Software Component Decision-making: In-house, OSS, COTS or Outsourcing - A Systematic Literature Review

Deepika Badampudi[a], Claes Wohlin[a], Kai Petersen[*,a]

[a]*Blekinge Institute of Technology, Karlskrona, Sweden*

Abstract

Context: Component-based software systems require decisions on component origins for acquiring components. A component origin is an alternative of where to get a component from.

Objective: To identify factors that could influence the decision to choose among different component origins and solutions for decision-making (For example, optimization) in the literature.

Method: A systematic review study of peer-reviewed literature has been conducted.

Results: In total we included 24 primary studies. The component origins compared were mainly focused on in-house vs. COTS and COTS vs. OSS. We identified 11 factors affecting or influencing the decision to select a component origin. When component origins were compared, there was little evidence on the relative (either positive or negative) effect of a component origin on the factor. Most of the solutions were proposed for in-house vs. COTS selection and time, cost and reliability were the most considered factors in the solutions. Optimization models were the most commonly proposed technique used in the solutions.

Conclusion: The topic of choosing component origins is a green field for research, and in great need of empirical comparisons between the component origins, as well of how to decide between different combinations of them.

Key words:
OSS, COTS, In-house development, Outsourcing, Decision-making, Component-based software engineering

1. Introduction

Component-based software development usually involves assembling and developing reusable components [32]. In this study a component is referred to as an asset, which when incorporated into a system adds value (functional and/or non-functional) to the end customer. Hence assembling or developing operating systems components or databases components are out of the scope of this study. Each component can be obtained from a different source, we refer to the sources as component origins in this study.

Four of the widely used component origins are considered in this study as follows:

1. In-house developed components: Components are developed within the company.
2. Components off-the-shelf (COTS): Commercial components are externally bought from another company/vendor.
3. Open Source Software (OSS): Open source components are externally obtained from the OSS community.
4. Outsourced components: The development of components is outsourced (or subcontracted).

These component origins have been chosen as they are well-defined options related to developing with components (see for example [2], [11] and [12]).

Depending on the goal, the decision-maker might decide to use one of the component origins. For example, if reducing the overall cost of the project is the goal, the decision-maker might select a component origin which is cost effective. However, this decision is not easy due to the uncertainties involved in software projects. In addition, there is often more than one goal that needs to be considered in such decisions. In order to make an informed decision, the decision-maker should consider all the factors and their trade-offs that could influence a decision to select a component origin. Factors are project attributes (for example, time and cost), project activities/processes (for example, maintenance and integration) and non-technical factors (for example, technical support and source code availability). The objective of this paper is to identify the factors that influence the decision to select a component origin and the solutions on how to make a choice based on the factors (for example, through an optimization model).

The strategic decision of where to get components from is a complex one, and has major implications that may not be overseen immediately. For example, on the decision level to choose component origin we may begin with a make-or-buy decision, which is to build in-house or to get components externally. Then when deciding to get components externally, additional implications may play into the decision. For example,

*Corresponding author
Email addresses: deepika.badampudi@bth.se (Deepika Badampudi), claes.wohlin@bth.se (Claes Wohlin), kai.petersen@bth.se (Kai Petersen)
URL: http://www.bth.se/ (Deepika Badampudi)

when outsourcing development to another company located, for example, in a different country, as we are customers, we can still influence the development processes, and have control over the evolution of components. When deciding to get COTS components, then we have less influence, but not the challenge of transferring knowledge, learning, etc. that may occur in outsourcing. This indicates that, from a long-term strategic perspective the decision of where to get components from may have important implications, on for example, effort. Practitioners can benefit from knowing the factors that could influence the decision to choose among different component origins and possible solutions that support decision-making process.

The individual component origins (COTS, OSS, in-house developed components and outsourced components) have already been well explored in a number of existing literature reviews. Hauge et al. [30], for example, conducted a systematic literature review on OSS, and Maras et al. [31] conducted a review on developing software with components such as COTS. The selection of actual components may take place on multiple decision levels, namely choosing the component origin, choosing the provider, and choosing the actual component. Secondary studies on component provider selection [3] and component selection [21] exist. No systematic literature review addressing decision-making with regard to component origins was identified, highlighting the need for such a review of the literature. Hence, the inclusion and exclusion criteria focus on the component origin, the decisions on different levels are not independent as is shown in the Figure 1. As there is no synthesis on the component origin decision level, this study focuses on component origin selection. The decisions on how to select and integrate component (after the component origin is chosen) are not considered within the scope of this study.

Systematic review studies are a means to aggregate evidence through a scientific and repeatable process. A systematic literature review was used as the research method to find the factors and solutions in the literature using the guidelines in [13] and [18]. The guidelines in [34], [33], [19] and [15] are used for quality assessment, data analysis, classification of research types and validation of the study respectively. Systematic review studies are a means to aggregate evidence through a scientific and repeatable process. The specific contributions achieved through the systematic review are the identification of:

- C1: the research types and methods used to evaluate the factors and solutions.
- C2: the factors considered in comparing two or more component origins in a way that facilitates decision-making.
- C3: the solutions to decision-making (for example, optimization).
- C4: the research gaps and future directions.

The remainder of this paper is structured as follows: Section 2 presents related work on existing literature reviews closely related to our study. Section 3 presents the research method. The results are presented in Section 4 and discussed in Section 5. Section 6 concludes the paper.

2. Related Work

Three different levels of decisions are recognized as shown in Figure 1. The light grey block represents the decision level. The white block represents the decision space where the different options (dark gray) are traded-off. The arrows represent the decision control flow. As seen in Figure 1 the decision is not taken on the same level, the control shifts across different levels. For example, once the component origin is chosen, the provider is selected followed by the actual component selection. However, while evaluating components, the decision-maker might want to consider alternative provider or even an alternative component origin. This indicates that once a decision is taken at a decision level, it can be revised by the activities in the next decision level. The decision on each decision level is a research study in itself.

The focus is delimited to decision-making choosing between the four component origins: OSS, COTS, developing components in-house, and outsourcing the development of components.

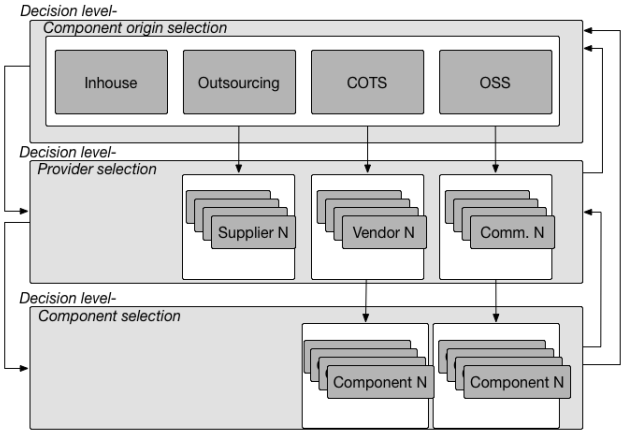

Figure 1: Decision levels

Secondary studies on decision level for provider [3] and component selection [21] have been conducted. No systematic literature review on the topic of component origin selection while using a systematic approach for study identification (snowball sampling [17] and database search [16]) was identified. Primary studies to select component origins exist, although the research problem to select component origin is not among the frequently researched topics [20]. In the study [20] different research topics on component-based software engineering from its inception were reviewed. The results in [20] indicate that none of the most researched topics is based on decision-making to select the component origin. However, industrial research partners have corroborated that it is an area where improvements can be made. Based on the industry interest and the limited number of studies in the area, we conclude that it is an area worthy of further research.

Secondary and primary studies have considered the decision to select the provider and to select components after the component origin is chosen. Examples of secondary and primary

studies outside our scope, but still related to decision-making are listed in Tables 1 and 2. The secondary and primary studies related to the provider and component selection decision levels are presented in Table 1. Primary studies in Table 1 are a subset of existing studies related to the provider and component selection. While, Table 2 provides a list of secondary studies related to the different topics with respect to the individual component origins. The primary studies that compare the component origins are included in the result of the review study and are mentioned in the result section.

Table 1: Secondary and primary studies related to the provider and component selection decision levels

Decision Level	Secondary studies	Primary studies
Provider selection	Supplier selection [3]	Vendor selection [4], community selection [5]
Component selection	OSS component selection [21]	COTS selection [6, 7, 8], OSS selection [9, 10]

Table 2: Systematic reviews related to individual component origins

Component origins	Secondary studies
Open source	Software evolution [22, 29], software quality (reliability, interoperability) [23], development processes [24], software quality (maintainability) [25], OSS in commercial development [26, 30], challenges on integrating open and inner source [27], empirical methods used in OSS research [28]
COTS	–
In-house	Synthesis of research on component based software engineering [31]
Outsourcing	–

As seen from Table 2 no secondary studies on COTS and outsourcing were conducted. Using the selection of relevant keywords of each comparison origin in Scopus, in combination with the research method of systematic literature review and systematic mapping studies, COTS and outsourcing of components did not yield any results. As the focus of this study is on decision-making in choosing the component origin, no deeper analysis of the studies is provided.

3. Method

3.1. Need for the review

While we could not find any existing review on decision-making to choose a component origin, practitioners would greatly benefit from being aware of possible solutions and relevant factors for choosing between the component origins. This review is conducted to explore the factors that are considered in decisions to choose a component origin and also to understand the existing solutions available to guide the decision-making process. The following research questions were used to drive the review study:

- *RQ1: What are the research types, methods and quality of the contributions?* Through this research question, the identified papers are classified according to the six types of research defined by Wieringa [19]: evaluation research, validation research, solution proposal, philosophical papers, opinion papers, and experience reports. In addition, the research methods, rigor and relevance used by the primary studies to obtain the results are discussed (see contribution C1). The answer to RQ1 research question allows determining the strength of evidence of the existing literature.
- *RQ2: What are the different factors that influence the decision to choose among different component origins?* Four component origins are of particular interest in this study. In order to choose between component origins, comparisons may be carried out. The goal of this research question is to identify the different factors that are compared to evaluate the different component origins. The list of factors considered in the decision to choose a component origin and the evidence supporting the results can be used by decision-makers in their decisions, in particular they provide a list of potential component origins to consider when making such a decision (see contributions C2).
- *RQ3: What solutions have been proposed to choose the component origin?* With the influencing factor as input, different solutions might be proposed to make a decision. For example, prioritization based approaches making trade-offs, or models that optimize with regard to a specific criterion under given constraints. The available solutions provide an inventory of solutions to help practitioners choose a solution for themselves when making a trade-off among component origins. The limitation of the solutions will point the direction of future work in proposing new solutions (see contributions C3 and C4).

3.2. Study identification

For study identification, two approaches were used. First, the first and second author collaborated on the identification of studies through snowball sampling. Then the third author, independently of the other authors, conducted a database search to complement the snowball sampling. The research on decision-making to choose a component origin is relatively new. In addition, the terminology is not yet established or consistently used in the research papers. Hence, snowball sampling is chosen as the main method instead of database search. That is, the snowball sampling is conducted first and then database search is conducted to validate the snowballing process and to ensure the coverage of the primary studies. As suggested in [13] all papers for a search could be identified only using snowball search, though to add confidence to the study identification process, also a database search was conducted. We evaluated the preliminary search results from snowballing sampling and database search [14] and conclude that the implemented search strategy was efficient and reliable. However, it was noted that the primary studies that were identified were not from conferences related to component-based software engineering. To ensure that this is not a threat to validity, a manual search of the proceedings for the component-based software engineering conference

(CBSE) between 2004-2015 was conducted. No new papers were found in the manual search.

3.2.1. Inclusion and exclusion criteria

The inclusion and exclusion criteria have been used to focus this study on the component origin selection. The same inclusion and exclusion criteria have been used for all searches, i.e. start set in snowballing, backward snowballing, forward snowballing, and the database search. These are explained in the following sections.

In order to include a paper, at least one of the following inclusion criteria should be fulfilled:

- Papers discussing the decision-making process for selecting a component origin, i.e. answer the question: Why select the component origin? Not how to select a component once the component origin is decided.
- Papers comparing two or more component origins, for example, papers discussing cost, time, scope or quality trade-offs (or proxies of these four aspects) for two or more component origins are of particular interest.
- Papers proposing solutions that support the decision-making process in the selection of the component origin in relation to other component origins, i.e. answer the question: How is it possible to make the decision process for selecting a component origin more effective and efficient?
- All papers should be peer-reviewed.

Papers are excluded if they fulfill any of the following exclusion criteria:

- Papers that are not related to component-based software systems.
- Papers discussing architectural aspects of component-based software systems, such as papers answering questions: How to integrate the components in the system once the component origin is decided?
- Papers discussing development solutions (Programming) for component-based software systems.
- Papers discussing adoption of platforms, technology, IT services or operating systems.
- Papers discussing end-user adoption of software packages.
- Papers that discuss only one component origin without contributing to the decision-making process on why a specific component origin should be used or not.
- Duplicate reports of the same study.
- Grey literature.
- Articles not written in English.

3.2.2. Snowball sampling

The snowball search consisted of the following steps. First, a start set of papers was established through a search in Google Scholar and the application of the inclusion and exclusion criteria. Thereafter, backward snowballing (based on reference lists) and forward snowballing (based on citations) were conducted. The snowball procedure was conducted jointly by the first two authors of the paper.

Establish a start set: Since we are interested in looking at how the different component origins are compared and how the decisions are made, the following search strings were generated:

- In-house vs. outsourcing and software
- In-house vs. COTS
- In-house vs. OSS
- COTS vs. OSS
- COTS vs. outsourcing and software
- Outsourcing vs. OSS and software

Papers were retrieved from Google Scholar as an unbiased start set was wanted, i.e. not papers available in a specific publisher's database. A cutoff was chosen to get a start set. The first 10 entries of each search string were considered. Among the 60 entries found, there were many entries that were not software related, particularly for the following search strings: In-house vs. outsourcing, Outsourcing vs. OSS and COTS vs. Outsourcing. The common keyword in the above search strings was "outsourcing". As outsourcing is done in various domains, "software" was added to the search string to restrict the papers to the software domain.

In total, the search resulted in 90 entries for which the inclusion and exclusion criteria were applied. The inclusion and exclusion of papers for the start set was carried out in two phases. The objective was to find at least one paper for each search string.

Start set - Phase 1: Both authors reviewed the meta-data (title, abstract and introduction) independently for all 90 entries. The decision point was noted while reviewing the papers, where decision point refers to the point at which the decision of inclusion or exclusion was made, in this case, title, abstract or introduction. The paper was either tentatively accepted for Phase 2 or rejected based on the inclusion/exclusion criteria. Both authors performed the inclusion and exclusion independently and once the authors completed this phase a review meeting was held to compare the review results. The decision rules shown in Table 3 are defined for inclusion and exclusion according to the outcome of the review process conducted by the first two authors. An inclusive approach has been adopted, i.e. when in doubt the paper has been included for further evaluation in Phase 2.

Table 3: Inclusion and exclusion decision rules

Case	Action
Both authors accept a paper	Include the paper for next step
Both authors reject a paper	Exclude the paper
Either one of the authors accepts a paper	Include the paper for next step

Start set - Phase 2: Often the title, abstract, and introduction are insufficient to get a full and comprehensive picture of

the articles, and hence a final decision was reached by further reading the papers. The papers were reviewed and a summary of the papers was reported along with the final decision made. The decision was either to include the paper in the start set or exclude it. This process was conducted by the first author and the second author reviewed the report and either agreed or disagreed to the decisions made. At the end of this step, the papers were finally included in the start set when both authors agreed to include the paper. If one of the authors disagreed for a particular paper, the paper was discussed until the authors reached a consensus. After completing Phase 2, a total of five papers were included, which were used to conduct backward as well as forward snowballing (cf. [13]).

Backward snowballing: Backward snowballing is the process of reviewing the references of a relevant paper. Backward snowballing started by randomly selecting a paper from the start set. The process and order of reviewing the references was as follows:

1. Title of the referenced paper.
2. The reference context of the paper. The reference context refers to the text surrounding the reference citation within the primary study.
3. Abstract of the referenced paper.

Phases 1 and 2 were repeated on the references of the start set papers and decision points (title, abstract or full text) were noted. The first two authors performed this step independently. At the end of the reviewing process, a second review meeting was conducted to compare the results. Again the same steps were followed as in Table 3 and the same inclusion and exclusion criteria were applied.

Forward snowballing: Forward snowballing is the process of reviewing the citations of a relevant paper. The starting point of forward snowballing was the randomly selected paper from the start set. For each paper citing the paper in the start set, the following was reviewed:

1. Title of the paper citing.
2. Abstract of the paper citing.
3. The reference context to the paper being cited.

Phases 1 and 2 were conducted for the paper citing the papers in the start set and decision points (title, abstract or full text) were also noted. The first two authors performed this step independently. And at the end of this process, a third review meeting was conducted to compare the results. Again the same steps were followed as in Table 3. Backward and forward snowballing were performed for all papers in the start set. The papers accepted in backward and forward snowballing were added to the final set of papers to be included in the review. Backward and forward snowballing were repeated for each newly added paper in the final set of papers until no new papers were found.

Initially, the backward and forward snowballing was done independently by both first and second author. As the reviewing continued the decisions (accept/reject) made by the authors were the same, i.e. the authors accepted/rejected the same papers. From there onwards only the first author conducted the snowballing and the second author reviewed the snowballing process reported by the first author.

Figure 2 shows the number of papers retrieved at each stage through the snowball sampling process.

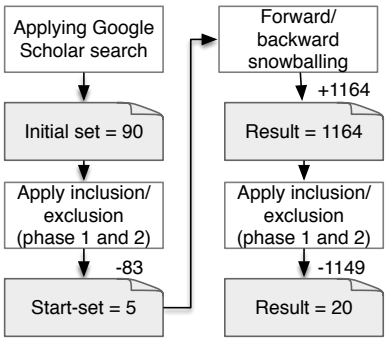

Figure 2: Snowball search process

3.2.3. Database search

The main purpose of the database search was to increase the confidence in the selection of papers. Thus, the database search is conducted as a complementary search to the snowballing procedure.

The search terms were divided into population, intervention, comparison, and outcome (PICO). The population was restricted by searching for *"software (X)"* and *"component (Y)"*.

The intervention relates to making trade-offs, decisions, or selecting a component origin, search keywords are: *"trade-off (I1)"*, *"decision (I2)"*, and *"selection (I3)"*
Outcome for the following component origins:

- A: (COTS OR "components off the shelf" OR "component off the shelf")
- B: (In-house OR inhouse)
- C: (open-source OR "open source" OR OSS)
- D: (outsource OR out-source OR outsourcing OR "third-party")

Comparison relates to the different component origin combinations. Given that different combinations of component origins are of interest, the following strings are constructed:

- E: A AND (B OR C OR D)
- F: B AND (C OR D)
- G: C AND D

This resulted in the following search strings:

- *Search 1:* (X OR Y) AND (I1 OR I2 OR I3) AND E
- *Search 2:* (X OR Y) AND (I1 OR I2 OR I3) AND F
- *Search 3:* (X OR Y) AND (I1 OR I2 OR I3) AND G

The databases searched were Scopus and Inspec/Compendex, i.e. three index databases were used

that encompass publications from Elsevier, IEEE, ACM, etc. The number of search results per search string and database is shown in Table 4 and Figure 3 depicts the database search process.

Table 4: Number of hits for database search

Database	Search	Hits
Scopus	Search 1	60
Scopus	Search 2	101
Scopus	Search 3	51
Inspec/Compendex	Search 1	71
Inspec/Compendex	Search 2	110
Inspec/Compendex	Search 3	38

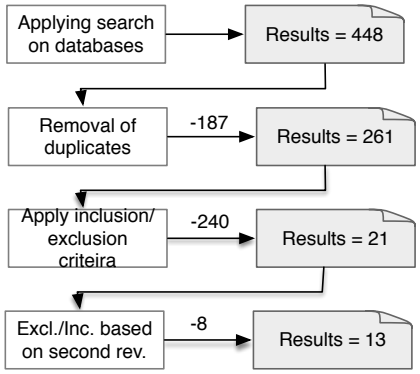

Figure 3: Database search process

Applying the inclusion and exclusion criteria: The inclusion and exclusion criteria were applied by the third author independently. The results of the database search were reviewed by the first author. In cases of disagreements, the authors discussed the papers until the disagreements were resolved.

3.2.4. Combined results

In total (snowballing + database search), 24 unique papers were identified. The number of included papers unique to snowballing, database search as well as the papers found by both approaches are shown in Table 5.

Table 5: Search type and number of relevant papers

Search type	Number of papers
Snowball sampling	Number of papers 11
Database search	Number of papers 4
Both	Number of papers 9

3.3. Data extraction and classification

A data extraction form was designed to extract the data in an explicit and consistent way. Which data to extract was driven by the research questions. Table 6 provides an overview of the data extracted, and how the data fields link to the research questions.

The research type was classified according to Wieringa et al. [19]; Six types (evaluation research, validation research, solution proposal, philosophical papers, opinion papers, and experience reports) were distinguished. Only evaluation and validation research are of empirical nature, where evaluation research is rooted in practice, and validation research is conducted in laboratory settings. Hence, the classification shows the strength of evidence by illustrating the ratio between the research types.

Table 6: Data extraction form

Information	Description	RQs
Meta-information	Paper ID	–
Study focus and methods		
Research facet	Classification of research type based on Wieringa et al. [19]	RQ1
Topic specific information		
Factors	Factors considered for decision-making and comparison (emerging)	RQ2
Solution	Description of the solution to make a decision (emerging)	RQ3

The first author designed the extraction form, and it was reviewed by both the second and third author prior to extraction. In addition, as mentioned in Section 3.2 full text of the papers was read, this gave an overview of the depth and breadth of the available data. Thereafter, the first and third authors divided the papers and extracted the information from the primary studies. The first author extracted data from 18 papers, and the third author extracted data from 6 papers. The extraction was done by identifying the text segments that were related to the research questions. In total 96 text segments were extracted. The text of each text segment was recorded in data extraction form. In most cases, the exact text was recorded with the exception of paraphrasing the text that contained other irrelevant information.

3.4. Quality assessment

The quality of the empirical studies is assessed by evaluating the rigor and relevance. Rigor and relevance are considered as the two main criteria for evaluating the study quality [34]. The guidelines proposed by Ivarsson and Gorschek [34] have been used to score the rigor and relevance of empirical studies. The details of the rubrics used to evaluate are presented in Sections 3.4.1 and 3.4.2. It is to be noted that not all primary studies included in this study are empirical, hence the quality of only a subset of primary studies is evaluated.

3.4.1. Rigor

The rigor is assessed by determining the degree to which the study context (C) validity threats (V), and research design (R) are described in the primary studies. The rigor aspects are rated as strong (1), medium (0.5) or weak (0).

Strong description (1): The rigor aspect is rated as strong when the following conditions are fulfilled:

1. *Context (C)* - C equals to 1 if the context describes sufficient details of context such as the size of the company/organization, the profile, size, and domain of the projects investigated.
2. *Validity threats (V)* - C equals to 1 if all the applicable validity treats are considered, and appropriate actions are taken to mitigate the threats.
3. *Research design (RD)* - The research design is rated as 1 when the research methodology is executed following the best practices. In addition, it should report details such

as data collection, sampling, analysis, treatments, and outcome variables.

Medium description (0.5): The rigor aspect is rated as medium (0.5) if any one of the descriptions mentioned as strong description is missing.
Weak description (0): The rigor aspect is rated as low (0) if no descriptions that are mentioned as strong descriptions are provided.

3.4.2. Relevance

Relevance evaluation consists of evaluating the realism of the environment in which the results are obtained. Evaluating the subjects/users (S), context (C) and scale (S) determines the relevance. In addition, the suitability of the research method (R) used to obtain the results is evaluated. The relevance aspects are rated as either high (1) or low (0).

1. *Subjects/users (SU)* - **Contribute to relevance (1):** S equates to high (1) when the subjects/users used in the evaluation are representative of the intended users of the process, i.e. industrial practitioners.
   Do not contribute to relevance (0): S equates to low (0) when the subjects/users used in the evaluation are not representative of the intended users of the process, i.e. students, researchers or subjects are not mentioned.
2. *Context (C)* **Contribute to relevance (1):** C equates to high (1) when the context in which the results are obtained is representative of the intended usage context, i.e. industrial context/setting.
   Do not contribute to relevance (0): C equates to low (0) when the context in which the results are obtained is not representative of the intended usage context, i.e. laboratory setting or other simulations.
3. *Scale (S)* **Contribute to relevance (1):** S equates to high (1) when the scale of application used in evaluation is of realistic size, i.e. projects investigated are of an industrial scale or the number of interviews in the case study, or number respondents of the survey are of realistic size.
   Do not contribute to relevance (0): S equates to low (0) when the evaluation is performed using applications of unrealistic size, i.e. down-scaled or toy example.
4. *Research Method (RM*) **Contribute to relevance (1)**: RM equates to high (1) when the research method used allows to investigate real situations, and that is relevant for practitioners, i.e. industrial survey or case study.
   Do not contribute to relevance (0): RM equates to low (0) when the research method used does not lead itself to investigate real situations or if no research method is used, i.e. laboratory experiment (software or human subjects).

The results of the quality assessment of the empirical papers are provided in Section 4.1.3.

3.5. Analysis

Thematic analysis is used for analyzing the data extracted from the primary studies. The thematic analysis identifies, analyses and reports patterns (themes) within data [33]. After the text segments are extracted, they are coded into themes by the first and the third author independently using the coding rules defined in Table 7. Some text segments can be assigned to more than one theme, for example, if the text compares the integration effort for COTS and OSS and also discusses integration effort as an influencing, the text segment is coded into the theme called "effort" and "integration". The themes that include only one text segment are not considered since the analysis aims to identify recurring themes.

Table 7: Rules defining the themes

Themes	Rules	Example of Text Segment
Time	Description of the time in terms of the duration	"Time to market was the common project profiles which was most emphasized" [35]
Cost	Any costs in terms of the money involved in a project	"Relying on OSS helped the project to overcome budget related issues" [45]
Effort	Estimation of effort in terms of man-hours to perform an activity	"Unsatisfactory integration effort estimation is a common risk of OTS components" [35]
Quality	Discussing quality attributes such as reliability and security	"Consumers make product quality inferences based on cues such as brand and store name through a process called affect-referral." [42]
Market Trend	The ability of the component to evolve based on the needs of the market place	"Market rules force COTS vendors to release a new version of their products even if they know they are not fully working." [36]
Source Code Availability	Text describing the availability of source code	"Open access to source code allows greater opportunities for customizing OSS software to the specific needs" [50]
Technical Support	Discussions on the support provided or available for fixing bugs and required changes in functionality	"As soon as the attack the OSS community fixes or countermeasures are devised quickly" [44]
License	Text describing the license agreement, obligations or restrictions	"License requires that modified versions of the software also be open (often referred to as a copyleft provision)" [42]
Integration	Text describing the integration activity	"Required integration work is used in evaluation criteria" [38]
Requirement	Discussion on requirements as a phase and nature of requirements (complex or unique)	"Recommended changes to a customer's requirement when we felt that by doing so we could satisfy it with an OSS component." [45]
Maintenance	Related to upgrading the system and maintaining system stability	"The OSS user is not obliged to upgrade." [36]

The coding rules for assigning text segments to the themes were commonly defined by the first and the third author. Such rules reduce the threat of bias and mistakes during the data extraction. The text segments from the primary studies are summarized and coded into existing themes, new themes are added and defined whenever necessary in an inductive manner. The text segments coded into the themes is compared within and across the themes to compare and contrast the properties of dif-

ferent themes. In order to keep the traceability between the data and the primary studies, the text relevant for the extraction (text segments) within primary studies is highlighted, and tagged by a comment linking it to the themes using Adobe reader. This helps in checking the reliability and revising the extraction if needed.

In total 11 themes were identified which were the factors that could influence the decision to choose a component origin.

3.6. Validity threats

Petersen and Gencel [15] proposed to discuss five types of validity threats, namely descriptive validity, theoretical validity, generalizability, interpretive validity, and reliability in software engineering research in general.

3.6.1. Descriptive validity

Descriptive validity concerns threats to the ability to accurately capture and describe observations made.

This threat is mainly relevant in the data extraction activity. To reduce the threat, a data extraction form was designed, which was then reviewed by two experienced researchers having conducted a number of systematic literature reviews (authors two and three). Furthermore, it helped to keep the traceability between the original text of the papers and the data extracted by highlighting and tagging the text within the document. That way it was always possible to return to the source when in doubt. Hence, this threat is well controlled.

3.6.2. Theoretical validity

Theoretical validity threats may lead to a lack of ability to capture what is intended to be captured, for example, due to confounding factors that the researcher may not be aware of.

Study identification and sampling: In this study, different search strategies were utilized, i.e. snowball sampling and database search to increase the reliability of the study identification. Furthermore, multiple researchers have been involved in the selection of studies. As the complementary database search only yielded a small set of additional papers, the confidence in the sample is increased. The database search added papers in the component origins comparison category for which there was no paper found through the snowballing process.

Researcher bias: In the data extraction as well as study selection researcher bias is a threat. To reduce bias, more than one researcher was involved in each step. The snowball sampling procedure was conducted by two authors. The result of the database search was checked by the first author. In addition, during the data extraction when in doubt both researchers were in the same room during the extraction to immediately discuss the doubt. Hence, the threat due to the extraction not being reviewed is reduced.

3.6.3. Generalizability

The generalizability is concerned with generalizing within groups (i.e. the same community or organization) and between groups (i.e. between different communities and groups).

Internal generalizability: The ability to generalize within the group (decision-making for choosing a component origin) is highly dependent on the studies published (for example, the different contexts studied). Hence, internal validity is limited to the research contexts of the studies. In this case, limited information is available on the context reported, hence it is impossible to make a judgment regarding this. Furthermore, the total number of primary studies is low. Per category (comparison of component origins), only a small number of papers was identified. In particular, in-house vs. OSS and in-house vs. outsourcing are only represented by two papers each.

External generalizability: To determine whether the same approaches are also applicable when having a different component origin to choose from is a subject for further investigation and cannot be determined here.

3.6.4. Interpretive validity

Interpretive validity is concerned with threats related to the ability to draw objective and valid conclusions based on the data collected, i.e. this threat is relevant when interpreting the findings. Coding rules were defined and regular meetings were conducted during the data analysis to ensure that the data is interpreted in the same way by all authors. Hence, the threat was mitigated by involving multiple researchers in the interpretation and discussion of the findings.

3.6.5. Reliability

In order to achieve reliability, the research steps must be repeatable. Repeatability is the ability of other researchers replicating the results. The detailed steps taken in the searches are reported. For example, the search strings used, databases used and inclusion/exclusion criteria are documented, hence reliability is increased. Repeatability of the data extraction is important. The relevant text segments were highlighted and tagged with appropriate codes within the document. This facilitates traceability and repeatability of the data extraction process.

4. Results

The answers to the research questions are provided in this section. The contribution of the primary studies with respect to the comparison categories is shown in Table 8.

Table 8: Total number of Primary studies per comparison category

Component origin	Count	References
OI-OC	8	[39, 40, 43, 48, 49, 51, 53, 54]
OC-OS	6	[35, 36, 37, 41, 44, 50]
OS	5	[42, 46, 47, 52, 57]
OI-OS	1	[58]
OI-OO	2	[55, 56]
OI-OC-OS	2	[38, 45]

OI-OC: In-house vs. COTS, **OC-OS:** COTS vs. OSS, **OS:** Only OSS , **OI-OS:** In-house vs. OSS, **OI-OO:** In-house vs. outsource, **OI-OC-OS:** In-house vs. COTS vs OSS.

We assigned identifiers to the component origins as follows: OI - In-house development, OC - COTS, OS - OSS and OO -

Outsourcing, where O stands for origin. Hence, Ox-Oy represents the decision between component origins.

In the snowball sampling and database search, we found papers comparing or discussing influencing factors for all of the component origin comparisons listed above. In addition, new comparisons (OS and OI-OC-OS) also emerged where studies discussed more than two component origins and some papers discussed the factors influencing the adoption of a specific component origin. It is important to note that no primary studies discussing COTS vs. Outsourcing and OSS vs. Outsourcing were found, whether or not such discussion are considered in practice is a topic for further research.

4.1. Research types, methods and quality (RQ1)

In this section we discuss the research types, the research methods and the quality of the primary studies. Section 4.1.1 describes the research types, Section 4.1.2 describes the research methods and Section 4.1.3 describes the results of quality assessment of the primary studies.

4.1.1. Research types

Identifying the research types gives a good overview on the strength of evidence of the primary studies. Table 9 provides an overview of the research types of the primary studies based on Wieringa et al.'s classification [19]. Evaluation research is an investigation of a problem in practice or implementation of a technique in practice. Solution papers are papers proposing solutions and arguing its relevance. Philosophical papers are papers that sketch a new way of looking at things. Validation research investigates the properties of a solution proposal that has not been implemented in practice and opinion papers are papers that contain opinions of the authors. The most common type is "Evaluation research", which indicates that the data collected is originating from the software engineering practice and is empirical. Validation research also represents empirical studies, indicating that 9 out of 24 studies are empirical. In addition, 8 out of 24 papers are neither empirical studies nor solutions papers.

Table 9: Research types

Research types	Count	References
Evaluation research	8	[35, 37, 38, 40, 42, 43, 47, 58]
Solution proposal	7	[39, 48, 49, 51, 53, 54, 56]
Philosophical papers	4	[44, 46, 50, 52]
Validation research	1	[55]
Opinion papers	2	[36, 41]
Experience papers	2	[45, 57]

Table 10 shows how the primary studies are distributed in relation to the comparison of component origins. Even though the highest number of studies is classified as evaluation research, the evaluation research studies are distributed among "In-house vs. COTS" to " In-house vs. COTS vs. OSS", which means that in each category only a few studies are rooted in practice. Of the solutions proposed for "In-house vs. COTS", only two are evaluated in practice [49] and [54].

It is also interesting that, even though contributing with fewer studies, all research types proposed by Wieringa et al. [19] could be identified in the set of primary studies.

Table 10: Research types and studied combinations of component origins

Research type	OI-OC	OC-OS	OS	OI-OS	OI-OO	OI-OC-OS
Evaluation research	2	2	2	1		1
Solution proposal	6				1	
Philosophical papers		2	2			
Valdation research					1	
Opinion papers		2				
Experience papers			1			1
OI-OC: In-house vs. COTS, OI-OS: Inhouse vs. OSS, OC-OS: COTS vs. OSS						
OI-OO: In.house vs. Outsourcing, OS: Only OSS						
OI-OC-OS: In-house vs. COTS vs OSS						

4.1.2. Research methods

Research methods for the evaluation and validation research studies (see Table 9) have been studied. Table 11 provides an overview of the research methods of the primary studies. The most common method is "Survey", which indicates that the results are generalized across different domains. Table 12 shows the distribution of research methods in relation to the comparison categories studied. The surveys address "COTS vs. OSS" (OC-OS) and "OSS" (OS) adoption decisions. The comparison category "In-house vs. COTS" (OI-OC) has a mix of case study and interview study methods. However, for the remaining comparison categories primary studies with same research methods are addressing them. The number of primary studies using the same research method per comparison category is also low (maximum two per category).

Table 11: Research methods

Research methods	Count	References
Survey (practitioners)	4	[35, 37, 42, 47]
Interview study	2	[38, 43]
Case study	2	[40, 58]
Case study (academic)	1	[55]

Table 12: Research methods and studied combinations of component origins

Research method	OI-OC	OC-OS	OS	OI-OS	OI-OO	OI-OC-OS
Survey		2	2			
Interview study	1					1
Case study	1			1		
Case study (academic)					1	
OI-OC: In-house vs. COTS, OI-OS: Inhouse vs. OSS, OC-OS: COTS vs. OSS						
OI-OO: In.house vs. Outsourcing, OS: Only OSS						
OI-OC-OS: In-house vs. COTS vs OSS						

Among the four surveys conducted as shown in Table 11, only two primary studies have reported domain details [35] and [37]. The domains covered by the two surveys is provided in Table 13. Among the interview studies and case studies, only one study each [43] and [40] respectively provides domains details which is also provided in Table 13.

Table 13: Domains of the primary studies

References	Telecom	Finance	ICT sector	Military
[35]	✓	✓	✓	
[37]	✓	✓	✓	
[43]	✓	✓	✓	
[40]				✓

The primary studies [35] and [37] belong to "COTS vs. OSS" comparison category and the primary studies [40] and [43] belong to "In-house vs. COTS" comparison category. From the results in Table 13 we can see that the results for "In-house vs. COTS" and "COTS vs. OSS" comparison categories are from different domains. Hence, the results are generalized with respect to the domains.

4.1.3. Quality assessment results

Table 14 shows the results of quality assessment. The empirical studies are evaluated with respect to rigor and relevance based on the rubrics discussed in Section 3.4. Note, the strength of evidence of empirical studies is only assessed, as the results of non-empirical papers such as opinion and philosophical papers are not based on any evidence. The total rigor and relevance is a sum of the scores assigned to each rigor and relevance aspect. If the total score of rigor is 1.5 or greater, the rigor is said to be high. And if the sum is less that 1.5, rigor is low. Similarly when the sum of the score of relevance is 2 or above, the relevance is high, and when the sum is below 2 the relevance is low.

Table 14: Rigor and relevance scores of the empirical studies

Ref.	Rigor				Relevance				
	C	V	RD	Total Rigor	SU	C	S	RM	Total Relevance
[35]	0.5	1	1	High (2.5)	1	1	1	1	High (4)
[37]	0.5	1	0.5	High (2)	1	1	1	1	High (4)
[38]	0.5	0	0.5	Low (1)	1	1	1	1	High (4)
[40]	0.5	0	0.5	Low (1)	1	1	1	1	High (4)
[42]	0	0	1	Low (1)	1	1	1	1	High (4)
[43]	1	1	1	High (3)	1	1	1	1	High (4)
[47]	0.5	0.5	1	High (2.5)	1	1	1	1	High (4)
[55]	0.5	0	0	Low (0.5)	0	0	0	0	Low(0)
[58]	0	0	1	Low (1)	1	1	1	1	High (4)
C - Context, V - Validity threats, RD - Research design, SU - Subjects/users, S- Scale, RM - Research Method									

Overall most of the empirical studies included in this study have high relevance. Except for the empirical study [55], which is performed in an academic context. Four out of nine studies are evaluated as having high rigor. Five out of nine studies are low in rigor. Most of the studies do not report any validity threats that might be taken into account. The context descriptions are not completely described. For example, in studies [47] and [55] only the company size is discussed, the size or domains of the projects are not mentioned. In addition, the research design descriptions are not completely reported in primary studies. For example, the study presented in [38] mentions that semi-structured interviews are conducted. However, the domain and the sampling of the interviewees are not mentioned and in the study presented in [40] the number of interviews and the roles interviewed are missing.

4.2. Influencing factors (RQ2)

The primary studies that discuss factors are mapped to the component origin comparisons as shown in Table 15. The "count" column indicates the number of studies specifically answering RQ2 out of the total number of primary studies found in this review study.

Table 15: Primary studies discussing influencing factors for component origin (RQ2)

Component origin	Count	References
OI-OC	2/8	[40, 43]
OC-OS	6/6	[35, 36, 37, 41, 44, 50]
OS	5/5	[42, 46, 47, 52, 57]
OI-OS	1/1	[58]
OI-OO	0/2	•
OI-OC-OS	2/2	[38, 45]
OI-OC: In-house vs. COTS, **OC-OS:** COTS vs. OSS, **OS:** Only OSS , **OI-OS:** In-house vs. OSS, **OI-OO:** In-house vs. outsource, **OI-OC-OS:** In-house vs. COTS vs OSS.		

The majority of primary studies considered OSS, followed by COTS, and In-house. Although OSS is discussed in most studies, only one study considers the comparison between in-house and OSS. The most researched combination of component origins is "OC-OS: COTS vs. OSS". There were no papers identified for the comparisons "OI-OO: In-house vs. Outsourcing". It is important to note that, the primary studies reported in Table 15 only discuss factors that are used in evaluation to choose a component origin (RQ2). They do not however discuss how the decision should be made or what decision-making process to follow (RQ3), this is discussed in Section 4.3. Therefore even though there are no papers discussing the comparison of in-house and outsourcing, the solutions to choose either in-house or outsourcing exists which are discussed in 4.3. Hence outsourcing is not discussed in Section 4.2.

The relevant text segments from the primary studies describing the influencing factors were coded into themes as shown in

Table 16: High-level themes, themes and codes

High-level themes	Themes/factors	Text segments
Project metrics factors	Time	Time to test and integrate
		Time to market
	Cost	Cost of components
		Total cost of ownership
		Cost of replacing components
		Maintenance cost
	Effort	Selection and integration effort
		Development effort
	Quality	Quality in general
External factors	Market trend	Component evolution
	Source code availability	Access and use of source code
		Source code documentation
	Technical support	Response time
		Support availability
		Code customization
		Changes in requirements
	License	License fee
		License obligations
Software development activity factors	Integration	Ease of integration
	Requirements	Task complexity
		Task uniqueness
		Requirement uncertainty
		Requirements negotiations
		Requirements suitability
	Maintenance	Ease of maintenance

Table 16. The themes are then grouped into higher-level themes which are also shown in Table 16.

The project metrics factors can be quantitatively measured and estimated. These factors can be estimated using estimation models. The external factors are those that are dependent on the provider (COTS vendor or OSS community). These factors usually define the access and control agreement. The software development activities that affect the decision are grouped into software development activity factors.

Sections 4.2.1, 4.2.2 and 4.2.3 provide the description of the factors (**bold**) and text segments (*italics*). Tables (17, 19 and 22) indicate the primary studies addressing the factors for each comparison category. Tables (18, 20 and 21) represent the summary of the description is terms of the effect the factors have on the component origin. It also indicates the level of evidence supporting the effect.

4.2.1. Project metrics factors

The project metrics factors extracted from primary studies with respect to the comparison categories (columns 2-5) are presented in Table 17, along with the count (C) of the primary studies reporting the factor and the number of primary studies that are empirical (E). The project metrics factors are discussed for COTS, OSS and In-house decisions. Among the primary studies in Table 17, most of the primary studies are discussing "OC-OS: COTS vs. OSS" (6) and "OI-OC-OS: In-house vs. COTS vs. OSS" (5) comparison category. Cost is the most discussed factor and has the most number of empirical studying reporting it as factor. This indicates that cost is an important project metric factor that influences the decision.

Time: The inclusion of COTS components causes delays as *testing and integrating components* (COTS) takes longer than in-house developed components [38]. However, when *time to market* in critical, COTS and OSS are preferred [35].

Table 17: Project metrics factors

Factors	OI-OC	OC-OS	OS	OI-OS	OI-OC-OS	C	E
Time	-	[35]	-	-	[38]	2	2/2
Cost	[40, 43]	[35, 36]	[42, 47]	-	[38, 45]	8	6/8
Effort	-	[35]	-	-	[38]	2	2/2
Quality	-	[35, 50]	[42, 46]	[58]	[45]	6	3/6

OI-OC: In-house vs. COTS, **OC-OS:** COTS vs. OSS,
OS: Only OSS, **OI-OS:** In-house vs. OSS,
OI-OC-OS: In-house vs. COTS vs OSS.
C: no.of primary studies, **E**: no.of empirical studies

Cost: *Cost of component* is a motivating factor for adopting OSS as it is available for free [35], [42], [47]. However, it was not necessarily a disadvantage for COTS component users as some projects preferred COTS because it is paid, therefore, assume it to have better quality [35]. In spite of the differences in the price of a component, the *total cost of ownership*, which includes the cost to integrate, test and maintain the components can end up to be the same for OSS and COTS components [36]. When the *cost of replacing components* is considered, there are no budget related issues as there was no investment involved in buying the component [45]. However, replacing COTS components based on customers' requirements was not cost effective [35]. *Maintenance cost* is used as a decision criterion in in-house vs. COTS decisions [38], [43]. The maintenance cost of externally acquired components like COTS and OSS can be really high in comparison to in-house components [38], [40].

Effort: The reduced development effort is the common motivation for selecting either COTS or OSS [35]. However, estimating the *selection and integration effort* for COTS and OSS components has been identified as a challenge [35]. The component selection is on a different decision level than the decision to select component origin (see Figure 1). However, the effort to select component is considered in the decision to choose component origin. The actual process of selecting the right components by considering all the risks is part of component selection decision level. Integrating external components requires considerable effort and if the component is small and requires more integration effort than *development effort* then, it is not the best option to acquire components externally [38].

Quality: OSS components might have better quality than COTS components however, it depends on various factors such as compensation, software demand, number of programmers, reward systems, programmer's cost of effort, and coordination of programmers [50]. In addition, it highly depends on the field testing and post-delivery fault reporting [58]. Quality has been regarded as an motivation factor for adopting OSS [35], [42], [46]. The quality of OSS components has been compared with COTS and in-house components [45], [50]. In some cases the quality of OSS was better than COTS and in-house components, as a hastily developed in-house component might not match the quality of an OSS component which has been widely used by

Table 18: Positive and negative influences of project metrics factors

Project Metrics Factors		COTS			OSS			In-house		
		H.H	L.H	N.E	H.H	L.H	N.E	H.H	L.H	N.E
Time	Time to test and integrate		-						+	
	Time to market	+			+			-		
Cost	Cost of components	+			+	+				
	Total cost of ownership			=			=			
	Cost of replacing components	-					+			
	Maintenance cost		-			-			+	
Effort	Selection and integration effort	-	-		-	-				
	Development effort	+			+			-		
Quality	Quality in general			-		+	+			-
H.H = High Rigor and High Relevance, L.H = Low Rigor and High Relevance, N.E = No evidence										

many OSS users [45].

Table 18 provides a summary of project metric factors with respect to the quality of the primary studies.

Though the results for time and effort factors are provided by only two studies respectively, the evidence supporting these factors is from empirical studies (see Table 17). The results for cost and quality are provided by empirical studies (H.H and L.H) and non-empirical studies (N.E). The empirical and non-empirical studies have different contribution towards the cost factors. For example cost factors such as cost of buying component, cost of replacing components and maintenance cost for COTS come from empirical studies. Where as, the cost factors from non-empirical studies are discussing total cost of ownership. Therefore, providing more information related to cost factors. In addition, the evidence and non-evidence studies allow comparisons. For example, according to empirical studies COTS has negative impact due to the cost of replacing component and the results from non-empirical studies indicate that OSS has positive impact. This allows us to compare how COTS and OSS are affected by cost when the results from both empirical and non-empirical studies are considered. Therefore, the results from empirical and non-empirical studies are complementary and there are no conflicts in results. OSS has more number of positive results except for maintenance cost and selection and integration effort factors. Which indicates that, when time, cost, effort and quality are important then OSS can be a suitable option.

4.2.2. External factors

The external factors extracted from primary studies with respect to the comparison categories are presented in Table 19 along with the count (C) of the primary studies reporting the factor and the number of primary studies that are empirical (E). These factors depend on external provider (COTS vendor and OSS community) and market.

The external factors are discussed for COTS, OSS and In-house development decisions. Among the primary studies in Table 19, most of the primary studies are discussing "OC-OS: COTS vs. OSS" (16) and "OS: Only OSS" (9) comparison category. Source code availability and technical support are the most discussed factors in the primary studies. However less than half of these studies are empirical as shown in column E.

Market trend: (*Component evolution*) Features that are novel today may become commodity tomorrow, and hence the market trend in terms of features affects which components that are suitable to, for example, develop in-house or use open source. Market trends are the needs of market place [40]. OSS development might evolve due to the market trends, however, the evolution might also be due to political or social reasons within the OSS community [35] or OSS developer ideas [41], which might not be necessarily align with the needs of market place. COTS is preferred over in-house development when following market trends and the need to include the newest technology is a priority [40] and [43]. The market pressure can however negatively affect the development of COTS components [36]. The market pressure can force the COTS vendor to release the products before they are fully developed or tested. Hence, following market trend can have positive as well as negative effect on COTS adoption.

Table 19: External factors

Factors	OI-OC	OC-OS	OS	OI-OS	OI-OC-OS	C	E
Market trend	[40, 43]	[35, 36, 41]	-	-	-	5	3/5
Source code availability	-	[35, 36, 37, 41, 44]	[42, 46, 47, 57]	-	[38, 45]	11	5/11
Technical support	[40]	[35, 36, 37, 41, 44, 50]	[46]	-	[45]	10	4/10
License	[40, 43]	[35, 36]	[42, 46, 47, 57]	-	[45]	8	4/8
OI-OC: In-house vs. COTS, **OC-OS:** COTS vs. OSS, **OS:** Only OSS, **OI-OS:** In-house vs. OSS, **OI-OC-OS:** In-house vs. COTS vs OSS. **C**: no.of primary studies, **E**: no.of empirical studies							

Source code availability: *Access to the source code* has been stated as one of the advantages of using OSS components [35] and unavailability of the source code is stated as one of the biggest disadvantages of using COTS components [36]. The existing literature explores the *usefulness of source code*. Code visibility allows the integrators to make necessary changes in the component [41]. However, a survey indicates that source code has been read to some extent but seldom changed [37].

Table 20: Positive and negative influences of external factors

External Factors		COTS			OSS		
		H.H	**L.H**	**N.E**	**H.H**	**L.H**	**N.E**
Market trend	Component evolution	+		-	-		-
Source code availibility	Access and use of source code			-	**+**	**+**	+,-
	Source code documentation					**-**	+,-
Techical support	Response time	+		-	-		+
	Support availability	-			+		
	Code customization			-	-		+
	Changes in requirements	+		-			-
License	License fee				+		
	License obligations					-	-
H.H = High Rigor and High Relevance, L.H = Low Rigor and High Relevance, N.E = No evidence							

Hence, if no such need arises to change the code, the availability of the source code does not seem important.

During the testing and maintenance phase, source code makes it easier to locate and fix errors as an integrator has access to the source code [38]. The integrators can learn and understand the OSS components and thereby save consulting costs [47]. Although source code is initially not available for COTS components, a survey [37] indicates that 33 percent of the COTS vendors opened their source code. The availability of the OSS source code can also be a disadvantage with respect to trust related issues [44]. It becomes easier to correlate all the information and plan an attack. Hence, the availability of source code can make the system vulnerable to such attacks [44]. Reviewing the source code written by someone else can be a laborious task [57]. Hence, the availability of source code can be an advantage, however the vulnerability to attacks can make it a disadvantage.

It can be difficult to understand the OSS *source code documentation* [41], [42] particularly, by inexperienced developers [46]. However, in one particular instance the documentation of the OSS code was better than the source code of a COTS component that was acquired at an additional cost [45]. The differences in results might be due to the dependencies on developer skills [46] and project profile [45]. The discussions on dependencies in continued in Section 5 and depicted in Figure 5.

Technical support: (*Response time of the vendor/community*) Prompt response of COTS vendor support has been reported to be good [35]. However, delayed response from COTS vendor are also reported [45]. Similarly the responsiveness of OSS community is reported to be good [45] and [44]. However, not having enough support from OSS community is also reported [35]. *Support availability:* The COTS vendors might refuse to provide support, if the users do not have the latest version and if there are not enough users that are affected. The support provided by OSS community does not pose such limitations [36], [46]. OSS users widely practice *customization of the code* and COTS vendors who open their source code do not support such changes and proclaim that any modification will result in loss of support [50]. However, conflicting results indicating that OSS community might not extend their support if the code is customized is reported [35].

It is also possible to get the *required changes* from the COTS vendors [37]. However, smaller organizations adopting COTS and OSS components have little influence on the component evolution [41]. Developing and maintaining an effective relationship with vendors is a new activity, which is not needed in in-house development [40]. The results related to technical support are conflicting and it is uncertain how the COTS vendors or OSS communities react to changes in the source code. However, it highly depends on the profile of the vendor/community [50], size of the organization adopting COTS and/or OSS [41] and number of users [36], [46]. The discussions on dependencies in continued in Section 5 and depicted in Figures 4 and 5.

License: License includes the terms, conditions, and costs for a given product for use by an organization over a particular period of time [40]. It also covers the vendor relationship including integration support [40]. Low *cost of licenses* has been a motivation for adopting OSS components [35], [47]. However, the *license obligations* such as the general public license are considered too restrictive as, any modifications made to the source code should also be open which is referred to as copyleft [36], [42]. In addition, all applications linked to the code must also be made available [42], [45]. In order to avoid this and due to intellectual property (IP) concerns, the components with less restrictive licenses are preferred [47]. In addition, obtaining such a non-restrictive license has been regarded as a daunting process for a mission critical applications [45]. Hence, the license obligations must be thoroughly evaluated before adopting OSS components [43], [57].

Table 20 provides a summary of external factors with respect to the quality assessment of the primary studies.

As Table 20 is summarizing external factors, these factors do not have an impact on in-house development. Hence, in-house option is not mentioned in the table. The results are from empirical studies (H.H, L.H) as well as non-empirical studies (N.E). The results from empirical studies are contradicting results from non-empirical studies. The evidence from empirical study indicates that COTS is preferred when following market trend is important. However the results of non-empirical studies state that the market pressure can have negative impact as discussed earlier in this section. The conflicts in source code availability and technical support factors are due to the dependencies mentioned earlier, these dependencies are also discussed in Section 5 (see Figures 4 and 5). From the Table 20, we can see that overall OSS is preferred when external factors are considered.

4.2.3. Software development process factors

The software development process factors extracted from primary studies with respect to the comparison categories are presented in Table 22 along with the count (C) of the primary studies reporting the factor and the number of primary studies that are empirical (E).

Table 22: Software development process factors

Factors	**OI-OC**	**OC-OS**	**OS**	**OI-OS**	**OI-OC-OS**	**C**	**E**
Integration	-	[35, 36, 41]	-	-	[38]	4	2/4
Requirements	[40, 43]	[35]	[46, 52]	-	[38, 45]	7	4/7
Maintenance	[40]	[35, 36, 41]	-	-	[38]	5	3/5
OI-OC: In-house vs. COTS, **OC-OS:** COTS vs. OSS, **OS:** Only OSS, **OI-OS:** In-house vs. OSS, **OI-OC-OS:** In-house vs. COTS vs OSS. **C**: no.of primary studies, **E**: no.of empirical studies							

The software development process factors are discussed for COTS, OSS and In-house development decisions. Among the primary studies in Table 22, most of the primary studies are discussing "OC-OS: COTS vs. OSS"(7) and "OI-OC-OS: In-house vs. COTS vs. OSS"(4) comparison category. Requirements factor is the most discussed factor and has the most number of empirical studying reporting it as factor. This indicates that the decision to choose the component should be taken as early as the requirements phase.

Integration: (*Ease of integration*) OSS integration and COTS integration is similar if the source code is not exploited [36]. It is difficult to identify if the defects are inside or outside the COTS or OSS components, even if the source code is available for OSS most users usually do not look at the code [35]. However, without the source code it is difficult to understand the working of the component (COTS) in particular when there are issues such as environmental differences, deviations from supported protocols, etc. Consequently making it difficult to integrate COTS components [41]. The difficulty level of integrating COTS components is considered in the in-house vs. COTS decisions [38].

Requirements: (*Task complexity and uniqueness*) The decision to choose between in-house, COTS and OSS is mostly taken in the requirements phase [43], [46]. The developers prefer to use pre-built components such as OSS libraries when the task complexity is high to improve productivity [46]. However when unique development efforts (developing innovative functionality) are required to fulfil a requirement, OSS is unlikely to be used [46]. In addition, when there is *requirement uncertainty*, the tendency to adopt existing OSS components is high [52]. In-house development starts with a specific set of system requirements and builds a system that meets those requirements. However, the COTS based system development starts with a generic set of requirements and then the market place is explored to see how closely they match the needs of the system [40]. Though this does not apply to all COTS based systems.

The requirements should be carefully analyzed before the decision to acquire external components such as COTS is made [38]. The *requirement negotiations* and evaluations of external components are too long. During the negotiations and evaluations the technical requirements might change and the component might no longer be needed [38]. The available OSS or COTS component might not perfectly match the requirements and therefore, might not be *suitable* to use. The customers might agree to the requirements changes that are necessary to utilize OSS component [45]. However, not adapting sufficiently to the customers' requirements and not having the possibility to negotiate customer requirements has been perceived as a common risk for COTS and OSS components [35]. This could be the reason why OSS or COTS components may not be preferred when the requirements require unique development effort and are not very flexible to changes.

Maintenance: One of the challenges in COTS deployment and maintenance is to strike a balance between system stability and being up to date with the market place [35, 40]. The COTS vendor usually decides the rate at which the component changes and the organization that acquires the COTS component should decide if they want to upgrade or lose the support for the component they are currently using [41]. Maintenance policies related to updates from provider can force the component to be upgraded to a newer version so that the technical support provided from the vendors can be retained. Therefore, maintenance policies are used as an evaluation criterion in the in-house vs. COTS decisions [38]. The OSS users are not obliged to change whenever there is any upgrade [36].

Table 21 provides a summary of software development process factors with respect to the quality of the primary studies.

The results are supported by empirical (H.H, L.H) and non-empirical (N.E) studies. The results from empirical studies are supporting non-empirical studies and there are no conflicts

Table 21: Positive and negative influences of software development process factors

Software development process Factors		**COTS**			**OSS**			**In-house**		
		H.H	**L.H**	**N.E**	**H.H**	**L.H**	**N.E**	**H.H**	**L.H**	**N.E**
Integration	Ease of integration			-	-				+	
Requirements	Task complexity						+			-
	Task uniqueness						-			+
	Requirement uncertainty						+			
	Requirements negotiations		-							
	Requirements suitability	-			-					
Maintenance	Ease of maintenance	-	-	-			+	+		
H.H = High Rigor and High Relevance, L.H = Low Rigor and High Relevance, N.E = No evidence										

in results. OSS and in-house development component origins have more number of positive results compared to COTS. Which indicates that, when integration and maintenance are important then either OSS or in-house development can be a suitable option.

4.3. Solutions (RQ3)

The primary studies that discuss solutions to choose a component origin (RQ3) for the software components are listed in Table 23. The "Count" column indicates the number of studies specifically answering RQ3 out of the total number of studies found in this review study.

Table 23: Primary studies proposing solutions for choosing a component origin (RQ3)

Component origin	Count	References
OI-OC	6/8	[39, 48, 49, 51, 53, 54]
OC-OS	0/6	•
OS	0/5	•
OI-OS	0/1	•
OI-OO	2/2	[55, 56]
OI-OC-OS	0/2	•

OI-OC: In-house vs. COTS, **OC-OS:** COTS vs. OSS, **OS:** Only OSS , **OI-OS:** In-house vs. OSS, **OI-OO:** In-house vs. outsource, **OI-OC-OS:** In-house vs. COTS vs OSS.

4.3.1. OI-OC: In-house vs. COTS

For the "OI-OC: In-house vs. COTS" decisions six studies provided solutions. The solutions [39], [48], [49], [51], [53] and [54] define optimization models.

Papers [39] and [51] help to select the component that should be developed in-house and the components that should be bought. An optimization model is proposed which minimizes cost under reliability and delivery time constraints by selecting the right origin (in-house development or COTS) of components. It also considers the amount of testing required to develop the component in-house into the decision-making. The selection of component origin is done after the software design is available. For COTS components, the time is measured in terms of the time taken by the vendor to provide the component and the time taken to adapt the component. For the in-house component, the time taken to develop the component and to achieve the desired level of reliability is measured. Cost is measured in terms of the cost required to buy and adapt the COTS components, similarly the cost for developing in-house components is also estimated.

The paper [48] is an extension of the solution proposed in [39], suggesting an optimization model by taking into account the reliability, delivery time and cost. The addition to the model is the ability to take the decision as soon as the requirements are available.

A multi-objective approach for an optimal "OI-OC: In-house vs. COTS" decision for a fault tolerant system was formulated by [49]. The two objectives in decision-making are maximizing system reliability and minimizing the system cost under the constraint of delivery time. In addition, it also considers compatibility between available alternative components.

The solution in the paper [53] proposes an architecture optimization approach based on a meta-heuristic swarm intelligence algorithm to decide the origin (In-house or COTS) of the components. The algorithm considers delivery time, cost and reliability constraints. It also considers the number of test cases for in-house developed components.

The paper [54] presents a solution to maximize the intra-module coupling density under a number of constraints such as delivery time, cost, reliability, compatibility, cohesion, coupling, requirements fit and test success.

4.3.2. OI-OO: In-house vs. Outsourcing

The solution in paper [55] proposes a methodology and tool support to decide the components that need to be outsourced and which ones should be developed internally. The tool classifies the projects' software components by means of a graph based model of the components' requirements and their corresponding clustering with respect to requirements dependencies and priorities.

The paper [56] proposes a solution that determines the outsourcing potential (if it is high the component should be outsourced), input is knowledge specificity (based on business, functional and technical), and interdependencies (priority, between software components and communication intensity among developers). A decision is made with the help of decision tables.

5. Discussion

From the high-level themes discussed in Sections 4.2.1, 4.2.2 and 4.2.3, we can see that OSS has more positive influence on the decision when project metrics and external factors are considered. In-house development and OSS have more positive influence when software development process factors are considered. However, when the individual factors/themes are considered each component origin has some advantage over some other component origins. The results indicate that some trade-offs between factors and dependencies on the context need to be considered which is discussed in this section along with the advantage and disadvantages of the component origins.

The advantages are discussed in Table 24. The advantages of outsourcing are not discussed as none of the primary studies have mentioned them. The information presented in Table 24 can be used as an initial set of factors to be considered by practitioners in the decision-making.

In addition, there are some trade-offs that need to be considered along with the advantages. Figures 4, 5 and 6 illustrate the trade-offs between the different factors. The trafe-offs that should be considered along with advantages of COTS over OSS are depicted in Figure 4 and discussed in Section 5.1. The trafe-offs that should be considered along with advantages of OSS over COTS are depicted in Figure 5 and discussed in Section 5.2. Whereas, Figure 6 discusses trade-offs that are related to the advantages of in-house development over COTS and OSS, which is discussed in Section 5.3. Section 5.4 discusses the advantages of COTS and OSS over in-house development, no

Table 24: Advantages of component origins

Component origin	Advantages
COTS over OSS	Market Trend [35]
OSS over COTS	Low cost of replacing component [35, 45], Source code available [35, 38, 41, 47, 50], Low licensing cost [35, 47], Ease of debugging integration defects [41], Ease of maintaining system stability [36], Better quality [35, 42, 46, 50]
In-house over COTS & OSS	Ability to add unique functionality [46], Reduced maintenance cost [38, 40], Easy integration [35, 41], Reduced testing time [38]
COTS & OSS over In-house	Reduced time to market [35], Reduced development effort [35], Ability to add complex functionality [46]

trade-offs were mentioned for the advantages discussed in Section 5.4 in the primary studies. The trade-offs and dependencies depicted in Figures 4, 5 and 6 are aggregated from different primary studies. The "↔" arrow represents the trade-off between different factors and "→" arrow represents the dependencies on the context that affects the factors. Depending on the context, the factors have positive or negative impact.

5.1. COTS over OSS

If following the market trend is important, practitioners might choose COTS components as COTS components are more likely to follow the market trend. One should also consider the trade-off between the following factors: market trend, maintenance and technical support as shown in Figure 4.

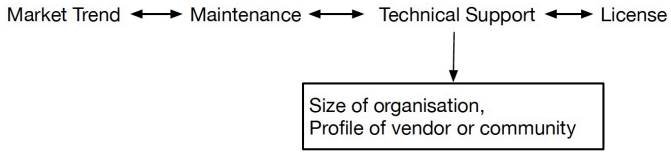

Figure 4: Trade-offs and dependencies for advantages of COTS over OSS

Keeping up with the market trend can mean that components must be frequently upgraded [35], [36], [40]. Frequent upgrades might hinder the system stability. In addition, if components are not updated to the latest version, there is a risk of losing technical support offered by the COTS vendor [41]. Hence, the trade-off between market trend, maintainability and technical support should be considered. Technical support offered depends on the license agreements and context information such as size of the organization [41] (large-scale organization have better chances with license negotiations) and profile of the vendor or community [50] (depends on how flexible the vendor/community providing the component).

5.2. OSS over COTS

OSS is preferred over COTS when there is a possibility for lot of requirement changes, as there is no impact on the budget [35], [45]. If OSS components are chosen because the source code is available, the trade-off between source code availability, technical support and license must be considered as shown in Figure 5.

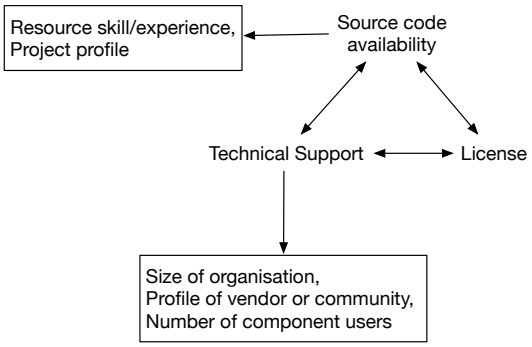

Figure 5: Trade-offs and dependencies for advantages of OSS over COTS

The source code can help to identify defects during integration, although it depends on the developers' skill or experience in finding the defects in source code [46]. In high-risk projects, the source code might need to be reviewed [45]. However, in low-risk projects, such reviews might not be necessary. Hence, if the project has a high-risk profile and the developers or integrators are skilled and experienced, the availability of source code is beneficial. On the other hand, if the project does not have a high-risk profile and the developers or integrators do not have the required skill or experience, the availability of source code is not of any benefit to the practitioners.

Another benefit of source code being available is the possibility to change the code. However, the trade-off between technical support and license needs to be considered. Some OSS communities might refuse to support if the component is customized, although it highly depends on the profile of the vendor of community [50], size of the organization using the component [41] and number of users using the component [36], [46]. In addition, OSS communities might impose a restrictive license, any changed code must be given back to the OSS community [35].

5.3. In-house over OSS and COTS

One of the advantages of COTS and OSS is reduced development effort [35]. However, integrating COTS and OSS is time consuming [35]. Hence, the trade-off between development effort and integration effort must be considered as shown in Figure 6.

Figure 6: Trade-offs and dependencies for in-house, COTS and OSS factors

If integrating COTS or OSS components might take longer than developing components, then they should be developed in-house rather than acquired externally. The development time might vary based on the size of the component. In addition, some studies [38] and [40] report that maintenance cost of in-house developed components is lower than COTS and OSS components.

5.4. COTS and OSS over In-house

The pressure to develop faster due to market competitiveness is identified as a challenge in in-house development [1]. COTS and OSS are known to be viable options when time to market is a criterion [35]. Constant requirements changes resulted in a lot of wasted development effort [1], which can be avoided when external components are used: In particular, for OSS there are no additional costs involved in buying the component [35]. Note, the effort is referred to man-hours and not calendar time. Challenges related to understanding complex requirements were identified [1]. The developers prefer to use pre-built components such as OSS libraries when the task complexity is high to improve productivity [46].

5.5. Research gaps

The purpose of this systematic review is also to identify research gaps and directions for future studies. Overall, only a few studies have been found in the area researched. Thus, very specific research gaps could be identified looking at the existing literature.

As shown in Table 23 the majority of solutions has been proposed for "OI-OC: In-house vs. COTS". The solutions that have been proposed were optimization models using different factors such as time, cost and quality (reliability in specific). Other component origins have no solutions, or very few (at most two studies). Given that solutions are often dependent on the context in which they are applied, more evaluations of them are needed, and a wider array of new solutions may be needed to address the complex decision problem of choosing a component origin.

The factors that influence the decision to choose a component origin (Sections 4.2.1, 4.2.2 and 4.2.3) and the factors that are used in solutions proposals (Section 4.3) are mapped in Table 25.

Table 25: Mapping of factors influencing the decision and factors considered in the solutions

Factors influencing the decision to choose a component origin	**Factors considered in solutions**
Project metric factors	
Time to market and time to test and integrate	COTS - Time taken by the vendor to provide components and the adaptation time. In-house - Time taken to develop components.
Cost of buying, replacing and maintaining components	Cost of buying and adapting COTS components
Selection, development and integration effort	Not considered
Quality	Reliability
External factors	
Market trend	Not considered
Source code availability	Not considered
Technical support	Not considered
License	Not considered
Software development activity factors	
Integration	Not considered
Requirements	Not considered
Maintenance	Not considered

As seen in Table 25, the solutions only consider project metric factors (cost, time and quality) excluding effort. The external factors and software development activity factors are not considered by any of the solutions. In addition, the costs considered are for buying and adapting COTS components. However, some of the primary studies also discuss the influencing cost factors that need to consider cost of buying [35], [42], [47], replacing [42] and maintaining components [38], [40], [43]. Also even though time to obtain and adapt components are considered, time to market is not considered by any of the solutions. Reliability is considered in almost all proposed solutions, although it did not emerge as a most important factor in the studies that compared the different component origin. Unlike the solutions, non-technical aspects such as vendors' profile are used to evaluate quality [42]. The integration effort has not been considered by any of the solutions, even though it is mentioned as an influencing factor in the in-house vs. COTS and OSS decisions [35], [36], [38], [41]. In addition, an important trade-off between integration effort and development effort has been identified as shown in Figure 6. One possible reason for not including integration effort may be because it is difficult to estimate the effort required to integrate external components such as COTS and OSS.

Among the solutions provided for "OI-OC: In-house vs. COTS" and "OI-OO: in-house vs. outsourcing", in-house is the common component origin, although the solutions provided do not have any common factors that are considered in the solutions.

Considering the above remarks, it can be said that the proposed solutions have not necessarily considered the most important factors that are needed in the decision to choose a component origin. This means that we may say that solutions do not consider factors that are from a practical perspective potentially relevant, this is a disadvantage as the solutions focus on too few factors, i.e. when looking at practice they may not be fully applicable, not reflect the reality.

Few studies (maximum 6, see Table 15) per combination of component origins were identified. Thus, it is hard to judge whether the component origins discussed are the most relevant ones for the decision to be made. One study [57] highlighted, according to what the authors believed, the most important factors that need to be considered while selecting OSS components. The relative importance of factors for the different combinations of component origins is unknown and hence based on the existing studies it is impossible to suggest the ones that are most important. In addition, the order in which the factors should be considered is not known. Though, the findings can serve as an initial inventory of factors to be considered. If a factor is mentioned for one of the component origins, but not for another, also does not necessarily mean that it is irrelevant for that component origin.

Conflicting statements with regard to external factors were found, but not for all other combinations. This may be due to the low number of studies and indicates that to make a reliable statement with regard to a positive or negative effect in relation to different factors, it is not possible to provide reliable guidance to practitioners in their decision-making. In particular, to

have good decision support it is important to understand the strengths and weaknesses of the alternatives with regard to the factors and outcomes empirically. With this understanding, in combination with knowledge about the importance of the factors, more informed decisions can be made. With the current solutions, insufficient data is available to do so.

6. Conclusion

A systematic review of the factors affecting the decision to choose a component origin has been conducted. A component origin is a source to get the components from (here in-house development, open-source, COTS and outsourcing). Three research questions were asked:

RQ1: What are the research types, methods and quality of the contributions? All the research types classified by Wieringa [19] were identified. Hence, the primary studies consisted of a good mix of empirical and non-empirical results. This distribution of the research types allowed to identify conflicting results based on context dependencies. The most common research type is evaluation research indicating that the research is empirical and rooted in industrial practice, which is a positive result. The most common research method is survey indicating that the results are not specific to a particular context. However the context is not described sufficiently in the primary studies. The results of quality assessment indicate the lack of rigor in the primary studies in terms of reporting the context, validity threats and research design.

RQ2: What are the different factors that influence the decision to choose among different component origins? In total eleven factors grouped into three high-level themes that influence the decision to choose a component origin were identified. The project metrics factors are: time, cost, effort and quality. External factors such as market trend, source code availability, technical support and license were identified. In addition, the software development activity factors such as integration, requirements, and maintenance activities that have an impact on the decision were identified. The results from empirical are supporting and complementing results from non-empirical studies. However in some cases, conflicts are identified (see for example Table 20). The conflicts are due to the context dependencies.

The project metrics and external factors positively affect the adoption of OSS components. However, the trade-off between development effort and integration effort needs to be considered when choosing between in-house and OSS components (see Figure 6). Whereas, when external factors are considered, the context dependencies on the source code availability and technical support factors need to be considered (see Figure 5). The software development activity factors positively affect the decision to adopt OSS components and in-house development. Overall COTS has a negative impact on adoption however, it is preferred over OSS and in-house when following the market trend is important.

RQ3: What solutions have been proposed to choose the component origin? The solutions were mainly focused on choosing between in-house development and COTS and were mainly based on optimization models.

The overall implication is that there are too few studies, and hence too few contexts investigated to provide recommendations for practitioners on how to make decisions between the component origins. That being said, the initial set of studies identified may serve as a basis for future studies.

Three decision levels are shown in Figure 1. The factors influencing provider selection are different from component origin selection [3]. The factors influencing component selection are similar to the factors influencing component origin selection [21]. For example, the component integration effort is considered in component selection. However, the comparison with development effort is not considered. The decision to select the component origin must be taken first before selecting the component. While there are secondary studies on provider selection [3] and component selection [21] level, this systematic literature review is the first secondary study on selecting component origin. The research field of choosing between component origins is in great need of future studies to provide comprehensive approaches to support decision-making. This is true in terms of the number of solutions proposed, and the extent to which empirical evaluations have been conducted.

As a consequence, future work has to focus on (a) determining the order of importance and magnitude of the factors; (b) providing empirical evidence on comparisons of component origins with regard to the factors; and (c) proposal of novel solutions taking (a) and (b) into consideration.

7. Acknowledgement

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

References

[1] D. Badampudi, S. A. Fricker, A. M. Moreno, Perspectives on productivity and delays in large-scale agile projects, in: Proceedings of the 14th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP 2013), Springer Berlin Heidelberg, 2013, pp. 180–194.

[2] M. Morisio, M. Torchiano, Definition and classification of cots: a proposal, in: COTS-Based Software Systems, Springer, 2002, pp. 165–175.

[3] S. U. Khan, M. Niazi, R. Ahmad, Factors influencing clients in the selection of offshore software outsourcing vendors: An exploratory study using a systematic literature review, Journal of systems and software 84 (4) (2011) 686–699.

[4] T. Helokunnas, M. Nyby, Collaboration between a cots integrator and vendors, in: Software QualityECSQ 2002, Springer, 2002, pp. 267–273.

[5] J. Xu, Y. Gao, S. Christley, G. Madey, A topological analysis of the open souce software development community, in: System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, IEEE, 2005, pp. 198a–198a.

[6] R. Land, L. Blankers, M. Chaudron, I. Crnković, Cots selection best practices in literature and in industry, in: High Confidence Software Reuse in Large Systems, Springer, 2008, pp. 100–111.

[7] T. Wanyama, B. Far, An empirical study to compare three methods for selecting cots software components, International Journal of Computing and ICT Research 2 (1) (2008) 34–46.

[8] C. Ayala, X. Franch, A goal-oriented strategy for supporting commercial off-the-shelf components selection, in: Reuse of Off-the-Shelf Components, Springer, 2006, pp. 1–15.

[9] V. Maxville, J. Armarego, C. P. Lam, Applying a reusable framework for software selection, Software, IET 3 (5) (2009) 369–380.
[10] N. D. Anh, D. S. Cruzes, R. Conradi, M. Höst, X. Franch, C. Ayala, Collaborative resolution of requirements mismatches when adopting open source components, in: Requirements Engineering: Foundation for Software Quality, Springer, 2012, pp. 77–93.
[11] D. J. Carney, F. Long, What do you mean by cots? finally, a useful answer, IEEE Software 17 (2) (2000) 83–86.
[12] C. Gacek, B. Arief, The many meanings of open source, Software, IEEE 21 (1) (2004) 34–40.
[13] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: 8th International Conference on Evaluation and Assessment in Software Engineering (EASE 2014), ACM, 2014, pp. 321–330.
[14] D. Badampudi, C. Wohlin, K. Petersen, Experiences from using snowballing and database searches in systematic literature studies, in: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2015, p. 17.
[15] K. Petersen, C. Gencel, Worldviews, research methods, and their relationship to validity in empirical software engineering research, in: Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on, IEEE, 2013, pp. 81–89.
[16] H. Zhang, M. A. Babar, P. Tell, Identifying relevant studies in software engineering, Information & Software Technology 53 (6) (2011) 625–637.
[17] S. Jalali, C. Wohlin, Systematic literature studies: database searches vs. backward snowballing, in: Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, ACM, 2012, pp. 29–38.
[18] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering–a systematic literature review, Information and software technology 51 (1) (2009) 7–15.
[19] R. Wieringa, N. A. M. Maiden, N. R. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: a proposal and a discussion, Requir. Eng. 11 (1) (2006) 102–107.
[20] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. d. M. S. Neto, Y. C. Cavalcanti, S. R. de Lemos Meira, Twenty-eight years of component-based software engineering, Journal of Systems and Software 111 (2016) 128–148.
[21] M. Morandini, A. Siena, A. Susi, Risk awareness in open source component selection, in: Business Information Systems, Springer, 2014, pp. 241–252.
[22] M. Syeed, I. Hammouda, T. Syatä, Evolution of open source software projects: A systematic literature review, Journal of Software 8 (11) (2013) 2815–2829.
[23] M. Sultan, A. Bakar, A. Diwani Bakar, H. Zulzalil, J. Din, Systematic literature review: the important factors in assessing the trustworthiness of oss., International Journal on Communication Antenna & Propagation (IRECAP) 3 (1) (2013) 56–60.
[24] S. T. Acuña, J. W. Castro, O. Dieste, N. Juristo, A systematic mapping study on the open source software development process, in: Evaluation & Assessment in Software Engineering (EASE 2012), 16th International Conference on, IET, 2012, pp. 42–46.
[25] A. B. M. Sultan, A. D. Bakar, H. Zulzalil, J. Din, Systematic literature review in open source software maintainability., International Review on Computers & Software 7 (5).
[26] M. Höst, A. Oručević-Alagić, A systematic review of research on open source software in commercial software product development, Information and Software Technology 53 (6) (2011) 616–624.
[27] K.-J. Stol, M. A. Babar, P. Avgeriou, B. Fitzgerald, A comparative study of challenges in integrating open source software and inner source software, Information and Software Technology 53 (12) (2011) 1319–1336.
[28] K.-J. Stol, M. A. Babar, B. Russo, B. Fitzgerald, The use of empirical methods in open source software research: Facts, trends and future directions, in: Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, IEEE Computer Society, 2009, pp. 19–24.
[29] H. P. Breivold, M. A. Chauhan, M. A. Babar, A systematic review of studies of open source software evolution, in: Software Engineering Conference (APSEC), 2010 17th Asia Pacific, IEEE, 2010, pp. 356–365.
[30] Ø. Hauge, C. Ayala, R. Conradi, Adoption of open source software in software-intensive organizations–a systematic literature review, Information and Software Technology 52 (11) (2010) 1133–1154.
[31] J. Maras, L. Lednicki, I. Crnkovic, 15 years of cbse symposium: impact on the research community, in: Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering, ACM, 2012, pp. 61–70.
[32] I. Crnkovic, Component-based software engineeringnew challenges in software development, Software Focus 2 (4) (2001) 127–133.
[33] D. S. Cruzes, T. Dybå, Recommended steps for thematic synthesis in software engineering, in: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, IEEE, 2011, pp. 275–284.
[34] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Empirical Software Engineering 16 (3) (2011) 365–395.

Systematic Review Study

[35] 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: Proceedings of the 28th international conference on Software engineering, ACM, 2006, pp. 897–900.
[36] P. Di Giacomo, Cots and open source software components: are they really different on the battlefield?, in: COTS-Based Software Systems, Springer, 2005, pp. 301–310.
[37] J. Li, R. Conradi, O. P. N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, M. Morisio, Validation of new theses on off-the-shelf component based development, in: Software Metrics, 2005. 11th IEEE International Symposium, IEEE, 2005, pp. 26–26.
[38] T. Helokunnas, The dimensions of embedded cots and oss software component integration, in: Product Focused Software Process Improvement, Springer, 2002, pp. 509–518.
[39] V. Cortellessa, F. Marinelli, P. Potena, Automated selection of software components based on cost/reliability tradeoff, in: Software Architecture, Springer, 2006, pp. 66–81.
[40] L. Brownsword, T. Oberndorf, C. A. Sledge, Developing new processes for cots-based systems, IEEE Software 17 (4) (2000) 48–55.
[41] S. A. Hissam, C. B. Weinstock, Open source software: The other commercial software, in: 1st Workshop on Open Source Software at ICSE, 2001.
[42] K. J. Stewart, A. P. Ammeter, L. M. Maruping, A preliminary analysis of the influences of licensing and organizational sponsorship on success in open source projects, in: System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, IEEE, 2005, pp. 197c–197c.
[43] 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, Empirical Software Engineering 11 (3) (2006) 433–461.
[44] S. A. Hissam, D. Plakosh, C. Weinstock, Trust and vulnerability in open source software, IEE Proceedings-Software 149 (1) (2002) 47–51.
[45] J. S. Norris, Mission-critical development with open source software: Lessons learned, Software, IEEE 21 (1) (2004) 42–49.
[46] R. Torres, Developer-led adoption of open source software libraries: A conceptual model, in: 18th Americas Conference on Information Systems, 2012, pp. 583–586.
[47] W. Chen, J. Li, J. Ma, R. Conradi, J. Ji, C. Liu, An empirical study on software development with open source components in the chinese software industry, Software Process: Improvement and Practice 13 (1) (2008) 89–100.
[48] P. Potena, Composition and tradeoff of non-functional attributes in software systems: research directions, in: The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers, ACM, 2007, pp. 583–586.
[49] P. Jha, S. Bali, U. D. Kumar, H. Pham, Fuzzy optimization approach to component selection of fault-tolerant software system, Memetic Computing 6 (1) (2014) 49–59.
[50] B. Mishra, A. Prasad, S. Raghunathan, Quality and profits under open source versus closed source, ICIS 2002 Proceedings (2002) 32.

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

[52] S. Koch, Open source as a sourcing strategy for corporations, International Journal of Business Innovation and Research 5 (1) (2010) 1–16.

[53] A. A. Ssaed, W. Kadir, S. Z. M. Hashim, Metaheuristic search approach based on in-house/out-sourced strategy to solve redundancy allocation problem in component-based software systems, International Journal of Software Engineering and Its Applications 6 (4) (2012) 143–154.

[54] P. Jha, V. Bali, S. Narula, M. Kalra, Optimal component selection based on cohesion & coupling for component based software system under build-or-buy scheme, Journal of Computational Science 5 (2) (2014) 233–242.

[55] T. Kramer, M. Eschweiler, Outsourcing location selection with soda: a requirements based decision support methodology and tool, in: Advanced Information Systems Engineering, Springer, 2013, pp. 530–545.

[56] T. Kramer, A. Heinzl, K. Spohrer, Should this software component be developed inside or outside our firm?-a design science perspective on the sourcing of application systems, in: New Studies in Global IT and Business Service Outsourcing- 5th Global Sourcing Workshop 2011, Courchevel, France, March 14-17, 2011, Springer, 2011, pp. 115–132.

[57] J. Rudzki, K. Kiviluoma, T. Poikonen, I. Hammouda, Evaluating quality of open source components for reuse-intensive commercial solutions, in: Software Engineering and Advanced Applications, 2009. SEAA'09. 35th Euromicro Conference on, IEEE, 2009, pp. 11–19.

[58] S. M. Syed-Mohamad, T. McBride, A comparison of the reliability growth of open source and in-house software, in: 15th Asia-Pacific Software Engineering Conference (APSEC 2008), 2008, pp. 229–236.

*Biography

Deepika Badampudi is a PhD student at Blekinge Institute of Technology. Her research interests are component-based software engineering, agile software development, and evidence-based software engineering.

Claes Wohlin is a professor of software engineering and dean for the Faculty of Computing at Blekinge Institute of Technology, Sweden. He has previously held professor chairs at the universities in Lund and Linkoping. His research interests include empirical methods in software engineering, software metrics, software quality, and requirements engineering. Wohlin received a PhD in communication systems from Lund University. He is Editor-in-Chief of Information and Software Technology. Claes Wohlin was the recipient of Telenor's Nordic Research Prize in 2004 for his achievements in software engineering and improvement of reliability for telecommunication systems. He is a member of the Royal Swedish Academy of Engineering Sciences.

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

*Title page with author details

Manuscript submitted to JSS

Title: Software component decision-making: In-house, OSS, COTS, or Outsourcing – A Systematic Literature Review

Authors:

Deepika Badampudi
Blekinge Institute of Technology
Campus Gräsvik
Deepika.badampudi@bth.se

Claes Wohlin
Blekinge Institute of Technology
Campus Gräsvik
claes.wohlin@bth.se

Kai Petersen
Blekinge Institute of Technology
Campus Gräsvik
Kai.petersen@bth.se

LaTeX Source Files

Click here to download LaTeX Source Files: Latex Source Files.zip