeology, but its principles fit the analysis of data obtained by CV.

c) *Compositional data analysis*

CoDA is a multivariate statistical analysis framework for vectors of variables having a certain dependence structure: The values of each vector have a sum equal to a constant. Usually, for easy reference of the same problem, after dividing by that constant, the sum of the values of each vector becomes one. The important point here is that data is constrained to the [0,1] interval; therefore, the techniques applied to samples from the real Euclidean space are not applied in a straightforward manner.

Concerning the prioritization questionnaires using the $100 test, the data essentially represents proportions of the overall importance allocated to each of the aspects examined in a study. The relative importance of the aspects is represented by their ratios, so CoDA seems to

be the appropriate framework for their study. Historically, Karl Pearson in 1897 (Pearson, 1897) posed the problem of interpreting correlations of proportions while the milestone for this type of statistical analysis is the pioneer work of John Aitchison (Aitchison, 1982; Aitchison, 2003). The freeware package CoDaPack3D (Comas-Cufí & Thió i Fernández de Henestrosa, 2011) was used for compositional data analysis.

The data from the CV questionnaires have some special characteristics that can cause problems in the analysis. The problem of zeros is of principal importance. When the number of attributes is large, and the individuals are only few, the data matrix is usually sparse, with a large number of zeros. The presence and meaning of zeros in a dataset can be of crucial importance. Two types of zeros exist. Essential zeros it could imply complete absence of an attribute on the other hand, rounded zeros refer to the instrument used for the measurements where they did not or could not detect the attribute. In our study, the meaning of a zero proportion is interpreted as rounded zeros as the zeros are results from lack of recording. However, this structure causes problems of interpretation when we consider their relative importance. In order to address that problem, we used a simple method proposed by (Dunn, 1959), known as multiplicative replacement strategy and according to it every zero value is replaced with a very small value (0.01) and the rest of the vectors are adjusted and recalculated in order to sum up to 1. The advantages of multiplicative replacement are discussed extensively in (Martín-Fernández et al., 2000; Martín-Fernández et al., 2003a; Martín-Fernández et al., 2003b).

To treat the data with "common" statistical techniques we needed to transform them. Aitchison (Aitchison, 2003) proposed the centered log ratio (CLR) transformation for transforming a raw proportional dataset to real space and at the same time retaining its correlation structure.

In order to visualize the results from the CoDA, we used Biplot (Gabriel, 1971; Gabriel, 1981; Aitchison & Greenacre, 2002), which is a straightforward and useful tool for exploring trends and peculiarities in data. Its basic characteristics are the lines (or rays) and dots. Rays represent the variables, and dots represent the respondents. An important characteristic of the plot is the angle between the rays. The length of a ray shows the variance of the corresponding variable. Longer rays depict higher variances. A link is an imaginary line connecting the ends of two rays. It essentially shows the difference between the two variables. Large links show large proportional variation. It is essential to emphasize that, in terms of interpretation, links are considered more important than rays since the variables can be examined in a more relative and intuitive manner. Finally, the cosine of the angle between the rays approximates the correlation between the CLR transformations of the variables. The closer the angle is to 90°, or to 270°, the smaller the correlation. An angle nearer to 0° or 180° reflects strong positive or negative correlation, respectively (Aitchison & Ng, 2005). An extensive presentation of the CoDA framework is discussed in (Chatzipetrou et al., 2010; Chatzipetrou et al., 2015; Chatzipetrou et al., 2012).

3.4 Validity threats

The validity threats are distinguished between four aspects of validity according to Runeson and Höst (Runeson & Höst, 2009):

Construct validity reflects the extent to which the operational measures represent the study subject. In the present study, practitioners' views are measured on a numerical scale. However, since the practitioners work for different organizations offering different products, their views on component selection may differ. However, a deeper analysis on practitioners' decision

processes based on their inherent characteristics was valuable. The major threat to our study was whether our inquiry about previously experienced CSO decisions truly reflects a phenomenon in the industry. We addressed this by developing the questionnaire into a joint research effort with several senior researchers followed by a pilot run with a handful of selected respondents. Furthermore, our list of attributes appears to be rather comprehensive as the number of *Other* answers is low. Our initial construct captured CSO decisions and component selection as two separate activities, but our construct evolved during the study.

Internal validity refers to the examination of causal relations, which is the intended outcome of our investigation. In our case study, we focused on how the different inherent characteristics of the practitioners, i.e., the role, size of the company, etc., affect the decision process with regard to which CSO is chosen.

Regarding *external validity*, the study is clearly empirical and by no means can the findings be generalized to other software development organizations with similar characteristics. The population under study, i.e., practitioners involved in architectural decision making in component-based software evolution, is large and highly heterogeneous. Under these circumstances, the highest importance was to select a representative sample. 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. The practitioners who participated in the survey do not constitute a random sample; however, they were approached for their experience and expertise, so their responses are considered valid.

Regarding *reliability*, this aspect is concerned with the extent to which the data and the analysis are dependent on specific researchers. Hypothetically, if another researcher later on conducts the same study, the result should be the same. The data gathered are quantitative and independent from the influence of different research subjects or researchers' interpretation. The survey questions were piloted with a set of practitioners who provided feedback and improvement suggestions. A statistical analysis was performed with reliability in mind, with each step documented for potential replication.

4 Results

4.1 Descriptive statistics

a) *General results*

The results from the descriptive statistics are available in Table 2 and Fig. 1, where Table 2 summarizes the answers of the practitioners in absolute numbers and from two perspectives: the number of practitioners that chose an attribute and the total points an attribute received from all the practitioners. The practitioners were free to select any number of attributes. The attributes are in descending order, based on the practitioners' choices.

The analysis showed that *cost* is considered the most important attribute from the majority of the practitioners when making CSO decisions, selected by 121 out of 157 practitioners (77%). The inputs that are also considered important in the decision process and are mentioned by roughly half of the practitioners are: *support of the components* (74 out of 157, 47%), *longevity prediction* (71 out of 157, 45%), and *level of off-the-shelf fit to product* (62 out of 157, 40%). An interesting finding is that support of the components is the second most popular choice among practitioners. However, it is ranked 4th in the total points received and the

Table 2 Descriptive statistics

Attributes	Min	Max	Mean value	Std. deviation	Number of respondents	Total points
Cost	0	100	25.06	21.07	121	3935
Support of the component	0	50	8.55	11.46	74	1343
Longevity prediction	0	60	9.64	13.27	71	1513
Level of off-the-shelf fit to product	0	75	9.55	15.44	62	1499
API adequacy	0	100	7.61	12.92	56	1195
Access to relevant documentation	0	100	6.69	12.73	53	1050
Code quality	0	100	8.15	14.58	53	1280
Adherence to standards	0	50	5.31	10.57	42	833
Programming language performance	0	100	7.08	15.79	41	1112
Complexity	0	50	4.43	10.41	33	695
Size	0	60	2.77	9.63	18	435
Other	0	100	5.16	16.88	18	810

maximum amount of points it got from a practitioner is 50. In other words, the practitioners did not assign high values to this choice; however, they still consider it an important factor that should be taken into account. The same is true for longevity prediction (maximum amount of points, 60; however, it is second in the total amount of points received). We can hypothesize that complexity (and also maintenance costs) are likely to increase as more components are added from CSOs. Thus, if one considers *cost* as the most critical attribute, then *complexity* should matter. However, here components are probably seen as black box and their complexity is not seen as important.

On the other hand, *size* was not popular, as it was selected by only 18 out of 157 practitioners (11.5%). This rather surprising finding may suggest a discrepancy in views with regard to the total cost of acquisition and integration of a software component into the development environment. As pointed out by Jørgensen and Shepperd (Jorgensen & Shepperd, 2007) most research on software cost estimation focuses on the introduction and evaluation of estimation methods. Our hypothesis is that larger components require substantial

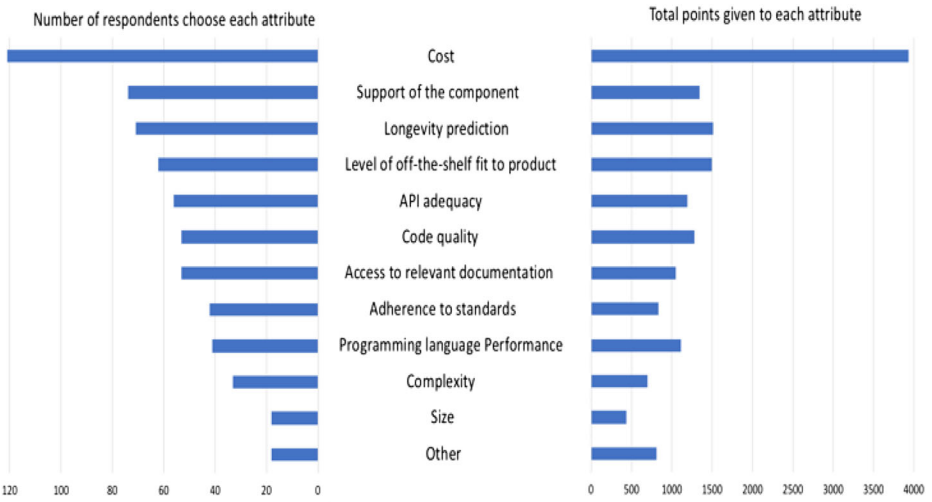


Fig. 1 Number of respondents vs. total points selected for each attribute

more cost and effort to be integrated into the development organization on top of their acquisition cost (e.g., purchase cost), as this hypothesis is also supported by previous work on integration and maintenance costs (Abts et al., 2000; Nguyen, 2010). Therefore, we believe that this discrepancy should be further explored by researchers and emphasized to practitioners making CSO decisions.

The number for other is also low (18 out of 157, 11.5%) which reveals that the list of the given attributes covers the most important criteria needed to evaluate a component by practitioners. Among the other answers, the most frequent responses include: licensing issues and long-term strategies such as differentiation in the market and vendor relations. Licensing issues are often highlighted as a primary obstacle by companies that want to utilize OSS components, as is organizational resistance, which can be overcome by increased education and knowledge regarding OSS license types. However, these were responses mentioned by just 18 practitioners and thus do not raise any threats to the attributes selected to be included in the survey.

4.2 Results from research questions: RQ1

What matters the most to industry practitioners when selecting CSOs?

We use radial plots and box-plots to further investigate which attributes affect the selection of certain CSOs or combination of CSOs in industry, i.e., how the preferences on specific CSOs affect the prioritization of the 12 attributes. Figure 2 shows the summary of which CSOs the 157 practitioners typically considered, and they could select between one and four options (multiple options were allowed for this question). The majority of the practitioners (90%) consider software developed internally either as the sole option or in combination with other

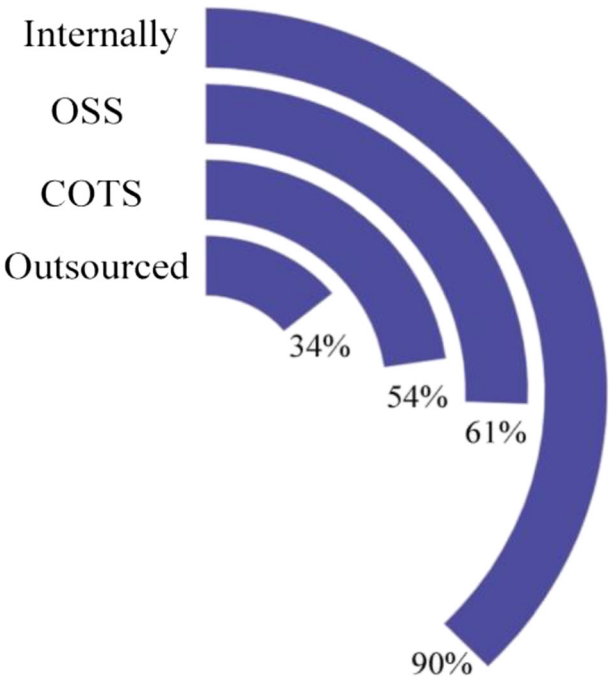


Fig. 2 Which CSOs the 157 practitioners choose?

CSOs. Moreover, more than half of the practitioners considered OSS and COTS to be possible options, at 61% and 54%, respectively. The least considered option (34%) was outsourcing. It is evident that more than one CSO are typically considered by practitioners. A potential interpretation could be due to the high pace of technology acceleration, requirements on maintenance and upgrades, which typically leads to migration and put extra pressure on practitioners to consider alternative CSO practices. An interesting aspect to investigate is at which product maturity or software lifecycle stage this happens and to what types of organizations. Moreover, in the present work, we were interested in studying the specific combinations of CSOs and on which attributes the practitioners based their choices.

Figure 3 depicts the choices of 157 practitioners regarding their selection of CSOs, or a combination of CSOs. The results showed that almost the 24% of the practitioners considered only one CSO when they were adding or buying a new component. Additionally, a strong majority of them (73%) only considered software developed internally when deciding among CSOs. A very small percentage of practitioners would only consider open-source software (16%, or 4% in total) and outsourced software (8%, or 2% in total), but no practitioners considered COTS as the sole choice for adding new software. The fact that only 4% of respondents would consider only OSS option is interesting and worth exploring further. Considering that many software products have a commodity functionality layer and a differentiation functionality layer (Bosch, 2018), they are treated differently. The commoditized functionality layer is typically optimized for minimizing the total cost of ownership and selecting OSS components could potentially provide an optimal solution for the cost of ownership minimization (Linåker et al., 2018). It appears that our respondents have not yet uncovered the potential of using only OSS components in the commoditized layer or there may be other obstacles or regulatory forces (Sulaman et al., 2014) to prevent them from doing that and thus they have to consider more than one CSO source.

More than one quarter (27.5%) of the reviewed practitioners would choose a combination of two CSOs. Among these, about half of them (13.5%) would opt for software developed internally and OSS and 8.5% would develop software internally and use COTS. However, a significant

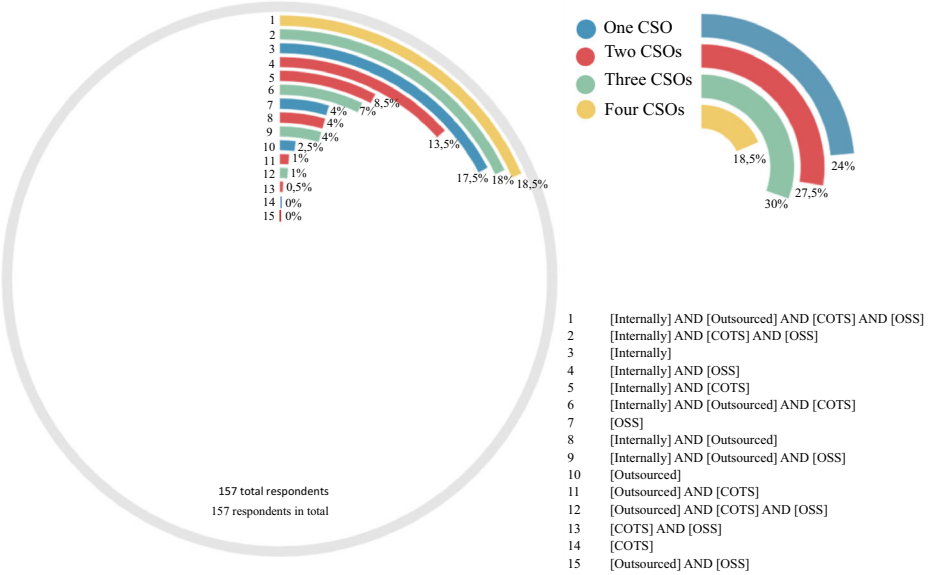


Fig. 3 Which CSOs or combination of CSOs the 157 practitioners chose?

number of practitioners (30%) would choose a combination of three CSOs, whereas 18% of them would choose to develop software internally, using OSS and COTS; 18.5% of the practitioners' reported that they would consider all four CSOs when adding or buying new components. Moreover, the majority of the practitioners (57.5%, see Fig. 3) consider two or three CSOs. A possible interpretation could be the lack of competence in the area where the component is needed, e.g., a company needs a database component that is not the core differentiator of the product but is a necessary "enabler" for the differentiating functionality to be offered to the customers. Building a database solution internally is neither feasible nor justified from an economic standpoint. Another explanation could be time pressure, which forces decision makers to look for alternative solutions other than in-house development. Obtaining the necessary components externally has the potential of reducing time-to-market to integration and system testing time.

From Fig. 3, we can conclude that software developed internally is primarily considered a CSO among practitioners, either as a sole option or in combination with other CSOs since the top-ranked choices include software developed internally. It is also shown that the second most popular option combined with internal software development is OSS (13.5%), followed by COTS (8.5%). Only a very small number of practitioners combine internal software development with outsourcing (5%) or outsourcing and OSS (4%). This is probably because it could cause an increase in costs in the quality evaluation of an integrated solution and the monitoring of the outsourced solution's quality.

We generated box plots representing practitioners' preferences separately for each combination of CSOs and grouped by the number of the CSOs they have considered. These are presented in Figs. 4, 5, 6, and 7.

Figure 4 illustrates how the 12 different attributes are prioritized by the practitioners who choose only one CSO to add or buy for their new system. It is clear that cost is the most important attribute. However, there are variations regarding how high cost is prioritized in each case. Moreover, the results show that for different CSOs, different attributes are important. Thus, to the practitioners that choose only to develop their software in-house (internally, Fig. 4), cost is the most important attribute, however they did not assign high values to this attribute (maximum 50) when *programming language performance* and longevity prediction

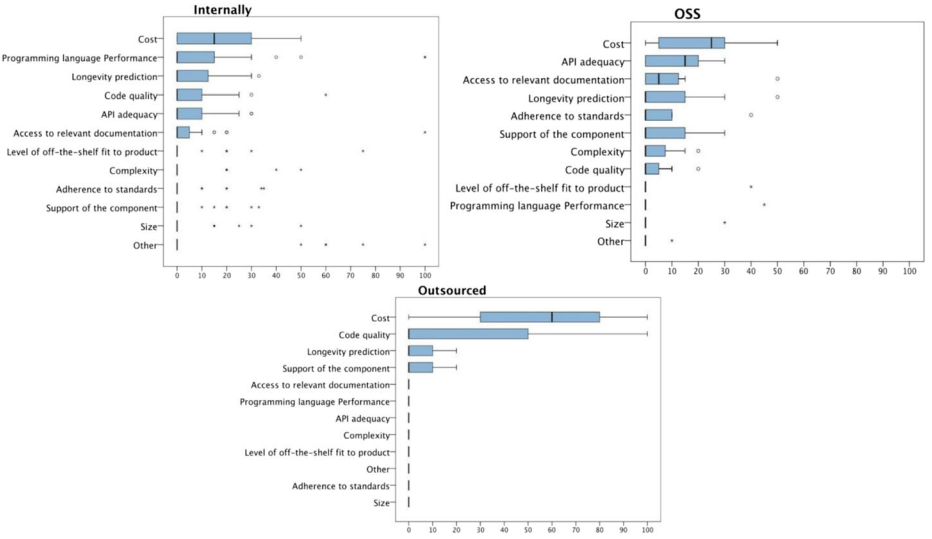


Fig. 4 Which attributes are the most important for the practitioners who chose one CSO?

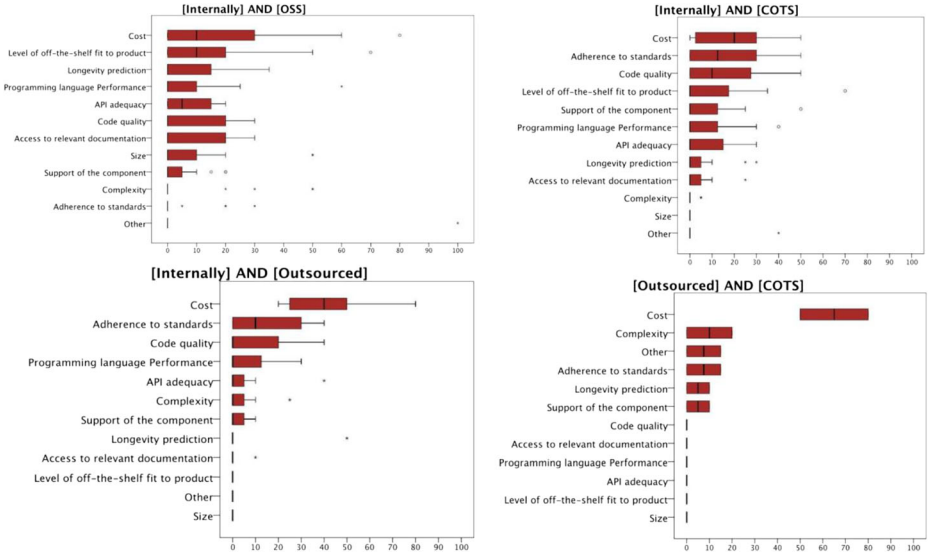


Fig. 5 Which attributes are the most important for the practitioners who chose two CSOs?

are also highly ranked. Regarding cost, the same is true for practitioners that chose only open-source software development. However, the practitioners that chose only this CSO prioritize higher *access to relevant documentation* and *API adequacy.*

The results are different for practitioners who only consider outsourcing software development (outsourced, Fig. 4). Cost is still here the highest-ranked attribute, but the practitioners also assign more importance to this attribute. *Code quality* is also very important, along with the support of the component and longevity prediction. The other attributes are of no importance to practitioners.

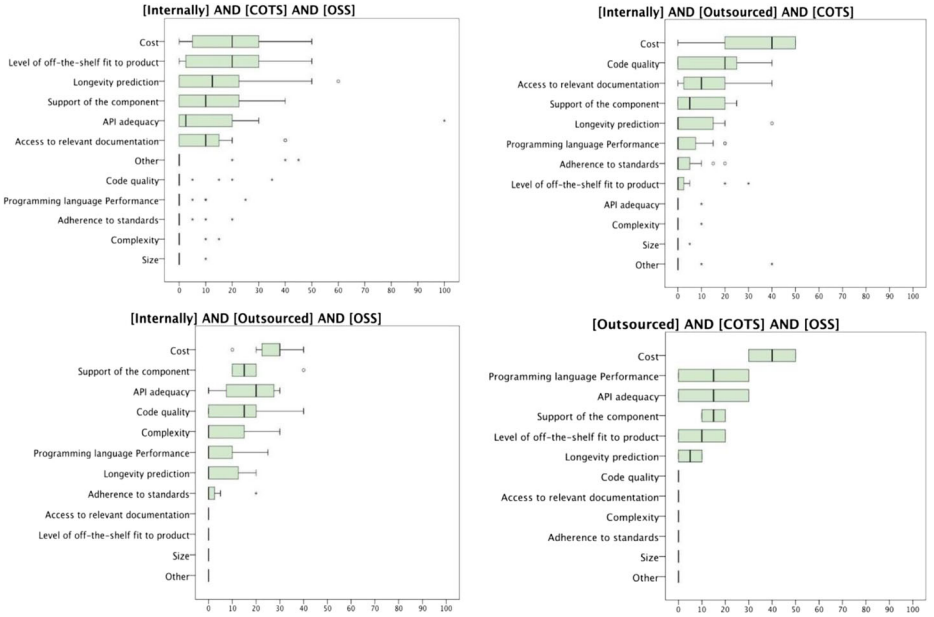


Fig. 6 Which attributes are the most important for the practitioners who choose three CSOs?

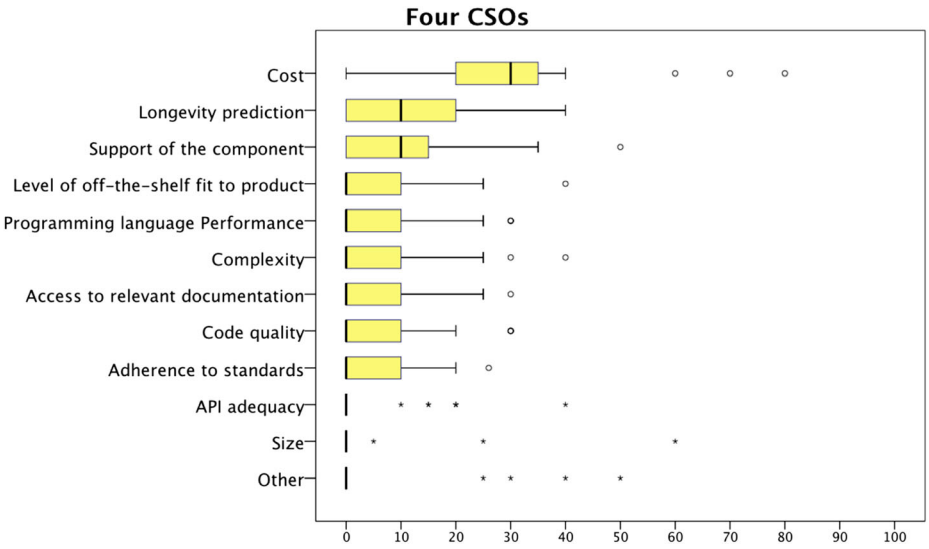


Fig. 7 Which attributes are the most important for the practitioners who chose four CSOs?

The results are not surprising, since in the outsourcing scenario, support for the delivered component is as important for the practitioners as the quality of the code. At the same time, our respondents seem to be less concerned about the code quality for internally developed code than for outsourced code. In other words, they believe that the internal teams would not have significant quality issues and have confidence that they can deliver high quality code. We believe that this assumption may not always be true and that some outsourcing partners could actually deliver code with higher quality than internal developers, mainly due to experience and expertise.

The practitioners that typically consider two CSOs also assign higher values to *cost* (Fig. 5). However, when they choose to develop their software in-house, in combination with open source software, *level of off-the-shelf fit to product* is their next priority. One possible explanation for this result is the inability to order functionality from an open-source community. OSS communities operate based on the meritocracy principle, where the most prominent contributors (individuals or organizations) have the most influence on a decision. Therefore, if an organization uses an OSS component without substantial contributions, the guarantee that it is compatible with the internally developed product is managed by the organization. Thus, many software organizations that use OSS components investigate the stability of interfaces and the health of the OSS ecosystems (Baars & Jansen, 2012) before joining or using their software. *Adherence to standards* and code quality are important attributes for the practitioners that consider developing their software in-house in combination with outsourcing or COTS. Adherence to standards suggests that these participants probably take responsibility for product software certification and therefore need to ensure that the third party providing software delivers it according to the required standards and with comprehensive documentation. Cost is particularly important to practitioners that combine outsourcing and COTS (Fig. 5d). This, however, may be due to the small number of samples.

Furthermore, a similar pattern appears when the practitioners consider three CSOs for their new components. Cost is the most important attribute, but the practitioners did not give it a high value (maximum 50). A possible explanation is that even if cost is an important attribute, other attributes are important too, i.e., level of off-the-shelf fit to product, code quality, support of the component, and programming language performance (Fig. 7).

Finally, when the practitioners considered all four CSOs: cost along with longevity prediction, and support of the component are the attributes that were ranked higher (Fig. 7). One of the interpretations of the results is that the more options the practitioners have, the more perspectives they include, or perhaps different perspectives are taken under consideration, which are greatly ignored when only one CSO source is considered. This confirms the need for supporting tradeoffs between various CSO sources and assisting decision makers in choosing between various CSOs for software components.

4.3 Results from research questions: RQ1

Is the decision process regarding the CSO affected by the practitioners' known characteristics (i.e., role, working experience and education)?

a) Non-parametric tests

For the next step, we investigated how the inherent characteristics of the practitioners influence their decisions regarding which CSO to choose. For this purpose, the prioritized data was transformed with the methods of CoDa described in the previous section (replacement of zeros and CLR transformation) and Kruskal-Wallis. A non-parametric test (Kruskal & Wallis, 1952) was applied to the obtained data. The test was performed in order to investigate the distribution of each attribute across all demographic characteristics. Pairwise comparisons were also performed using Dunn's (Dunn, 1964) procedure, with a Bonferroni correction (Dunn, 1964) for multiple comparisons. The results revealed that there are no statistical differences ($p > 0.05$) between:

- The different roles of the practitioners (a description of the roles is available in Table 3),
- Practitioners with different working experience, and
- The education of the practitioners

and the way the practitioners prioritize the 12 aspects.

Therefore, in order to explore trends and peculiarities among our population, we need a powerful descriptive tool designed for this type of data: the biplot.

b) CoDA analysis—biplot

1) All practitioners

Figure 8 illustrates the biplot for the prioritization of 12 aspects by all 157 practitioners. The practitioners are represented by dots, while the rays represent 12 attributes. The results showed that there is a wide distribution of the dots in all the axes, and long rays (thus long links too) for most of the aspects. These indicate high dissimilarity between the practitioners, large variability among the aspects and some interesting correlations between variables. The biplot clearly depicts the high level of complexity of the decision process to prioritize attributes and illustrates how dissimilarly the practitioners prioritize each choice. More specifically:

- The longest rays correspond to level off-the-shelf fit to product, code quality, cost, and longevity prediction indicating the aspects with the highest variance. In other words, practitioners allocated values from 0 to 100 to those attributes.

Table 3 Description of roles

Role	Description
Strategic management	An individual who focuses on developing long-term technology goals, e.g., a chief technology officer (CTO)
Product planning	An individual who focuses on internal decisions, steps, and tasks to develop a successful product, e.g., a product manager
End-user perspective	An individual who works with the project stakeholders and end users to elicit and understand the system's requirements, e.g., a requirements engineer
Operational management	An individual who is responsible for planning, organizing the development of the product, e.g., a project manager
Product development	An individual which is responsible for bringing the product from a concept to idea, e.g., a developer or a team leader
System view/architecture	An individual who is responsible to define the structure and the behavior of a system, e.g., a system view/architecture
Software maintenance/evolution perspective	An individual who is responsible of the modification of the system after it has been delivered to the customer
Quality assurance/system testing	An individual who is responsible to check whether the actual results match the expected results, e.g., a software tester
External business perspective	An individual who is responsible to be constantly updated with market trends
Internal business perspective	An individual who is responsible to oversee the financial reports, e.g., a controller financial
Legal perspective	An individual who is responsible for the legal aspects of the software development
Other	Other roles that are not covered from the previous categorization, e.g., consultant, researcher

- The longest links are the ones between cost and *access to documentation*, between longevity prediction and code quality and between level off-the-shelf fit to product and adherence to standards. Therefore, the largest differences, considering all aspects together, are located between these pairs of variables, which also seem to be negatively correlated (due to the nearly 180 angles that they have). For example, if a practitioner chooses to

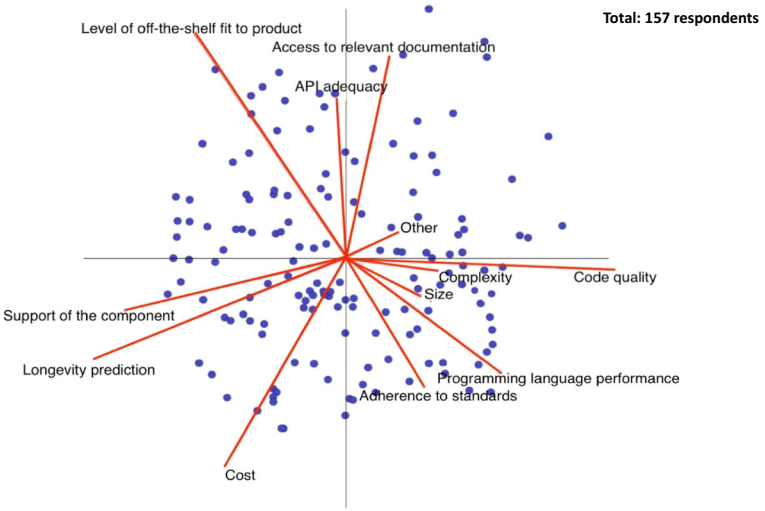


Fig. 8 All practitioners prioritize all attributes

allocate more monetary points to cost, then we assume that he/she will assign almost 0 points to access to documentation and vice versa.

- The shortest link connects the ray ends of size and complexity and indicates that the distribution in those two aspects is quite similar and positively correlated (due to the nearly zero angle that they have). Longevity prediction and *support of a component* seem to be positively correlated as well. In other words, the practitioners tend to allocate the same amount of points for the aforementioned aspect. For example, size and complexity.
- The nearly orthogonal pairs of variables (the rays that form right angles with each other), such as the ones corresponding to cost and code quality or level off-the-shelf fit to product indicate correlation of these aspects with cost close to zero, which means that we can claim that the way a practitioner allocates the points for the two above-mentioned attributes is not related, either positively or negatively.
- The angle between support of the component and longevity is close to zero, which indicates a positive correlation, which is not surprising since longevity is connected with suitable support. Otherwise, the organization has to maintain the component, which could be costly. Moreover, the same positive connection exists for API and level of off-the-shelf fit to product. API is essentially the interface that allows the connection of the component to the software. Hence, if the API is not adequate, the component will not fit the product, regardless of how well the functionality of the component is provided by the component.

Regarding the distribution of the practitioners with respect to the attributes prioritized, there are areas of high density as well as areas of low density. This means that groupings of practitioners exist, i.e., there is a group of high density near cost which means that a significant number of practitioners have assigned a larger proportion of effort to cost.

For the next step, a deeper analysis of the practitioners' decision process was conducted based on their inherent characteristics. The characteristics that were investigated were related to the role a practitioner holds in the company, their general working experience and educational level, the maturity of the product they are working on and the size of the company they belong to. Each group of practitioners was aggregated by computing the mean value of their preferences. In the following figures the groupings of the practitioners' inherent characteristics appear in dots in italics.

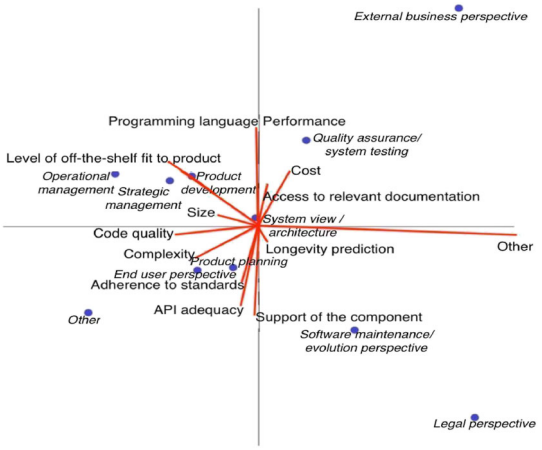
2) Role

Figure 9 illustrates a biplot of the practitioners grouped by their role within the company. It is clear that outside employees and those employees working with legal issues have completely different aspect prioritizations than the rest of the workforce. More specifically, they need to decide on changing a component based on other issues, i.e., licensing issues or vendor relations.

On the other hand, the practitioners who work with management (strategic and operational) but also product developers on average take into consideration the level of off-the-shelf fit to product and code quality.

3) Working experience

From Fig. 10, it is clear that employees with little work experience on average are more interested in issues related to level of off-the-shelf fit to product, access to the relevant documentation and API adequacy. On the contrary, more experienced employees on average focus more on code quality and the support of the component.



Responses:

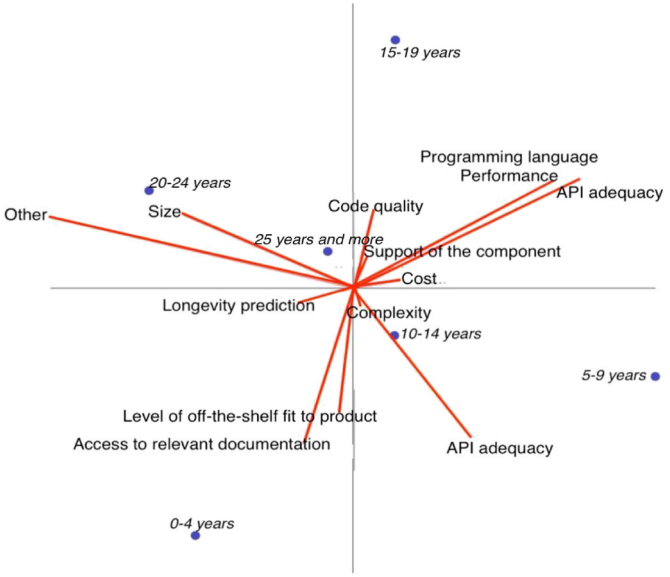
Strategic management:	17
Product planning:	17
End-user perspective:	6
Operational management:	7
Product development:	51
System view/architecture:	33
Software maintenance/evolution perspective:	3
Quality assurance/system testing:	12
External business perspective:	2
Legal perspective:	2
Other:	7
Total	157

Fig. 9 Practitioners grouped by their role

4) Educational level

Those practitioners with a university education on average seem to consider similar attributes for their decision process (i.e., level of off-the-shelf fit to product and access to the relevant documentation). Size also seems to be considered among practitioners with an academic background. When practitioners who attended professional courses or have a trade school education chose a new component, they considered attributes related with development and the usage of the new component to be more important, i.e., complexity, programming language performance, and code quality. (Fig. 11).

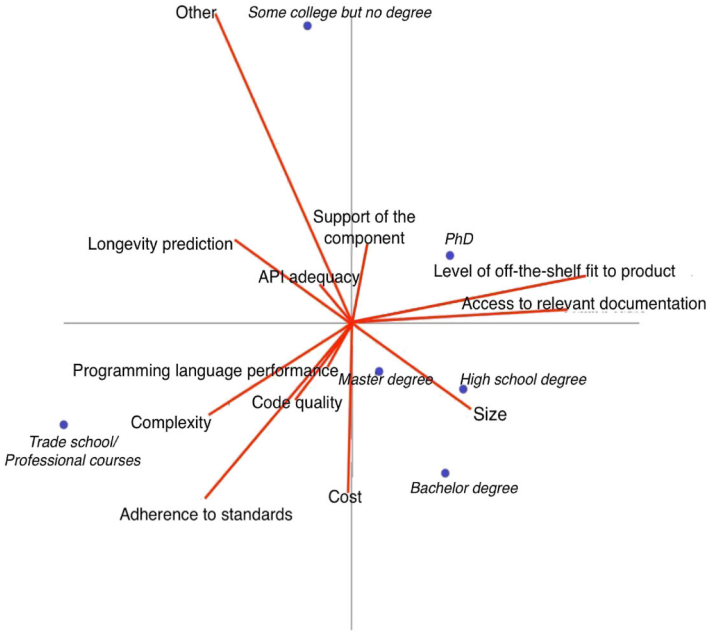
5) Maturity of the product



Responses:

0-4 years:	17
5-9 years:	25
10-14 years:	38
15-19 years:	32
20-24 years:	21
25 years and over:	24
Total:	157

Fig. 10 Practitioners grouped by their work experience

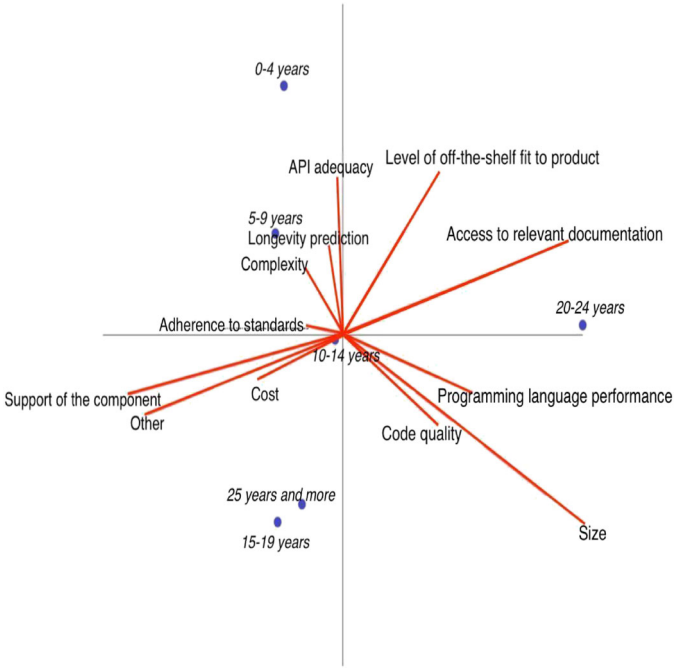


Responses:

High school degree or equivalent:	5
Some college but no degree:	8
Bachelor degree:	28
Master degree:	81
Ph.D:	31
Trade school/Professional courses:	4
Total:	157

Fig. 11 Practitioners grouped by their education

From Fig. 12, we can claim that the practitioners who work on more mature products (more than 15 years) place more emphasis on non-functional attributes



Responses:

0-4 years:	34
5-9 years:	29
10-14 years:	21
15-19 years:	18
20-24 years:	5
25 years and over:	50
Total:	157

Fig. 12 Practitioners grouped by the maturity of the product they are working with

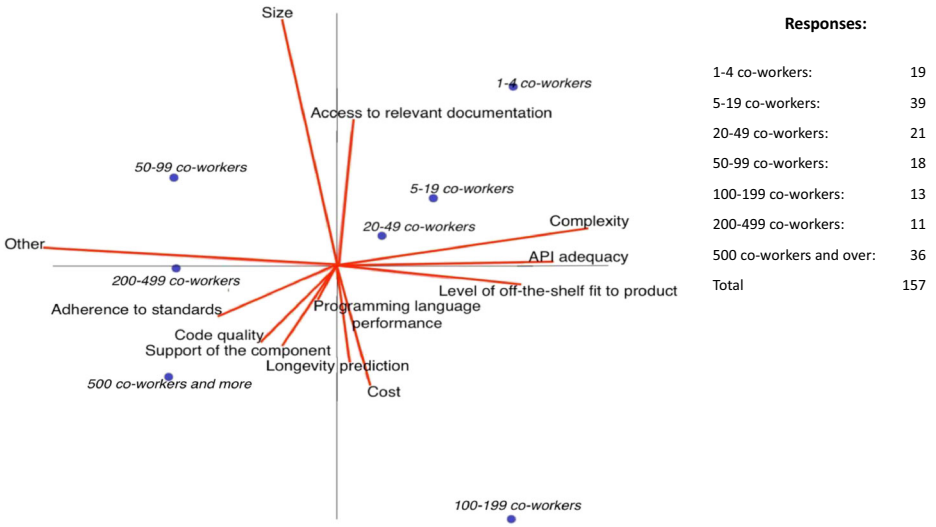


Fig. 13 Practitioners grouped by the size of their company

(i.e., size). However, cost is still their first priority. On the other hand, practitioners who work with less mature and newly established products (less than 10 years) seem to be more interested in complexity and API adequacy, and they are not focused as much on the cost.

6) Size of the company

Regarding the size of the company a practitioner works at, the results are available in Fig. 13. It seems that smaller organizations focus more on development and maintenance of the component (complexity, API adequacy, and access to relevant documentation). On the other hand, bigger organizations focus on properties associated with cost.

5 Conclusions and future work

The decision for an organization whether to develop components internally or to acquire them from external sources is crucial. The present study focuses on the investigation of what matters the most to industry practitioners during component selection. In this work, we focused on four CSOs: (1) Software developed internally (in-house), (2) Software developed outsourced, (3) COTS, and (4) OSS. The practitioners were free to choose more than one CSO if they believed that to be necessary. While choosing between the four CSOs, the practitioners had to indicate which information was the most important input for their decision process by prioritizing 12 attributes. Since few studies regarding CSO selection exist, the main contribution of our work is into the understanding of the CSO decision and selection. Firstly, we performed a descriptive of our data. The results show that cost is clearly considered to be the most important attribute during the selection of a component. Other

important attributes for the practitioners were: support of the component, longevity prediction, and level of off-the-shelf fit to product. Moreover, we examined our data in depth and we focused on the following two research questions.

- RQ1: We wanted to explore what matters the most to industry practitioners when selecting CSOs for their existing or new projects. The results showed that a number of practitioners consider in-house software development as the sole CSO; however, there is a trend to consider additional CSOs. A large proportion of the practitioners included all four options in their CSO decision, i.e., internal software development, OSS, COTS and Outsourcing. In that particular case, the prioritization of the attributes is cost, longevity prediction, support of the component, whereas the rest of the attributes are ranked in a similar way among the practitioners, with API adequacy, size, and other as exceptions. Different attributes appear in a different order of importance for each case.
- RQ2: We wanted to explore if the decision process regarding the CSO was affected by the practitioners' characteristics. After a detailed analysis, based on the practitioners' inherent characteristics, it seems that smaller organizations and more immature products focus on properties associated with ease of use, development and maintenance of the component. On the other hand, bigger organizations and more mature products focus more on the properties associated with cost. Therefore, smaller companies need support in order to identify components that allow for easier use in terms of development. On the other side of the spectrum, bigger organizations with mature products need, and are looking for, less costly components.

Our research has several implications for research and practice. Firstly, we observed a wide variety of decision processes experienced by the survey respondents, even though we were not surprised given the fact that we had to deal with heterogeneous contexts. Moreover, our survey showed that decisions are based on data, and we tried to understand what types of data are used, how they are used, and how the data is translated into the actual decisions.

The data gathered in such studies is affected by various sources of variation and are therefore subject to large variability. The statistical analysis of such data can reveal significant differences, trends, disagreements, and groupings between the practitioners and can constitute a valuable aid for understanding the attitudes and opinions of the interviewed persons and therefore a tool for decision making.

In terms of future work, we intend to focus on providing support to companies in improving their component selection process. Within our work program, we plan to continue research and efforts towards efficient and effective decision making in component-based software engineering.

Funding information Open access funding provided by Örebro University. The work was partially supported by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Abts, C., Boehm, B. W., & Clark, E. B. (2000). COCOTS: a COTS software integration cost model-model overview and preliminary data findings. In proc. of the 11th ESCOM Conf, Munich, Germany (pp. 325–333).

Aitchison, J. (1982). The statistical analysis of compositional data (with discussion). *Journal of the Royal Statistical Society Series B, 44*, 139–177.

Aitchison, J. (2003). *The statistical analysis of compositional data*. London: The Blackburn Press.

Aitchison, J., & Greenacre, M. (2002). Biplots of compositional data. *Journal of the Royal Statistical Society: Series C: Applied Statistics, 51*(4), 375–392.

Aitchison, J. & Ng, K. W. (2005). Conditional compositional biplots: theory and application. In: 2nd compositional data analysis workshop CoDaWork'05, http://ima.udg.edu/Activitats/CoDaWork05/CD/Session1/Aitchison-Ng.pdf

Baars, A., & Jansen, S. (2012). A framework for software ecosystem governance. In: International conference of software business (pp. 168–180). Springer, Berlin.

Badampudi, D., Wohlin, C., & Petersen, K. (2016). Software component decision-making: In-house, OSS, COTS or outsourcing-a systematic literature review. *Journal of Systems and Software, 121*, 105–124.

Berander, P. & Wohlin, C. (2004). Difference in views between development roles in software process improvement—a quantitative comparison. In: Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004).

Borg, M., Chatzipetrou, P., Wnuk, K., Alégroth, E., Gorschek, T., Papatheocharous, E., Shah, S. M. A., & Axelsson, J. (2019). Selecting component sourcing options: a survey of software engineering's broader make-or-buy decisions. *Information and Software Technology, 112*, 18–34.

Bosch, J. (2018). The three layer product model: an alternative view on SPLs and variability. In: Proceedings of the 12th international workshop on variability modelling of software-intensive systems (pp. 1–1). ACM.

Brownsword, L., Oberndorf, T., & Sledge, C. A. (2000). Developing new processes for COTS-based systems. *IEEE Software, 17*(4), 48–55.

Chatzipetrou, P., Angelis, L., Rovegard, P., & Wohlin, C. (2010). Prioritization of issues and requirements by cumulative voting: A compositional data analysis framework. In: Software engineering and advanced applications (SEAA), 2010 36th EUROMICRO conference on (pp. 361–370). IEEE.

Chatzipetrou, P., Papatheocharous, E., Angelis, L., & Andreou, A. S. (2012). An investigation of software effort phase distribution using compositional data analysis. In: Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on (pp. 367–375). IEEE.

Chatzipetrou, P., Papatheocharous, E., Angelis, L., & Andreou, A. S. (2015). A multivariate statistical framework for the analysis of software effort phase distribution. *Information and Software Technology, 59*, 149–169.

Chatzipetrou, P., Alégroth, E., Papatheocharous, E., Borg, M., Gorschek, T., & Wnuk, K. (2018). Component selection in software engineering-which attributes are the most important in the decision process? In: Euromicro software engineering and advanced applications, SEAA, 2018.

Comas-Cufí, M., & Thió i Fernández de Henestrosa, S. (2011). CoDaPack 2.0: a stand-alone, multi-platform compositional software.

Cortellessa, V., Marinelli, F., & Potena, P. (2008). An optimization framework for "build-or-buy" decisions in software architecture. *Computers & Operations Research, 35*(10), 3090–3106.

Dunn, O. J. (1959). Estimation of the medians for dependent variables. *The Annals of Mathematical Statistics, 30*(1), 192–197.

Dunn, O. J. (1964). Multiple comparisons using rank sums. *Technometrics, 6*(3), 241–252.

Firesmith, D. (2004). Prioritizing requirements. *Journal of Object Technology, 3*(8), 35–47.

Gabriel, K. R. (1971). The biplot-graphic display of matrices with application to principal component analysis. *Biometrika, 58*(3), 453–467.

Gabriel, K. R. (1981). Biplot display of multivariate matrices for inspection of data and diagnosis. In V. Barnett (Ed.), *Interpreting multivariate data* (pp. 147–173). New York: Wiley.

Hatton, S. (2008). Choosing the "right" prioritisation method. In: 19th Australian Conference on Software Engineering.

Jorgensen, M., & Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering, 33*(1), 33–53.

Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association, 47*(260), 583–621.

Leffingwell, D., & Widrig, D. (2003). *Managing software requirements: a use case approach* (2nd ed.). Boston: Addison-Wesley.

Li, J., Bjørnson, F. O., Conradi, R., & Kampenes, V. B. (2006a). An empirical study of variations in COTS-based software development processes in the Norwegian IT industry. *Empirical Software Engineering, 11*(3), 433–461.

Li, J., Conradi, R., Slyngstad, O. P. N., Bunse, C., Torchiano, M., & Morisio, M. (2006b). An empirical study on decision making in off-the-shelf component-based development. In: Proceedings of the 28th international conference on Software engineering (pp. 897–900). ACM. Accessed Oct 2005.

Linåker, J., Munir, H., Wnuk, K., & Mols, C. E. (2018). Motivating the contributions: an open innovation perspective on what to share as open source software. *Journal of Systems and Software, 135*, 17–36.

Martín-Fernández, J. A., Barceló-Vidal, C. & Pawlowsky-Glahn, V. (2000). Zero replacement in compositional data sets. In: H. Kiers, J. Rasson, P. Groenen and M. Shader (eds.), Studies in classification, data analysis, and knowledge organization, proceedings of the 7th conference of the International Federation of Classification Societies (IFCS'2000), pp. 155–160. Berlin, Springer-Verlag.

Martín-Fernández, J. A., Barceló-Vidal, C., & Pawlowsky-Glahn, V. (2003a). Dealing with zeros and missing values in compositional data sets using non-parametric imputation. *Mathematical Geology, 35*(3), 253–278.

Martín-Fernández, J. A., Palarea-Albaladejo, J., & Gómez-García, J. (2003b) Markov chain Monte Carlo method applied to rounding zeros of compositional data: first approach. In: S. Thió-Henestrosa and J.A. Martín-Fernández (eds.), Proceedings of CODAWORK'03—Compositional Data Analysis Workshop, ISBN 84-8458-111-X. Girona.

Nguyen, V. (2010). Improved size and effort estimation models for software maintenance. In: 2010 IEEE international conference on software maintenance (ICSM) (pp. 1–2). IEEE.

Pearson, K. (1897). Mathematical contributions to the theory of evolution: on a form of spurious correlation which may arise when indices are used in the measurements of organs. *Proceedings of the Royal Society, 60*, 489–498.

Petersen, K., Badampudi, D., Shah, S., Wnuk, K., Gorschek, T., Papatheocharous, E., ... & Cicchetti, A. (2017). Choosing component origins for software intensive systems: in-house, COTS, OSS or outsourcing?—a case survey. IEEE Transactions on Software Engineering.

Regnell, B., Host, M., Natt och Dag, J., Beremark, P., & Hjelm, T. (2001). An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering, 6*, 51–62.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*(2), 131–164.

Staron, M., & Wohlin, C. (2006). An industrial case study on the choice between language customization mechanisms. In: J. Münch, and M. Vierimaa (eds.) PROFES 2006, LNCS 4034, pp. 177–191. Springer-Verlag, Berlin.

Sulaman, S. M., Orucevic-Alagic, A., Borg, M., Wnuk, K., Host, M., & de la Vara, J. L. (2014). Development of safety-critical software systems using open source software—a systematic map. In: 2014 40th EUROMICRO conference on software engineering and advanced applications (SEAA) (pp. 17–24). IEEE.

Wohlin, C., Wnuk, K., Smite, D., Franke, U., Badampudi, D., & Cicchetti, A. (2016). Supporting strategic decision-making for selection of software assets. In *International conference of software business* (pp. 1–15). Cham: Springer.

Dr. 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 received her BSc degree in "Informatics," MSc in "Informatics and Business Administration," and Ph.D. in "Informatics" from the Department of Informatics, Aristotle University of Thessaloniki (AUTh), Greece. Her doctoral dissertation has the title: "Statistical methods in information systems project planning." She holds a master in pedagogy and didactics, and in parallel, she has been educated in special education, learning difficulties, and dyslexia. As a researcher, she mainly focuses on empirical studies under the different perspectives of software development. Her research interests include—but are not limited to—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 behavioral software engineering.

Dr. Efi Papatheocharous is a senior researcher at the Research Institutes of Sweden, RISE (RISE/ICT). Prior to this, she was a Post-doctoral Research Fellow at the Swedish Institute of Computer Science (SICS) and a Lecturer at the Department of Computer Science, University of Cyprus. She obtained her Ph.D. in Computer Science from the University of Cyprus, in 2012, her M.Sc. at the University of Manchester in Advanced Computer Science with ICT Management, in 2005, and her B.Sc. in Computer Science from the Department of Computer Science, University of Cyprus, in 2004. Since 2006, she has been collaborating in a number of research projects at national and European levels and contributing to the research community with many papers published in peer reviewed books, international conference proceedings journals. Her research is in software engineering, focusing on improving developer productivity, architectural decision-making, and software quality. Some of her research interests include software cost estimation, system architectures, agile methodologies, software ecosystems, human factors in software engineering, and systems-of-systems.

Krzysztof Wnuk is an associate professor at the Software Engineering Research Group (SERL) of Blekinge Institute of Technology, Sweden. He received his PhD from Lund University, Sweden (2012). Krzystof Wnuk's 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.

Markus Borg is a Senior Researcher with the Software and Systems Engineering Laboratory, RISE Research Institutes of Sweden AB, Sweden 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.

Dr. Emil Alégroth is a Software Engineering researcher at Blekinge Institute of Technology. He has several years of experience with industry–academia research collaboration as well as empirical experience as the Chief Executive Officer (CEO) of a spin-off company from his PhD studies. His research interests are primarily automated GUI-based software testing and also decision-making in software engineering, human factors, and technical debt.

Tony Gorschek is a Professor of Software Engineering at Blekinge Institute of Technology—where he works as a research scientist in close collaboration with industrial partners. Dr. Gorschek has over 15 years of 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™ and Lean product development and evolution. www.gorschek.com, tony.gorschek@bth.se

Affiliations

Panagiota Chatzipetrou[1] · Efi Papatheocharous[2] · Krzysztof Wnuk[3] · Markus Borg[2] · Emil Alégroth[3] · Tony Gorschek[3]

Efi Papatheocharous
efi.papatheocharous@ri.se

Krzysztof Wnuk
krzyszstof.wnuk@bth.se

Markus Borg
markus.borg@ri.se

Emil Alégroth
emil.alegroth@bth.se

Tony Gorschek
tony.gorschek@bth.se

[1] Department of Informatics, CERIS, Örebro University School of Business, SE-701 82 Örebro, Sweden

[2] RISE Research Institutes of Sweden AB, Stockholm, Sweden

[3] Software Engineering Research Lab (SERL), Blekinge Institute of Technology, Karlskrona, Sweden