Thesis no: MSSE-2016-07

Usage of third party components in Heterogeneous systems

An empirical study

Jaya Krishna Raavi

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Jaya Krishna Raavi
E-mail: jara14@student.bth.se

University advisor:
Ahmad Nauman Ghazi
Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

Context: The development of complex systems of systems leads to high development cost, uncontrollable software quality and low productivity. Thus Component-based software development was used to improve development effort and cost of the software. Heterogeneous systems are the system of systems that consist of functionally independent sub-systems with at least one sub-system exhibiting heterogeneity with respect to other systems. The context of this study is to investigate the usage of third party components in heterogeneous systems.

Objectives. In this study an attempt was made to investigate the usage of third party components in heterogeneous systems in order to accomplish the following objectives:

- Identify different types of third party components.
- Identify challenges faced while integrating third-party components in heterogeneous systems.
- Investigate the difference in test design of various third party components
- Identify what the practitioners learn from various third party components

Methods: We have conducted a systematic literature review by following Systematic literature review guidelines by Kitchenham to identify third party components used, challenges faced while integrating third-party components and test design techniques. Qualitative interviews were conducted in order to complement, supplement the finding from the SLR and further provide guidelines to the practitioners using third party components. The studies obtained from the SLR were analyzed in relation to the quality criteria using narrative analysis. The data obtained from interview results were analyzed using thematic analysis.

Results: 31 primary studies were obtained from the systematic literature review (SLR). 3 types of third components, 12 challenges, 6 test design techniques were identified from SLR. From the analysis of interviews, it was observed that a total of 21 challenges were identified which complemented the SLR results. In addition, from interview test design techniques used for testing of heterogeneous systems having third party components were investigated. Interviews have also provided 10 Recommendations for the practitioners using different types of third party components in the product development.

Conclusions: To conclude, commercial of the shelf systems (COTS and Open software systems (OSS) were the third party components mainly used in heterogeneous systems rather than in-house software from the interview and SLR results. 21 challenges were identified from SLR and interview results. The test design for testing of heterogeneous systems having different third party components vary, Due to the non-availability of source code, dependencies of the subsystems and competence of the component. From the analysis of obtained results, the author has also proposed guidelines to the practitioners based on the type of third party components used for product development.

Keywords: Heterogeneous Systems, Third party components, Test design, Testing techniques

ACKNOWLEDGEMENTS:

First I would like to pay gratitude to my parents, brother for their unconditional love, support, and being my huge strength throughout this journey. I am very grateful and thankful to my supervisor, Ahmad Nauman Ghazi for his valuable guidance, support, patience, motivation and unflagging support. I would like to pay my sincere gratitude towards Mr. Toiwo Cedergren and Mr. Mattias Svensson, employees at Ericsson, Karlskrona for helping me. I would like to thank my roommates, college friends and many interesting people I have come across in my life.

CONTENTS

ABSTRACT **3**

ACKNOWLEDGEMENTS: **4**

CONTENTS **5**

FIGURES **7**

TABLES **8**

1 INTRODUCTION **9**

1.1 PROBLEM DESCRIPTION: 9

2 BACKGROUND AND RELATED WORK **12**

2.1 BACKGROUND 12
2.1.1 Heterogeneous systems: *12*
2.1.2 Testing Techniques: *13*
2.2 RELATED WORK 14

3 RESEARCH METHODOLOGY **16**

3.1 AIMS AND OBJECTIVES: 16
3.2 RESEARCH QUESTIONS: 16
3.3 EXPECTED OUTCOMES 16
3.4 RESEARCH METHODOLOGY 17
3.4.1 Motivation for Selecting Systematic Literature Review *18*
3.4.2 Motivation for selecting Interviews *19*
3.4.3 Data Analysis *20*

4 SYSTEMATIC LITERATURE REVIEW **21**

4.1 SLR STRUCTURE: 21
4.2 SEARCH STRATEGY 22
4.3 INCLUSION AND EXCLUSION CRITERIA 23
4.4 QUALITY ASSESSMENT CRITERIA: 24
4.5 DATA EXTRACTION STRATEGY 25
4.6 SLR EXECUTION 26

5 INTERVIEWS: **28**

6 RESULTS AND ANALYSIS **32**

6.1 SLR RESULTS 32
6.1.1 Overview of Primary Studies *33*
6.1.2 Analysis of Literature based on the Third party components in Heterogeneous Systems *35*
6.1.3 Analysis of Literature based on the Challenges faced while using third party components *37*
6.1.4 Analysis of Literature based on the Testing of heterogeneous systems with third-party components: *39*
6.1.5 Discussions and Conclusions for SLR: *40*
6.1.6 SLR Validity Threats: *41*
6.2 INTERVIEWS RESULTS: 42
6.2.1 Codes and Themes for RQ1: *46*
6.2.2 Codes and themes for RQ2: *47*
6.2.3 Codes and Themes for RQ3: *48*
6.2.4 Codes and themes for RQ4: *50*

7 DISCUSSIONS AND VALIDITY THREATS 53
7.1 DISCUSSIONS 53
7.1.1 Discussion for RQ1: *53*
7.1.2 Discussion for RQ2: *53*
7.1.3 Discussion for RQ3 and RQ3.1: *55*
7.1.4 Discussion for RQ4 ..... *56*
7.2 THREATS TO VALIDITY 57
7.2.1 Internal validity threats: *57*
7.2.2 External validity threats: *57*
7.2.3 Construct Validity: *57*
7.2.4 Conclusion Validity: *57*
8 CONCLUSION 59
8.1 CONCLUSIONS 59
REFERENCES 60
APPENDIX A: QUALITY CRITERIA 64
APPENDIX B: INTERVIEW QUESTIONNAIRE: 65

FIGURES

Figure 1 Component-Based Approach 9
Figure 2 Heterogeneous system Architecture 12
Figure 3 Research Design 17
Figure 4 Expected outcomes 18
Figure 5 SLR Steps 21
Figure 6 SLR process 27
Figure 7 Interview Questionnaire 29
Figure 8 Year wise distribution of primary studies 34
Figure 9 Classification of primary studies based on contribution 35
Figure 10 Classification based on third party components 36
Figure 11 Classification based on challenges 38
Figure 12 Themes 43

TABLES

Table 1Testing techniques 14
Table 2 PICO strategy 22
Table 4 Quality Criteria 25
Table 5 Data Extraction Form 26
Table 6 Interviewee details 30
Table 7 Primary studies 33
Table 8 Distribution based on the type of Third Party Component 36
Table 9 Challenges from SLR 37
Table 10 Codes and Themes for interviews 45
Table 11 Codes and Themes for RQ1 46
Table 12 Codes and Themes for RQ 47
Table 13 Codes and Themes for RQ3 49
Table 14 Codes and Themes for RQ4 51
Table 15 Challenges obtained from SLR and Interviews 54
Table 16 Recommendations ..... **Error! Bookmark not defined.**

1 INTRODUCTION

Software evolves exponentially with the passage of time. Immensely colossal systems that evolve results in large development cost, uncontrollable software quality and low productivity [1]. For an organization to be competitive and cost effective, the development process needs to be continuously improved. As reusability of the software is considered to be one of the important factors for improving the development process, component-based software engineering is used in software development. Here, development mainly relies on the software reuse. The criteria followed in component-based approach are as follows: [4]

- Selection of appropriate components for the development
- Agglomerate them with a well-defined architecture

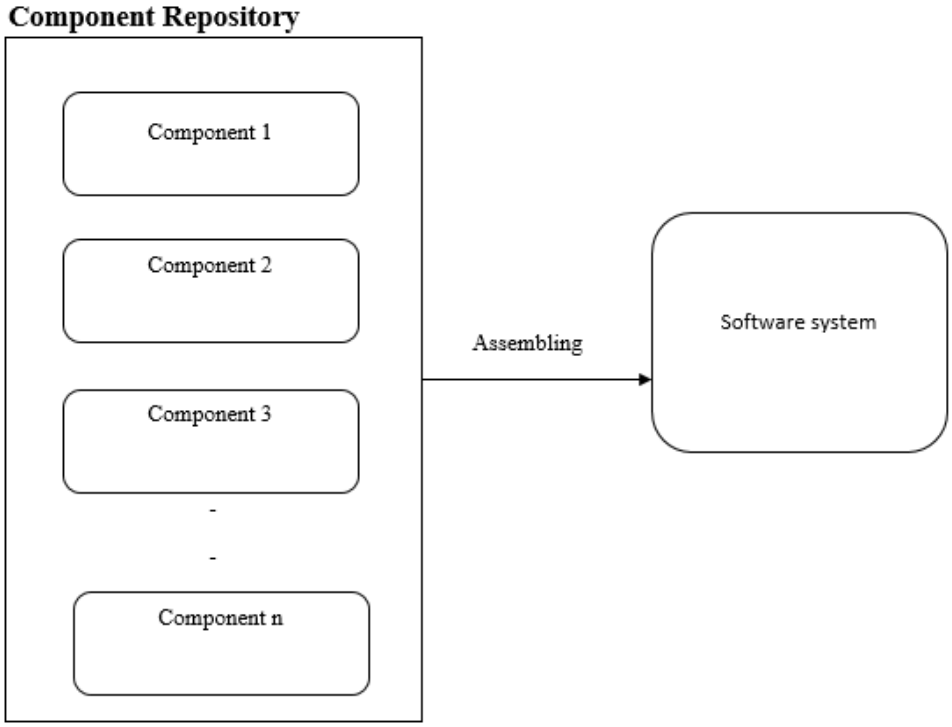

Figure 1 Component-Based Approach

These components based approach led to the development of complex systems called systems of systems. A set of individual systems, which together forms a new system, is called as a system of systems. System of systems has a single authority on different developed components which exhibit heterogeneity[5]. The main characteristics of the system of systems are Degree of centralization, Operational independence of participating system, Diversity of the constituent systems and independent evolution of participating systems [5]. In system of systems sub-systems exhibit heterogeneity in terms of both hardware and software [5].

1.1 Problem Description:

The systems of systems that have heterogeneity on the diversity of the subsystems are called Heterogeneous systems [2]. Heterogeneous systems consist of

functionally independent *n* subsystems where at least one subsystem exhibits heterogeneity with respect to other subsystems [2]. This heterogeneity may occur in different dimensions such as system complexities [6], platforms [7], type of systems [8], system supply (third party software) [9], development process and distribution sites [10].

Third party system supply is one of the dimensions of heterogeneous systems [9], which we intend to investigate in this thesis. The main third party components used in software development are Commercial off The Shelf (COTS)[11], Open Source Systems (OSS)[12] and in-house developed software [13]. "COTS is the software that is sold, licensed or leased to customers without giving the source code" [14]. Gosh and Schmid [15] studied about the robustness of the COTS to the operating system errors. Haddox and Kapfhammer provided an approach in their article [16] to test COTS and has provided developers an improved understanding of the COTS component's behavior within the system. The software systems which provide source code for modification and enhancement are called open source software [17]. These are the third party components mainly used for the development of complex systems. But the question arises in the usage of third-party components in the context of heterogeneous systems, so this study puts emphasis on identifying the third party components usage in the context of heterogeneous systems.

A set of activities performed to identify defects and improve the quality of the product is called testing [18]. This study also focuses on the challenges faced while testing of heterogeneous systems having third party components. As testing of heterogeneous systems is a challenge to the practitioners due to subsystem interaction [19]. As the main challenge arise at the integration level and system level due to change in the configuration of subsystems [19]. Some of the challenges faced while using third party components in heterogeneous systems are lack of documentation [20], interoperability [7], conformance[21], subsystem interaction [20] and version control of the subsystems [22]. This study further focus on exploring the additional challenges faced while using third party components in heterogeneous systems. Thus provides the practitioners a reasonable thought while using third party components for product development.

As per Mathur [23], test design techniques can be mainly classified into Black box testing, white box testing and experienced based testing. Black box testing is specification based testing which uses the external specification of the product such as requirements of the customer, features of the product, design, etc. for testing [23]. In the case of white-box, testing mainly depend on the internal structure of the software and source code [23]. Whereas in the case of experienced based, test design techniques mainly it depends on the experience, knowledge of the tester. This study also focuses on the identification of which test design technique is used for testing of different types of third party components in the context of heterogeneous systems.

The main objectives of this research are to identify different types of third party components, identify challenges faced while integrating third-party components in heterogeneous systems, investigate the difference in test design of different third party components and provide recommendations to the practitioners using different types of third party components. In order to achieve these objectives, a Systematic Literature Review (SLR) was performed to identify third party components, challenges faced while integrating third-party components and test design techniques used for testing heterogeneous systems. Kitchenham guidelines were followed to conduct the SLR. The results obtained from SLR are complemented or validated and supplemented through Interviews. Through Interviews, we provide

guidelines to the practitioners who are using third party components in product development.

Motivation:
As per Keating et al. [24], System supply (Third party systems) is one of the major dimensions of heterogeneous system that is being used in industry and has scope for research. The main challenge that arise while testing of complex heterogeneous systems is due to sub system interaction [19]. Therefore, all the research performed about testing of third party components in the context of heterogeneous systems is somehow related to research in system integration. However, in this thesis, the author has selected only one dimension that introduces heterogeneity in software system (system supply/third party components) and investigated that in detail with a clear focus on the third party components.

2 BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 Heterogeneous systems:

Heterogeneous systems are defined as complex systems comprised of multiple systems (system of systems) where at least one subsystem exhibits heterogeneity with respect to the other systems [25]. In heterogeneous systems, at least one subsystems are functionally independent with respect to the other [2]. Figure 2 shows an example of the heterogeneous systems that involves numerous functionally independent sub-systems. These sub-systems in heterogeneous systems also exhibit heterogeneity in terms of both hardware and software [5].

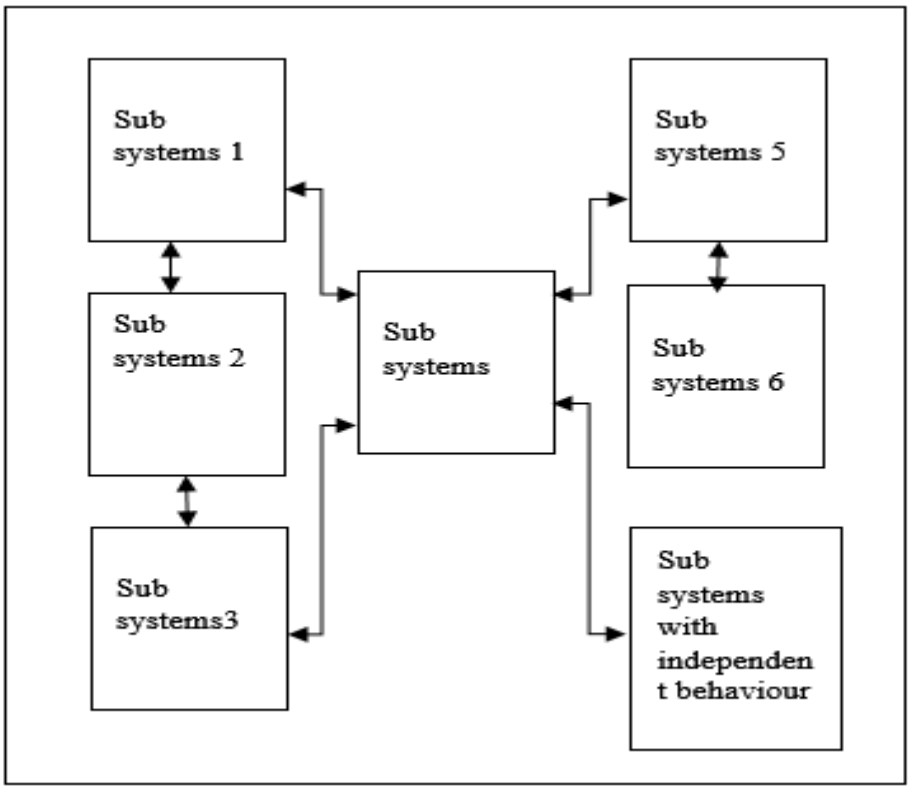

Figure 2 An example of heterogeneous system

A heterogeneous system is defined as a system comprised of multiple sub-systems where at least one sub-system exhibiting heterogeneity with respect to the other sub-systems [24]. These heterogeneous systems exhibit heterogeneity in multiple dimensions. One of these dimensions is heterogeneity due to use of third part components. Therefore, in the context of this thesis, we investigate the class of systems that use different third party components. As there is a lack of research literature with a specific focus on the heterogeneous systems having third party system supply as a heterogeneity dimension. Therefore, in response to the identified research gap we have chosen to investigate the usage and effects of third party system supply in heterogeneous systems.

The main third party components used for product development are commercial of the shelf components [11], Open source software systems [12] and in-house software systems [13].

COTS:
The software that is sold, licensed or leased to the customers without providing rights to access the source is called commercial of the shelf software [14]. Commercial of the shelf software is categorized into two types based on the type of functionalities [11]:

Horizontal**:**

In this type, the functionality provided is not specific to a domain. This type of products can be reused across many different domains.

Example: Web Browsers, Database management systems, graphical user interphases etc.

Vertical*:*

In this type, the functionality provided is specific to the application domain. These can reuse only in the application domain.

Examples: Banking applications, company related software, etc.

OSS:
The software that provides access to the source code and community-source software are called open source software [17]. The main characteristics of OSS are as follows [26]:

Free availability of source code (human-readable instructions)

Providing license for free distribution and free modification

In-house software:
The software that is developed by an organization for purpose of using it within the organization is called in-house software [27].
There is a lack of literature that discuss bout about the third party components used in heterogeneous systems. So author have focused on studying systematically the types of third party components used in heterogeneous systems.

2.1.2 Testing Techniques:

Quality of software is ensured through conducting thorough software testing i.e. conducting testing with all types of inputs. But in real time scenario, it is not ought to be effective in light of the fact that the input values may vary from product to product and in some cases it may be infinite or even may be substantial. Thus, techniques are chosen for a certain scenario based on which tests are designed, executed and analyzed. Moreover, classification and selection of test cases is based on the approach of test case generation.
As indicated by Mathur [23], there are two types of testing techniques, such as Black box and white box. Black box testing techniques doesn't rely on the code and the testing is done for the functionality of the product. When the requirements are vaguely specified, specification based techniques such as equivalence, boundary value analysis, decision table testing, state transition techniques etc. are used. But during white box testing, code is taken into consideration and is being tested i.e. internal structure of product. Structure based techniques come under this method such as decision, condition, statement testing, data flow and control flow techniques etc. To summarize, following table outlines the classification of testing techniques as entailed by [28] [29] [30].

Test Design Techniques	Black-Box	Boundary value analysis, , Cause-effect graphing, Pair-wise, Use-Case testing, State transition testing, Equivalence partitioning, Model-based testing, Random testing
	White-box	Condition coverage, decision coverage, path testing, Loop-testing, statement Coverage testing, Control flow, Data flow, LCSAJ testing
	Defect-based testing	Mutation testing i.e. standard mutation and selective mutation
	Static technique	Informal Reviews, Walkthroughs, Technical reviews, Inspection.
Test Execution technique	Dynamic techniques	Exploratory testing, Error guessing testing, ad-hoc, Negative testing, Fuzz testing
Test result analysis techniques		Active testing, Fault-tree analysis, root cause analysis, Pareto analysis, Cause-effect analysis (Fishbone diagram)

Table 1 Testing techniques

Definitions for each of the testing techniques can be seen in [28]. From Ghazi et.al [24], Testing techniques used for testing of heterogeneous systems having third party components need to be studied systematically. Exploratory testing is mainly used in the industry to test heterogeneous systems [24]. Therefore, to fill the identified gap author have conducted systematic literature review and interviews.

2.2 Related Work

Testing of heterogeneous systems has been a challenge for the testers due to sub-system interaction [19]. Wei Tek Tsai et al.[31] Conducted a study to examine the importance of dynamic WS unit test, even under the circumstances of unavailability of source code and the runtime feature. Based on these results to evaluate the trustworthiness of the WS components, they have proposed three techniques namely specification-based test generation, collaborative testing and group testing. Finally, they have outlined the structural layout of a full SOA-based OS where all the components of a system were implemented as services. Jiantao Pan et al.[32] Conducted an experiment to demonstrate the feasibility of automation process and analyzing failure causes. Results of this experiment provide users a tool to automatically harden the legacy software modules and COTS. Along with this, they have also proven that automatic protection code generation is feasible for COTS software. Jaymie Strecker et al.[33] Conducted an experiment with a set of test-suite and fault-related factors in the GUI testing for discovering the factors that affect fault detection. Finally, their experiment results support the hypothesis that the factors such as detectability with statement

coverage-adequate test suites; event-triple coverage and mutant type can influence the fault detection. From [34] Jaaksi have discussed the experiences of Nokia's product development for using open source components. From the experiences, the author has discussed the challenges and solutions faced while using open source software.

From Briand et.al [35], testing of systems having Commercial of the shelf components (COTS) is a challenge due to the non-availability of source code. An optimization algorithm was proposed to test the COTS components in a deployment environment. Further a case study using mutation analysis was conducted to validate the proposed algorithm. Leonardo Mariani [20] conducted an experiment to describe the main elements of Behavior Capture and Test (BCT). Zheng et.al [36] has proposed an approach I-BACCI (Integrated- Black box approach for component change Identification) and conducted a case study on ABB product that uses a medium scale internal ABB software component. The objective of this approach is to develop a lightweight process for regression test selection for the user functions that use software components when source code is unavailable. Kropp et al. have described the Ballista methodology for scalable, portable, automated robustness testing of commercial of the shelf (COTS) software [37]. Reza et al. proposed a hybrid method based on software fault tree analysis and UML communication diagram to test commercial off the shelf (COTS) software in safety critical systems [38]. So far there is a lack of empirical studies to investigate the test design techniques for testing of heterogeneous systems having third party software. Ghosh and Schimd [15] have proposed an approach and tool for accessing the robustness of Win32 applications for the operating systems failures. Jonge [39] discussed that usage of third party component in product development leads to less control over the product.

To fill the research gap from the literature, author has investigated different types of third party components used in heterogeneous systems, challenges faced while integrating third-party components and test design techniques used for testing of heterogeneous systems having third party components.

3 RESEARCH METHODOLOGY

In this section, we select an appropriate research method to answer the research questions. The objective of this research is to identify challenges faced by testers while testing third party components in the context of Heterogeneous system. To achieve the objective, we need to identify different types of third party components used in heterogeneous systems. Also, the challenges faced while integrating third-party components and test design techniques used for various third-party components.

3.1 Aims and Objectives:

The aim of this study is "to investigate the usage of third party components and their effects on testing of heterogeneous systems". To accomplish this aim the following are the goals that are to be followed:

- Identify different types of third party components.
- Identify challenges faced while integrating third-party components in heterogeneous systems
- Investigate the difference in test design of different third party components
- Identify what the practitioners learn from various third party components

3.2 Research Questions:

RQ1: *What are the different types of third party component used in heterogeneous systems?*

RQ2: *What are the challenges faced while integrating third-party components in heterogeneous systems?*

RQ3: *How does test design differ between different types of third party components?*
RQ3.1: *Why do they differ?*

RQ4: *What are the key recommendations for the practitioners regarding usage of various third party components?*

Motivation:
The main focus of the thesis is on RQ3, RQ3.1 and RQ4. RQ1 and RQ2 were investigated to collect the in-depth evidence from the existing literature to help us understand the context of the study. The research questions investigated in the thesis are specific to heterogeneous systems. In the context of heterogeneous systems, the subsystems are functionally independent and have a challenge of subsystems interaction [19]. Therefore, we expect different results for heterogeneous systems.

3.3 Expected outcomes

The expected outcomes from the research are as follows:

- Different types of third party components used in heterogeneous systems
- Challenged faced while using third party components in the development of heterogeneous systems.

- Test design techniques used for testing of different types of third party components
- Guidelines for the practitioners using third party components in heterogeneous systems

3.4 Research Methodology

An appropriate research methodology needs to be selected to perform the research and answer the research questions. After a rigorous analysis of the available research methods, the author has decided to conduct a Systematic Literature Review (SLR) and Interviews. The research methodology used is a qualitative approach. A Systematic literature review was chosen to answer RQ1, RQ2, RQ3. Interviews were conducted to validate the data obtained for RQ1, RQ2, RQ3, RQ3.1 and to answer RQ4. The research design followed is shown in figure 3. Figure 4 shows the Research method followed for each RQ and expected outcomes for the study.

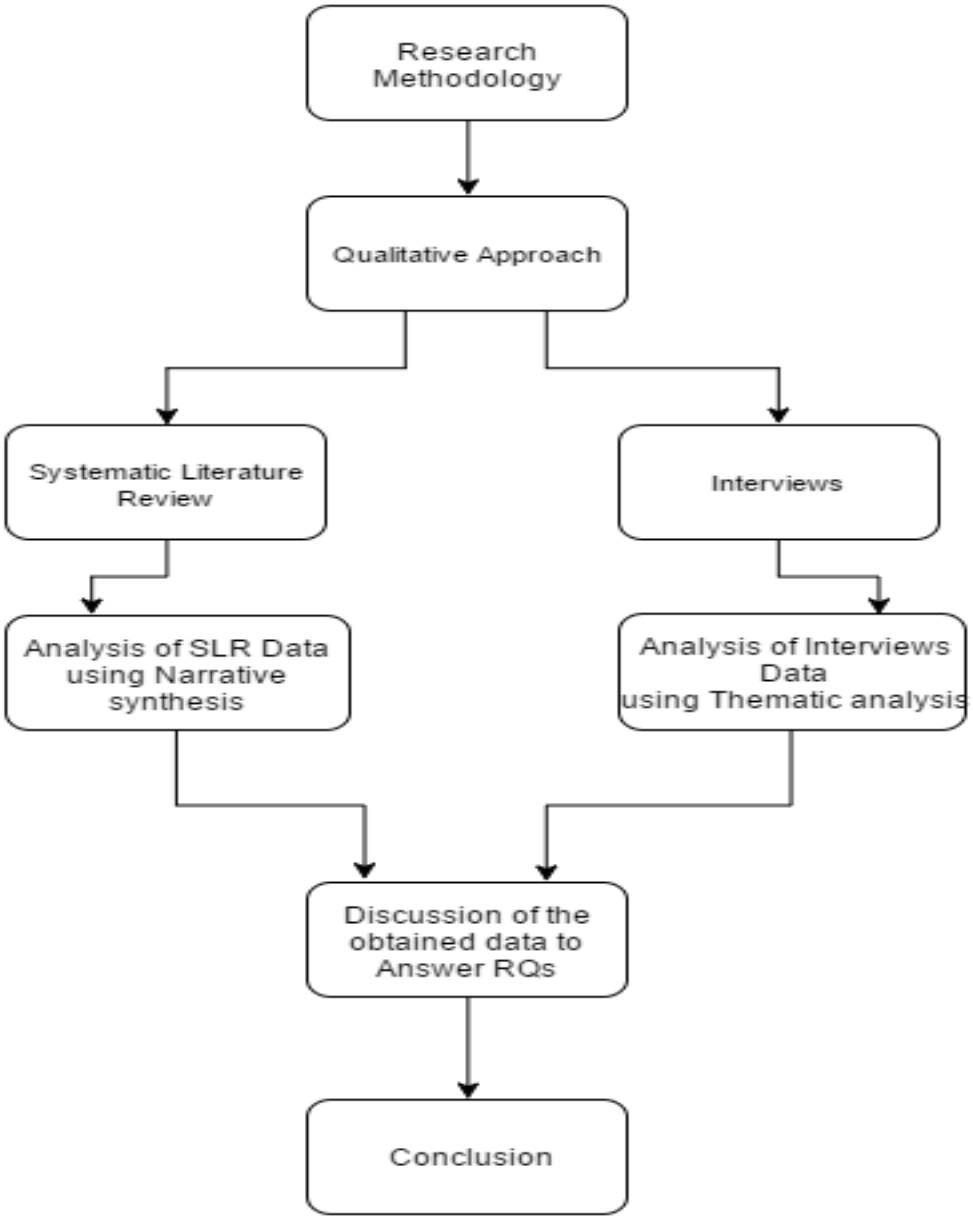

Figure 3 Research Design

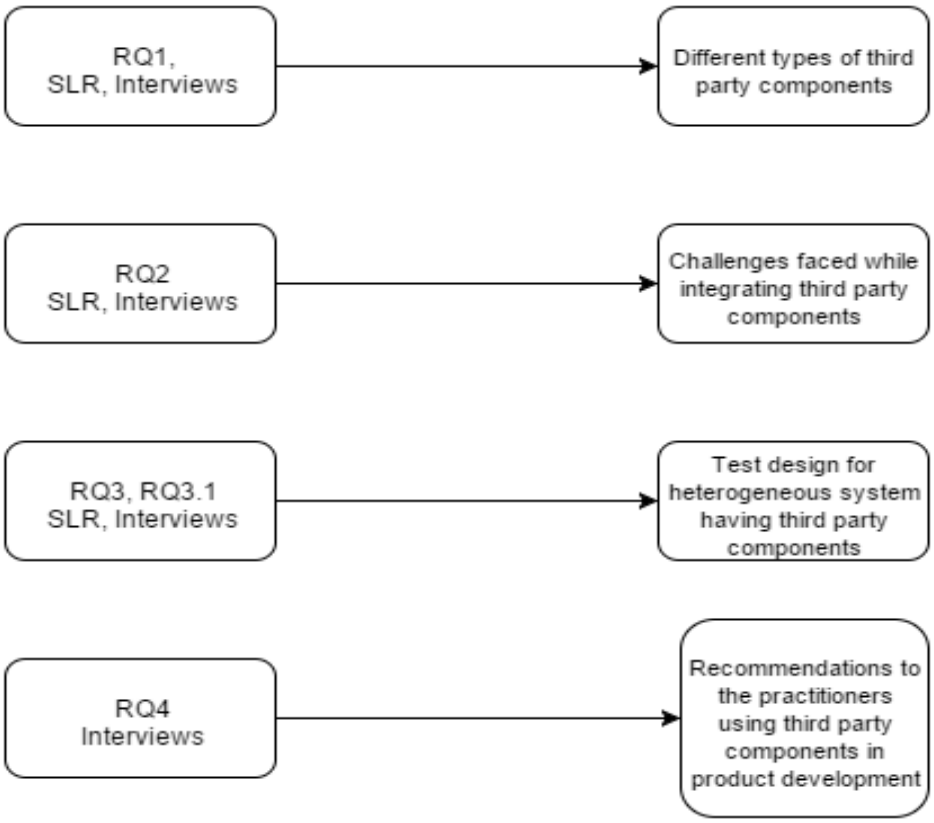

Figure 4 Expected outcomes

3.4.1 Motivation for Selecting Systematic Literature Review

A systematic literature review was performed following the guidelines provided by Kitchenham. A systematic literature review was used to achieve the following research objectives.

- To identify third party components used in heterogeneous systems
- Test design techniques used for testing third party components
- Challenges faced while using third party components in the context of heterogeneous systems.

A systematic literature review endeavor to identify evaluate and interpret research that is present on the particular research question, area or phenomenon [3]. According to Kitchenham, one of the reasons for performing systematic literature review is to summarize the existing evidence concerning treatment or technology [3]. A systematic literature review helps in identifying and aggregating the relevant literature in a particular research area [3]. The motivation for the choice of selecting SLR is that there has been no aggregation of literature available in the area which explains about different types of third party components, challenges and testing techniques used in heterogeneous systems. A systematic synthesis of these aspects helps the practitioners get a clear idea of the third party components, challenges faced and test design techniques used. Therefore a systematic literature review was performed. The findings from literature review can be employed by the researchers to investigate the issues caused due to the third party components, challenges faced and test design techniques used in the context of heterogeneous system. The results obtained from the SLR were used and validated in the interviews. SLR in this research is also used to investigate and identify gaps in the current research. The review is performed using steps and activities in an orderly manner to achieve reliable and valid results.

The alternative methods that can be used to achieve these objectives are Literature review and systematic mapping study. Though systematic mapping study is performed to interpret and summarize the literature, it only deals with the areas whose scope is generous and extensive. The results obtained are ad-hoc in systematic mapping study and cannot be associated as an averment to answer the research questions. So a systematic literature review was chosen instead of systematic mapping research. A literature review is another choice for answering RQ1, RQ2, and RQ3. As the literature review doesn't have any steps for analyzing data and accessing quality, which is possible in SLR. To achieve the research objectives the data obtained need to be reliable, valid and qualitative which is achievable using systematic literature review.

This assertion motivated us for conducting an SLR. A SLR in this research was performed by following the systematic procedures and activities stated in Kitchenham guidelines. In this research, SLR is performed to answer RQ1, RQ2, and RQ3.

3.4.2 Motivation for selecting Interviews

Interviews are performed to complement and validate the SLR results. From interviews, we procure guidelines to the practitioners who are using third party components in heterogeneous systems. A systematic way of talking and listening to the people's opinion is called interviews [40].
Motivation for choice of selecting interviews as a research instrument [41]

- Need to attain personalized views of the practitioners using third party components in heterogeneous systems to validate and complement the results obtained from the SLR.
- To provide a set of guidelines to the practitioners using third party components.

Interviews are classified into three types based on the questionnaire [42][43][44]:

- **Structured Interviews:** The questionnaire posed in this type of interview is closed ended. The responses obtained to the posed questions are mainly a word that signifies the intension of the interviewee.
- **Semi-Structured Interviews:** In this type of interview the questionnaire is designed in such a way that the questions posed are both open-ended and closed ended. In this type, the interviewer has complete control over the process. The main purpose of this kind is to complement the existing data and further supplement the results.
- **Unstructured Interviews:** In this type of interview, the interviewee and interviewer are held in a discussion where the interviewee discuss the topics posed and interviewer records the relevant data. Here the interviewee has more control rather than the interviewer.

In this study, semi-structured interviews were followed to answer the research questions. Semi-structured interviews have both open ended and close ended questions[44]. Questionnaire is formulated by taking inputs from the SLR and guidance from the supervisor. A fixed set of questions is formulated to conduct pilot interviews to check the efficacy of the questionnaire based on the understandability. We use Time glass model [44] to conduct an interview session. In this model, open questions were posed at the beginning and strengthens the structure in the middle and opens up again towards the end of interview [44]. The communication channels mainly used to perform interviews were Skype, team viewer and face-to-face meetings. Convenience sampling was performed to select the samples for the interviews. The samples used for conducting interviews were

developers and tester who are working with heterogeneous systems comprising of third party components. The data from the interviews will be recorded and transcribed into text. These data transcripts were reviewed by the interview subjects, which help to get clarification and expansion of answers.

3.4.3 Data Analysis

This section discusses the data analysis methods used in the research.

3.4.3.1 Narrative synthesis

The data extracted from the SLR process is analyzed using narrative synthesis approach. Narrative synthesis is the process to find out the summary of findings of primary studies [45]. This approach is of the most commonly used approach to analyze the data obtained from systematic reviews [46]. As this type of approach is also helpful to analyze reviews of both quantitative and qualitative research [45], so the data extracted from SLR was analyzed using narrative synthesis approach.

3.4.3.2 Thematic analysis

Thematic analysis was used to analyze the data obtained from the interviews. The method used for Identifying, analyzing and reporting patterns (themes) within data is called thematic analysis [47]. This approach is mainly used to synthesize the qualitative data. Grounded theory is one of the alternative choice for analyzing the qualitative data. From Wohlin and Aurum [48], It is hard to perform grounded theory by novice researchers. As the data obtained from the interviews were qualitative, so the thematic analysis is used to analyze the data.

3.4.3.3 Comparative analysis:

Comparative analysis is the method used to compare the elements to discover the difference between them [49]. It is used to apprehend the likeliness and contrast between the entities [50]. So to know the likeliness and contrast between the results obtained from SLR and Interviews regarding Third party components usage, challenges faced while integrating third-party components and test design techniques used for testing of heterogeneous systems having third party components, we have chosen qualitative comparative analysis. We have also used comparative analysis in discussions to interpret the findings.

4 SYSTEMATIC LITERATURE REVIEW

4.1 SLR structure:

A systematic literature review was performed by planning and designing the review protocol. In this study, SLR was conducted by following Kitchenham guidelines [3]. As there is less literature in the research area, the scope for performing Systematic literature review is high. SLR provides valid, reliable and efficient results from the available literature [3]. So it was chosen to answer the RQ1, RQ2, and provide inputs to RQ3. Kitchenham guidelines were followed to perform the Systematic literature review [3].
To conduct a systematic literature review, the following steps were followed: Formulated search string from the research questions, document retrieval, defining inclusion and exclusion criteria, finalizing the primary studies, defining quality assessment criteria to assessing the primary studies, data extraction and analyzing the data. These steps are shown in the figure 5:

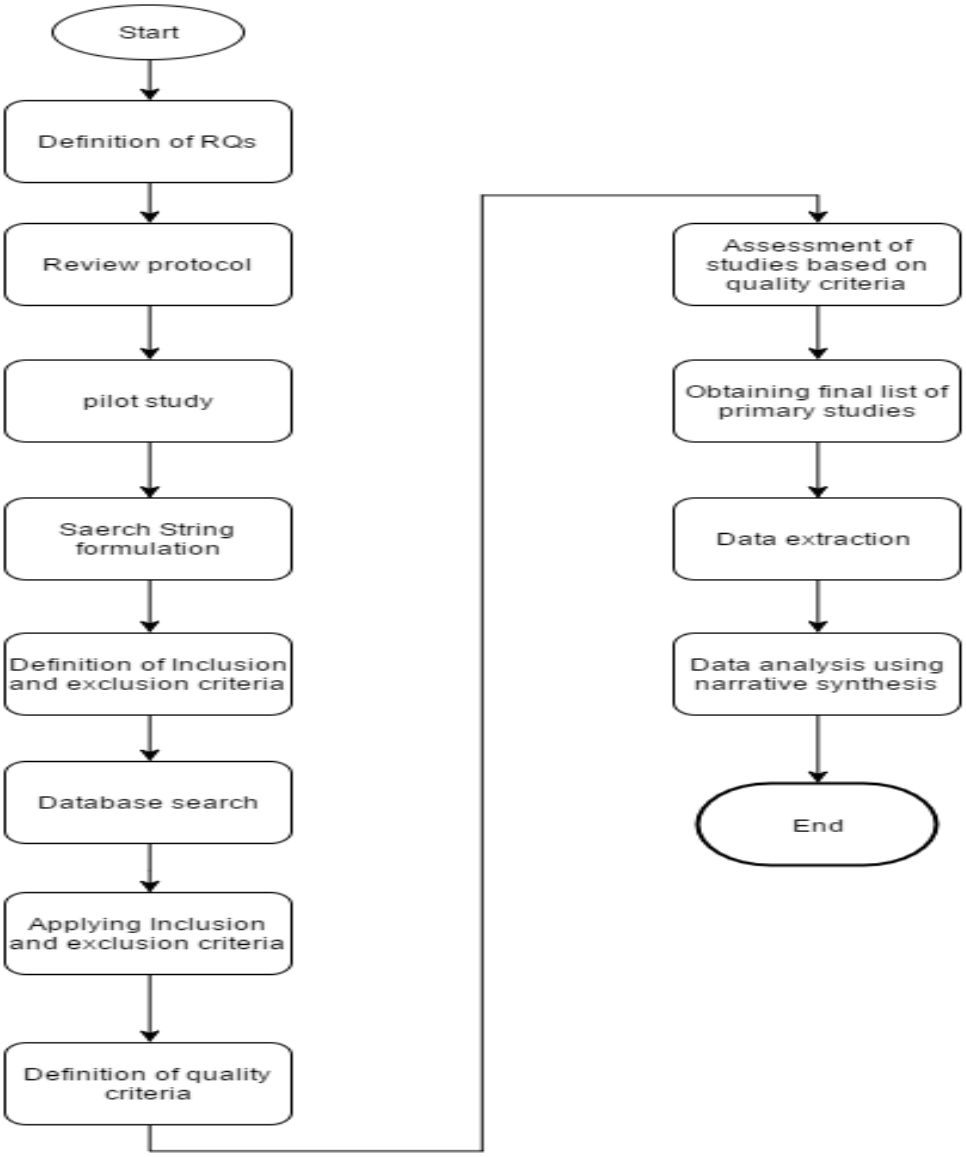

Figure 5 SLR Steps

4.2 Search strategy

The search strategy for this SLR is based on the following criteria:

- Identification of keywords
- Search string formulation
- Determining the resources

For this research, keywords for the search string were identified based on PICOC criteria: Population, Intervention and Outcomes [3]. In this study Population are the studies that focus and reflect their ideas on heterogeneous systems, multi-variant systems. Intervention is the third party components that are used in heterogeneous systems such as Commercial of the shelf (COTS), Open source software (OSS) and in-house software, challenges and test design techniques that are used for testing these systems. The outcome is the third party components, challenges, and test design techniques. The context of this study is software engineering, component-based software engineering. PICO strategy is shown in table 2.

Population	Heterogeneous systems, system of systems, multi-variant systems
Intervention	Third party components, challenges, test design techniques
Context	Software Engineering, Component based software engineering
Outcomes	Third party components, COTS, OSS, in-house software, test design techniques, challenges

Table 2 PICO strategy

The following are the steps followed for the formulation of search:

- By following the PICOC criteria, imperative terms were identified and taken from the research questions RQ1, RQ2, and RQ3.
- A pilot search was conducted to identify the primary keywords.
- Synonyms were identified and listed for the search terms.
- Boolean operators “OR” and “AND” were used. “OR” was used to separate the synonyms and operator “AND” was used to separate relevant search terms.

The keywords for this study were also formulated using PICOC criteria: Heterogeneous systems, Component based software engineering, Third party components, Challenges, Test design techniques.

Following research databases are used for systematic search:

- Inspec
- Scopus
- IEEE Xplore

The motivation for the choice of selecting the listed databases is that IEEE Xplore covers most of the relevant papers present in the particular area. Databases Inspec and Scopus cover the articles from ACM Digital Library, science direct and SpringerLink. Therefore, selecting these databases we covered most of the literature in this area of research. Articles, which are technically peer-reviewed such as journals articles, workshop publications, and conferences papers, were selected.

The search string was formulated from the search terms and keywords identified from the research questions. Synonyms for the keywords were chosen and combined with the search string using Boolean operators. The search string was checked for improvements by performing a pilot search. The table 3 shows the search string used for performing the electronic search on the following databases:

Inspec	(((software testing)OR(challenges)OR(test design techniques))AND(((commercial of the shelf) OR(COTS)OR(open source software)OR(OSS)OR(in house software)OR(in-house software)OR(components))AND((heterogeneous systems)OR(multi variant)OR(multi-variant))) WN KY)
Scopus	((“software testing” OR ”challenges” OR ”test design techniques”) AND ((“commercial of the shelf” OR COTS OR “open source software” OR OSS OR “in house software” OR “in-house software” OR “components”) AND (“heterogeneous systems” OR “multi variant” OR “multi-variant”)))
IEEE Xplore	((“software testing” OR ”challenges” OR ”test design techniques”) AND ((“commercial of the shelf” OR COTS OR “open source software” OR OSS OR “in house software” OR “in-house software” OR “components”) AND (“heterogeneous systems” OR “multi variant” OR “multi-variant”)))

Table 3 Search String

4.3 Inclusion and Exclusion Criteria

The initial studies that were identified from the database search were finalized by following the inclusion and exclusion criteria. The aim of defining inclusion and exclusion criteria is to eliminate the studies that are irrelevant and studies that are duplicates obtained from the initial search. The inclusion and exclusion criteria for the study are as follows:

Inclusion criteria:

- ***IC1:*** *Studies that are published in English.*
- ***IC2:*** *Studies published from the year 1999 to 2015.*
- ***IC3:*** *Studies that are related to heterogeneous systems having third party components were included.*

- ***IC4:*** *Studies that are available in full text were included.*
- ***IC5:*** *Studies that are related to testing of third party components.*
- ***IC6:*** *Peer review studies are included (journal articles, conference papers and work shop paper)*
- ***IC7:*** *Studies that focus on challenges faced while integrating third party components in Heterogeneous systems.*

Exclusion criteria:

- ***EC1:*** *Studies that doesn't belong to component-based software engineering.*
- ***EC2:*** *Studies that are not published in English*
- ***EC3:*** *Studies that doesn't related to search terms.*
- ***EC4:*** *Studies with duplicates at the database level were excluded based on title and abstract.*
- ***EC5:*** *Studies that describe about third party components but not in heterogeneous systems were excluded.*
- ***EC6:*** *Studies that are not peer-reviewed such as book chapters, books were excluded.*

4.4 Quality Assessment Criteria:

In this study, a quality assessment criterion was formulated to analyze the quality of the primary studies. Quality assessment criteria were formulated by following the kitchenham guidelines [3]. The quality criteria are performed to minimize the bias and maximize the internal and external validity threats for the primary studies [3][51] . The quality criteria for the study is shown below:

Quality Criteria	Score (0 or 1)
QC1: *Does the reader able to understand the aims and objectives of the research?*	
QC2: *Does the study performed clearly describes third party components used in heterogeneous systems?*	
QC3: *Does the study state the challenges faced while using third-party components?*	
QC4: *Does the data collection method used in the study clearly described?*	
QC5: *Does the data collected relate to the aim of the research?*	
QC6: *Does the data collected is analyzed based on the method described?*	

QC7: *Does the research conducted contribute to the software engineering field?*	
QC8: *Is the conclusion clearly stated and related to the aims and objective of the research?*	

Table 4 Quality Criteria

The scores for each question is "0" or "1". "0" indicates "NO" that it is not done or not used or not discussed in the study and "1" indicated the opinion "YES". The studies with the score more than 4 are qualitative. The detailed assessment of the quality for the primary studies is shown in Appendix A. The papers with higher quality discuss about third party components and their usage in heterogeneous systems and challenges faced while integrating these components. Papers of lower quality miss to provide the comparison of the third party components usage and miss to provide analysis and validation techniques. From the criteria QC2, QC3 the studies that are most relevant to the research area were categorized.

4.5 Data Extraction Strategy

From the inclusion and exclusion criteria primary and quality criteria primary studies were filtered and selected. Data extraction process for this research is followed in order to specify the type of data that is obtained through primary studies. In order to obtain relevant data to answer RQ1, RQ2, RQ3, the following data extraction checklist is followed. The checklist for the study is as follows:

SNO		Comments
D1	Study Identifier	
D2	Bibliographic Reference	Title of the paper, Author names
D3	Type of Article	Journal, conference etc.
D4	Type of Research method used	Case study, experiment, survey
D5	Any Explicit research question discussed	If "YES"(what are the questions) OR "NO"
D6	Research domain	Research area
D7	What type of the third party component discussed in the paper?	E.g.: Cots, OSS, In-house software
D8	Are there any Impacts of using third party software on testing of heterogeneous systems?	If Yes (explain) OR NO

D9	Are there any challenges faced while integrating third-party software in heterogeneous systems?	If Yes (explain) OR NO
D10	What solution (i.e. method/tool/technique/framework/model) is proposed?	If Yes (explain) OR NO
D11	Are there any Testing tools discussed for the testing of Heterogeneous systems that have Third party components?	If Yes (explain) OR NO
D12	Explicit test design techniques used for testing of heterogeneous system that contain third party components?	If Yes (explain) OR NO
D13	Validation of Data done?	If Yes (explain) OR NO
D14	Results of the study	Explain the results

Table 5 Data Extraction Form

From the above check list, D7, D9, D11, D12 were used to extract data for RQ1, RQ2 and RQ3. D13, D14 were used to find the rigor of the studies.

4.6 SLR Execution

An automated database search was performed to obtain relevant literature. The databases that were used for performing this search are IEEE, Inspec, Scopus, and Springer. The search string used for the process is shown in the table 4 search string. After completing the automated database search, duplicates that are at the database level were removed. The studies that are in English were chosen and selecting studies that were published from 1999 to 2015 were chosen. Studies that are selected from these criteria were filtered based on the title and abstract and Studies that are in full text were selected.

The search results obtained from the database search after application of the search string were 5212 articles. After applying the inclusion, exclusion criteria and removing the duplicates at the database level the articles obtained were 547 articles. The articles available in full text were selected. After reading the title and abstract to the remaining articles were filtered to 162. These 162 articles were filtered by reading introduction and conclusion to 92. 31 relevant studies were selected after reading the full text.

The figure below shows the primary study selection process.

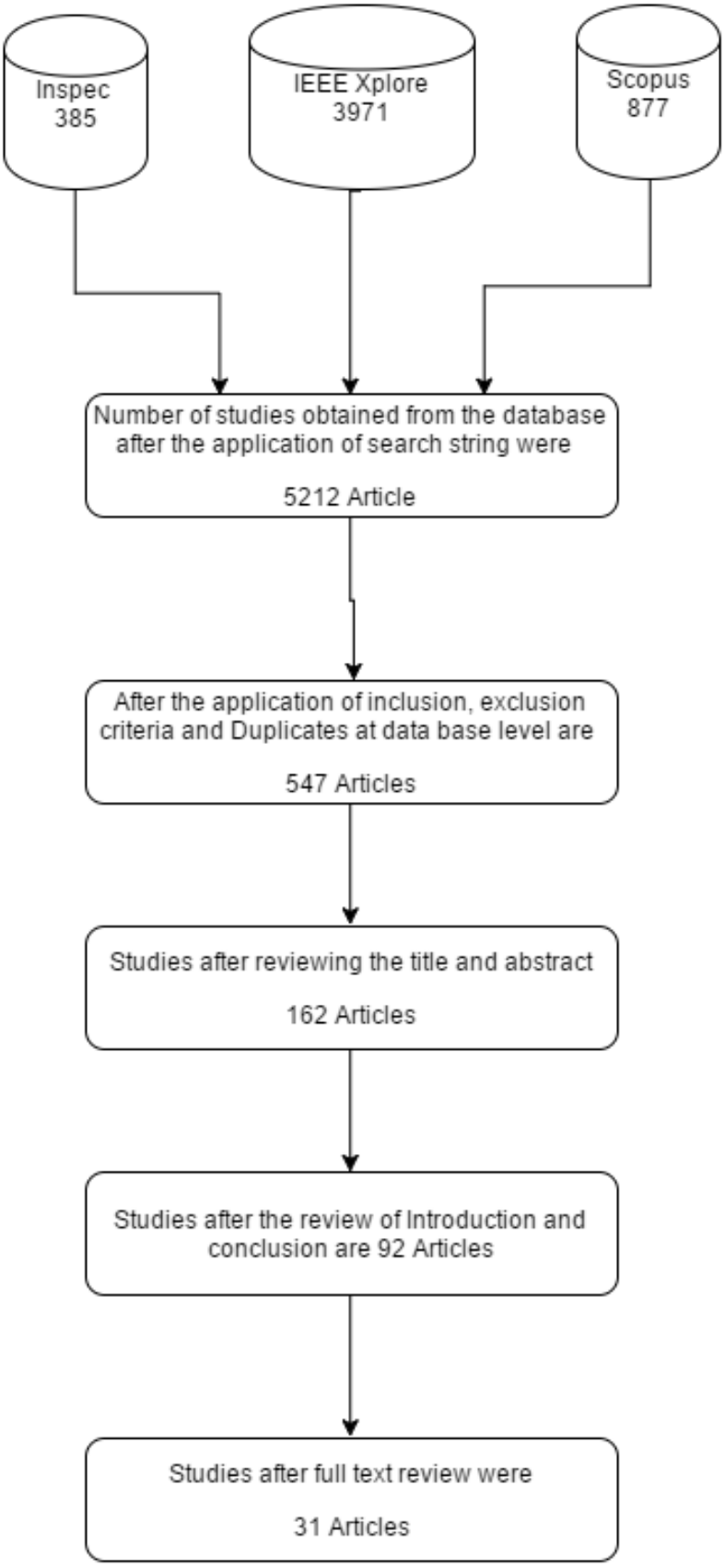

Figure 6 SLR process

5 INTERVIEWS:

The research instrument mainly used in this study for collecting data is an interview. “An interview is an interchange of views between two or more people on a topic of mutual interest” [43]. A systematic way of talking and listening to people is called as an interview [40]. Interviews were conducted to complement and validate the SLR results. In addition to collect recommendations for the practitioners using third party components.

Interview Conduction:

In this study, a semi-structured qualitative Interview approach was followed. A semi-structured interview has both open ended and closed ended questions where the interviewer takes full control of the interview and extracts the required information for the research [44]. The important stages for interview conduction are as follows:[52][41]

- Thematizing
- Designing
- Interviewing
- Transcribing
- Analysing

Thematizing:

The rationale for conducting the interview is discussed in this stage. The reason for conducting an interview is to validate the third party components used in heterogeneous systems, challenges faced while integrating third-party components, test design techniques used for testing of third party components and collect data to provide recommendations to the practitioners. The author of the research searched for the software industries who work on heterogeneous systems using third-party components. Then the practitioner using third party components in heterogeneous systems were sent an email about the topic, design, expected outcomes and request for an appointment regarding the interview.

Designing

An interview must be designed to gather the required information to answer the research question to accomplish the research. As we have followed a semi-structured approach, we develop a questionnaire which is used as a base for the interview. In the interview process, the questions posed were based on the discussions performed by the interviewee. The questionnaire used for interview is developed from the SLR results and research questions. The keywords from research questions RQ1, RQ2, RQ3, RQ3.1 and RQ4 were used in the formulation of interview questionnaire. The figure 7 shows the process for conductions of interview questionnaire.

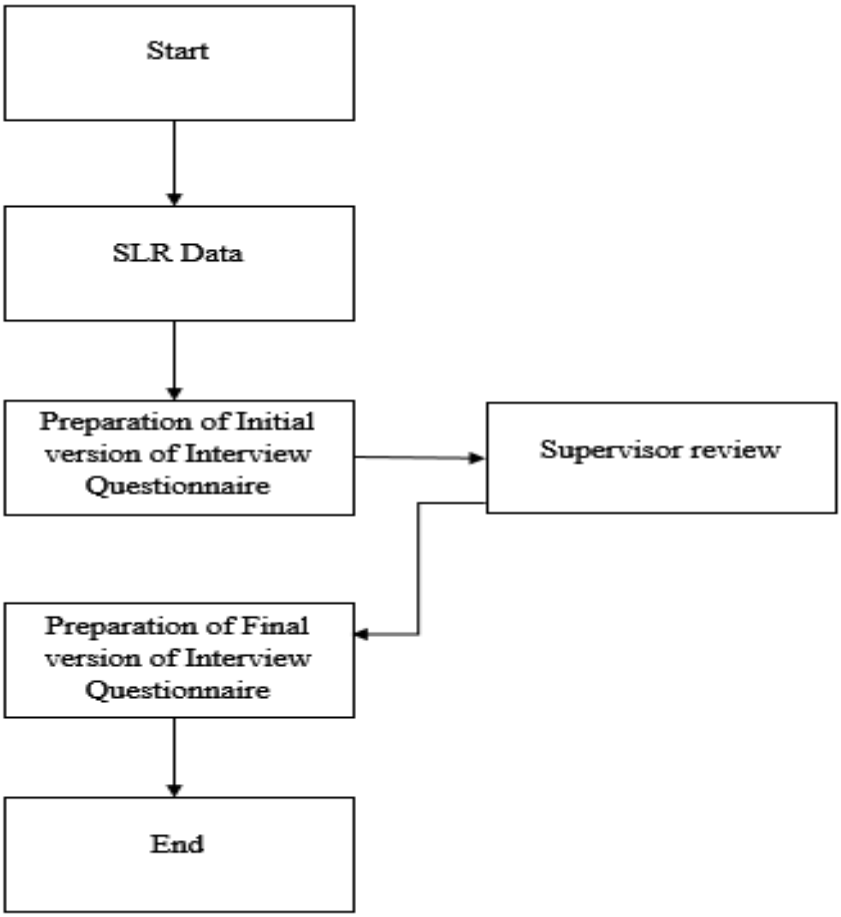

Figure 7 Interview Questionnaires

After conduction of each interview, the questionnaire is modified based on the data collected from the interviews. The interview questionnaire for the research is shown in the appendix. The order of the questions in the interview are not fixed and are modified based on the situation [40].

The interviewees were selected based on the research area that is usage of third party components in heterogeneous systems. Convenience sampling was used to select the interviewees. A practitioner who have experience in the field of heterogeneous systems using third-party components were chosen. As the research questions focus on the challenges faced while integrating third-party components and test design techniques used for testing of third party components, testers and developers were mainly focussed. Once the practitioners were identified by the author, an electronic email was sent to the practitioners to have a notification of approval by discussing the research area and schedule of the interview. Table 6 shows the profiles of the interviewee.

id	Interviewee details
INT1	• System Engineer • 10 years' experience and 2 years' experience in the field of third party component usage

INT2	• Test Analyst • 10 years of experience and 5 years of experience in the field of third party component usage
INT3	• Team lead • 8 years of experience and 3 year of experience in the field of third party component usage
INT4	• System manager and third party component coordinator • 15 years' experience and 8 years of experience in the field of third party component usage
INT5	• Software developer • 8 years of experience and 4 year of experience in the field of third party component usage
INT6	• Test Manager • 10 years of experience and 5 year of experience in the field of third party component usage
INT7	• Test Analyst • 10 years of experience and 3 year of experience in the field of third party component usage
INT8	• Third party coordinator • 10 years of experience and 5 year of experience in the field of third party component usage
INT9	• Test Analyst • 7 years of experience and 2 years of experience in the field of third party component usage
INT10	• Test Manager • 8 years of experience and 2 years of experience in the field of third party component usage
INT11	• Component Integrator • 12 years of experience and 10 years of experience in the field of third party component usage
INT12	• Software quality assurance tester • 8 years of experience and 4 years of experience in the field of third party component usage

Table 6 Interviewee details

Interviewing:

The author of the research has initially planned for 14 interviews, but due to the time constraint and unavailability of the practitioners 12 potential interviews were conducted. The interview time was initially planned for 30- 40 min but after proceeding with the interview, it approximately took 1 hour. Most of the interviews were planned online using Skype. The schedule of the meeting was made based on the comfort of the interviewee. The author studied the background of the interviewee to get more interaction in the interview session and use time in an efficient manner. A time glass model was followed to conduct an interview [44]. In this model open questions are posed at the beginning and strengthens the context in the middle and opens up again towards the end of the interview [44]. The data obtained from the interviewees were recorded using recording tools. These recordings were further used for the transcription of the interview.

Transcribing:

In this phase, the thoughts, speech or gathered data from the interview were converted into written format. Author has used express scribe transcription software to transcribe the data obtained from the recordings. The transcripts obtained from the interview were further validated by the interviewee to avoid ambiguity in the results.

Analysing:

In this phase, the transcribed data obtained from the interviews were analyzed to extract required data based on the research study. The relevant codes from the transcribed data were identified by systematically analyzing the recorded tapes and notes obtained from the interviews.

6 RESULTS AND ANALYSIS

6.1 SLR Results

Total 31 primary studies were relevant to this research after performing a systematic literature review. The primary studies obtained from the SLR with the IDs are shown in the table 7:

S.no	Primary studies	Identifier	reference
1	Anup K. Gosh and Matthew Schimd, 1999	[S1]	[15]
2	Barry Boehm, Winsor Brown and Richard Turner, 2005	[S2]	[53]
3	Oleksiy Mazhelis, Pasi Tyrväinen and Lauri Frank, 2013	[S3]	[54]
4	John brondum and Liming zhu, 2012	[S4]	[55]
5	Rui Zhou, Peter C. Sanza and Nikita A. Visnevski, 2006	[S5]	[56]
6	M Azmoodeh, N Georgalas and S Fisher, 2005	[S6]	[57]
7	Thomas Osterlie and Alf Inge Wang,	[S7]	[58]
8	Scott Hissam and David Carney, 1999	[S8]	[59]
9	Dong Ho Kim1, Hyun Seung Son, Woo Yeol Kim, and Robert Young Chul Kim, 2012	[S9]	[60]
10	Jeff Tian, 2011	[S10]	[61]
11	Weipeng Jing, Zhibo Wu , Hongwei Liu & Jian Dong, 2011	[S11]	[62]
12	Dasha Hu, Zhishu Li, Yunzhu Ni, 2009	[S12]	[63]
13	KM Gee, 2000	[S13]	[64]
14	Andr´e Takeshi Endo1, Adenilso da Silva Sim˜ao1, 2008	[S14]	[65]
15	Wei-Tek Tsai, Yinong Chen and Ray Paul, 2005	[S15]	[31]
16	Ahmad Nauman Ghazi, Kai Petersen, and Jürgen Börstler, 2015	[S16]	[24]

17	Pekka Mäki-Asiala, Mari Matinlassi	[S17]	[66]
18	Erik Putrycz1, Marius Slavescu2	[S18]	[22]
19	Michael A Mott and Faridullah Khan, 2000	[S19]	[67]
20	Jiang Zheng1, Brian Robinson2,Laurie Williams1,Karen Smiley2, 2006	[S20]	[36]
21	Muhammad Naeem Irfan, 2009	[S21]	[68]
22	Leonardo Mariani, Sofia Papagiannakis and Mauro Pezz`e, 2007	[S22]	[69]
23	Jiantao Pan, Philip koopman and Daniel Siewiorek, 1999	[S23]	[32]
24	Jaymie Strecker and Atif M Memon, 2008	[S24]	[33]
25	Sami Beydeda, 2006	[S25]	[70]
26	Leonardo Mariani, 2004	[S26]	[20]
27	Wen-Li Wang a, Dai Pan b, Mei-Hwa Chen b, 2006	[S27]	[71]
28	Said Nabi and M. N. A. Khan	[S28]	[72]
29	King Chun Foo1, Zhen Ming (Jack) Jiang and Bram Adams	[S29]	[73]
30	Swapna S. Gokhale and Kishor S, 2006	[S30]	[74]
31	Jung-Hua Lo, Sy-Yen Kuo, Michael R. Lyu, and Chin-Yu Huang. 2002	[S31]	[75]

Table 7 Primary studies

6.1.1 Overview of Primary Studies

From the SLR, 31 relevant studies were identified by the inclusion criteria, exclusion criteria, title and abstract review and full text review. The studies published between 1999 and 2015 were considered. The figure 8 shows the distribution of primary studies based on the year of publication. It was observed that most of the research was conducted on 2006 as 10 of the primary studies were published in that year. And 6 studies were published between 1999 and 2005.

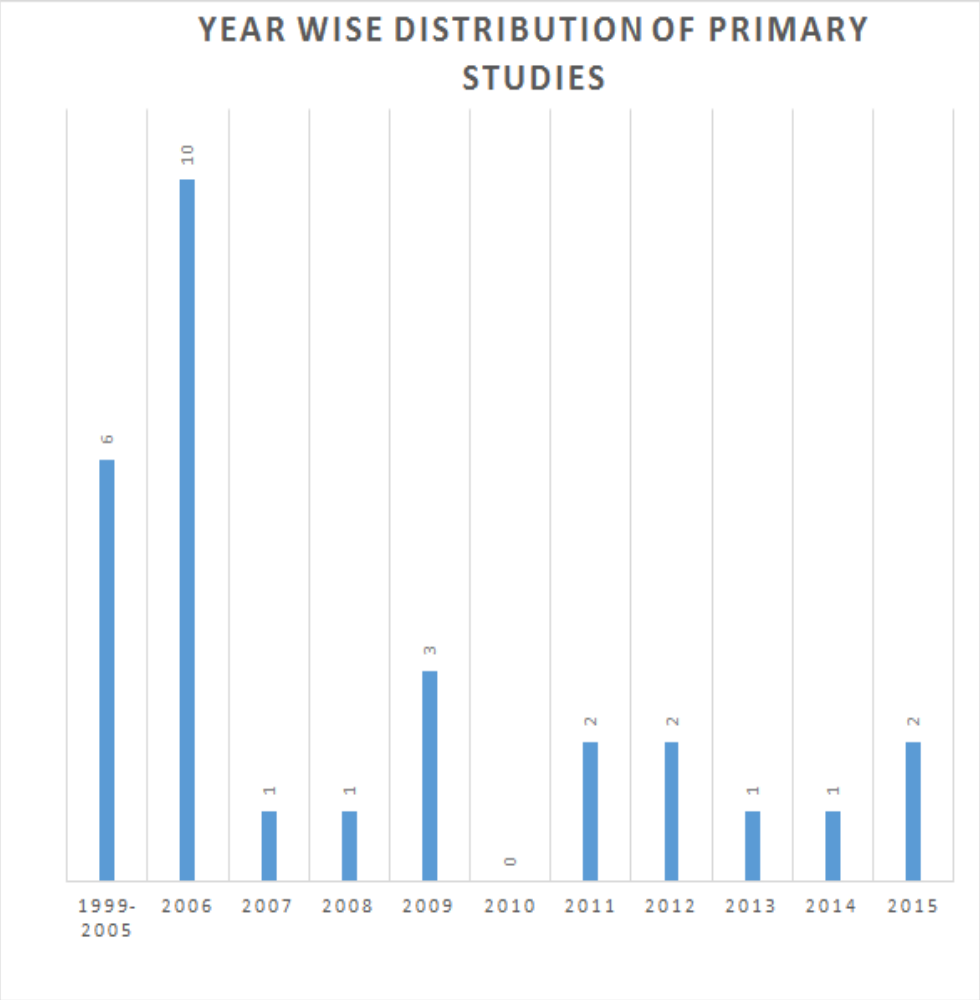

Figure 8 Year wise distribution of primary studies

Primary studies were reviewed based on the research questions and were classified based on the contribution to the research. The main contribution to the research based on the Research questions are as follows:

- Third party components used in Heterogeneous Systems
- Challenges faced while integrating third-party components to heterogeneous systems
- Test design techniques used for testing of heterogeneous systems with the third party component dimension.

Among 31 primary studies, 31 studies discussed the usage of third party components in heterogeneous systems. 26 studies discussed the challenges while third party usage.17 studies discussed the test design techniques used for testing of third party components. Among the 31 studies, 16 studies discussed the third party component usage, challenges faced and test design techniques. Figure 9 shows the classification of primary studies based on contribution.

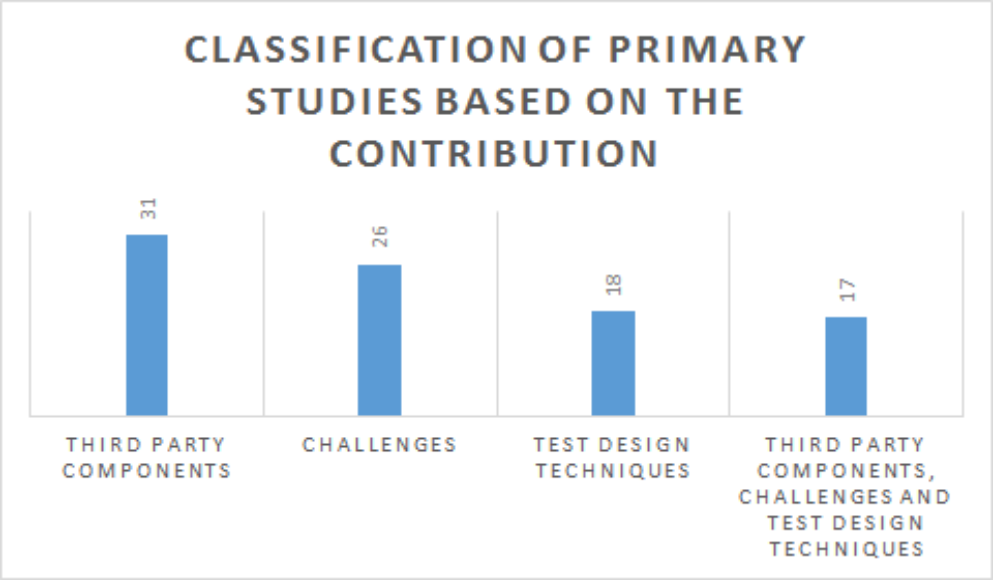

Figure 9 Classification of primary studies based on contribution

6.1.2 Analysis of Literature based on the Third party components in Heterogeneous Systems

In this section, the primary studies were analyzed based on the third party components used in heterogeneous systems. The primary focus of this study is to investigate the third party components used in heterogeneous systems. The studies that discussed about the third party components in heterogeneous systems are [S1] [S2] [S3] [S4] [S5] [S6] [S7] [S8] [S9] [S10] [S11] [S12] [S13] [S14] [S15] [S16] [S17] [S18] [S19] [S20] [S21] [S22] [S23] [S24] [S25] [S26] [S27] [S28] [S29] [S30] [S31].

From the obtained literature, the identified third party components used in heterogeneous systems are Commercial of the Shelf (COTS), Open Source Software (OSS) and In-house software. Among 31 primary studies, which discuss the usage of third party components in heterogeneous systems, 26 studies discussed the Commercial of the shelf components. 9 studies discussed the open source software and 2 studies discussed the in-house software. The figure 10 shows the classification of studies based on the usage of third party components in heterogeneous systems.

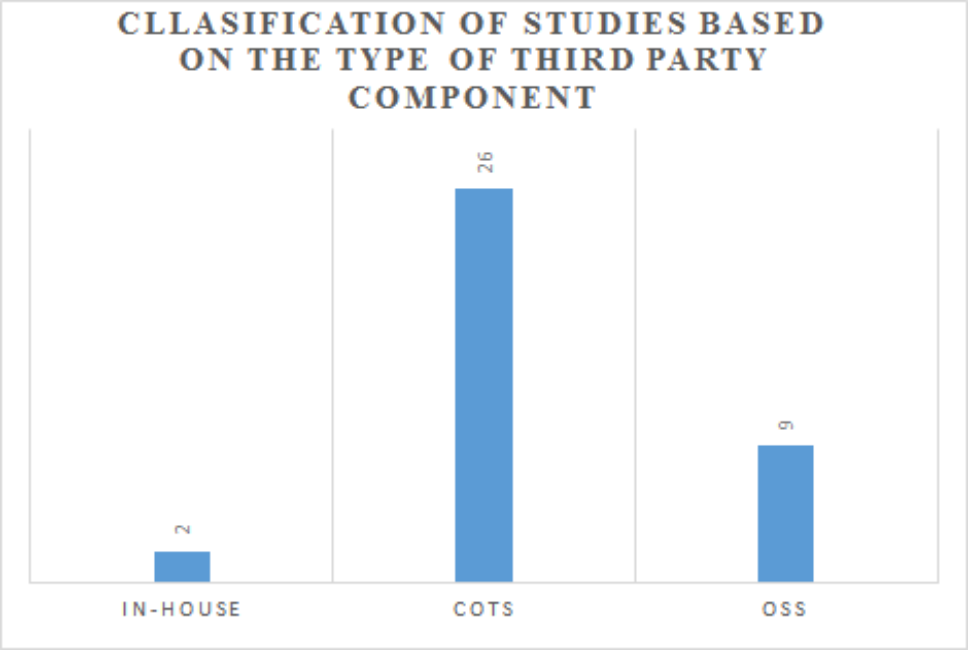

Figure 10 Classification based on third party components

The third party components that are used in heterogeneous system identified from the literature are shown in the table 8.

COTS	[S1] [S2] [S4] [S5] [S6] [S8] [S9] [S10] [S12] [S13] [S14] [S15] [S16] [S18] [S19] [S20] [S21] [S22] [S23] [S24] [S25] [S26] [S27] [S28] [S30] [S31].
OSS	[S2] [S7] [S10] [S13] [S16] [S17] [S26] [S27] [S29]
In-house software	[S3] [S17]

Table 8 Distribution based on the type of Third Party Component

Anup k et.al [S1] discussed the commercial of the shelf components. The author in the research has conducted an experiment and has proposed a failure simulation tool to access the robustness of the COTS based heterogeneous systems for operating system failures. Thomas et.al [S7] discussed the debugging of the integrated software systems with open source software as the third party system. A case study was conducted on Gentoo OSS community developers and third party software packages with different UNIX version and identified the characteristics for debugging of integrated systems. Jeff Tian [S10] discusses COTS and OSS components usage in heterogeneous systems. An experiment was conducted to measure the quality of the software components to provide the efficient usage of components, and to provide an unbiased assessment of the quality. Pekka et.al [S17] discussed OSS and in-house components in the usage of heterogeneous systems. The Authors in [S17] have conducted a literature review and interviews to investigate the perception and experiences of companies on the quality assurance factor of the open source components.

6.1.3 Analysis of Literature based on the Challenges faced while using third party components

In this section, the primary studies were analyzed based on the challenges faced while using third party components in heterogeneous systems. Challenges faced while using third party components is one of the areas of this research. The studies that discuss about the challenges are as follows: [S1] [S4] [S6] [S7] [S8] [S11] [S13] [S14] [S15] [S16] [S17] [S18] [S19] [S20] [S21] [S22] [S23] [S24] [S25] [S26] [S28] [S29] [S30] [S31].

Most of the literature discussed the challenges faced while using third party components in heterogeneous systems. John and Liming [S4] have conducted a case study, where a model was proposed to identify the complex dependency relationships of the simple modeling approach. Non-availability of source code and vendor support was identified as a challenge. Andre Takeshi et.al [S14] have performed a case study to assess the strategy, which assesses the quality of test activity for web services composition structural integration testing and discussed that interoperability issue as a challenge. Asiala and Matinlassi [S17] have performed an industrial study on nine different companies using OSS components. They have identified licenses issue a challenge faced by the companies while using OSS components.

Similarly, [S1] [S6] [S7] [S8] [S11] [S13] [S15] [S16] [S18] [S19] [S20] [S21] [S22] [S23] [S24] [S25] [S26] [S28] [S29] [S30] [S31] have discussed about the challenges faced while using the third party components. The distribution of studies based on challenges are shown in Figure 11.

The challenges faced while using third party components are shown in Table 9:

Challenge Id	Challenges	Studies Discussed about the challenges
SC1	Non availability of Source code	[S4], [S7], [S8], [S15], [S20], [S21], [S22], [S26], [S31]
SC2	Interface issues/standards	[S4],[S8]
SC3	Loss of support for older releases	[S4], [S8]
SC4	Cost Factors	[S6], [S14]
SC5	Vendors Support to Trouble reports/bug fixes	[S8]
SC6	Fault-tolerant Scheduling	[S11]
SC7	Lack of Quality documentation	[S13], [S26], [S30]
SC8	Interoperability Issues	[S13], [S14], [S16], [S28]
SC9	Trust worthiness	[S15], [S1]
SC10	Licensing issues	[S4],[S17], [S29]
SC11	Configuration management/ version control system	[S18], [S28]
SC12	Lack of flexibility	[S19], [S22], [S26]

Table 9 Challenges from SLR

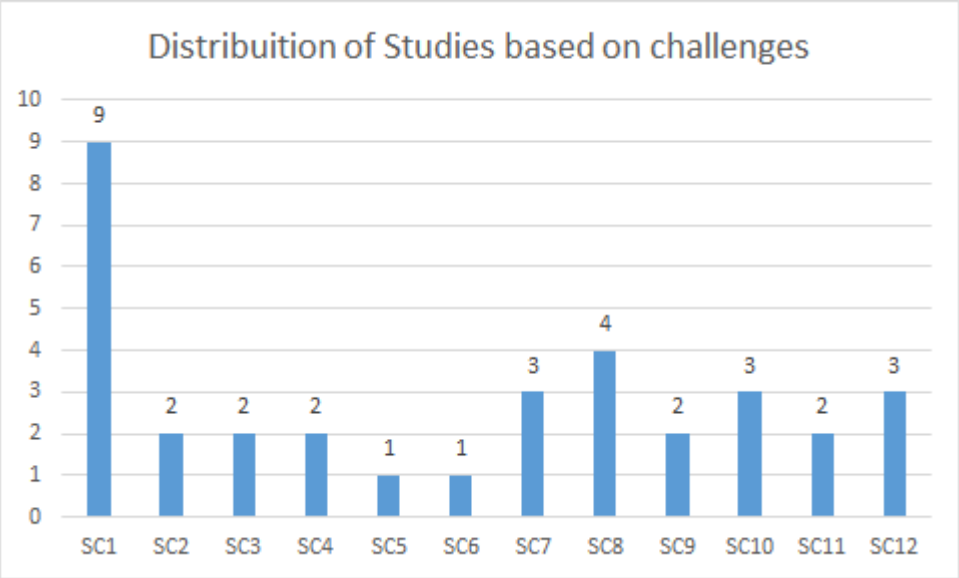

Figure 11 Classification based on challenges

SC1: Non-availability of Source Code: [S4], [S7], [S8], [S15], [S20], [S21], [S22], [S26], [S31]
Due to non-availability of the source code of the third party components, it is difficult to debug the systems and find for defects. It's hard to test the system due to unavailability of the source code. So non-availability of the source code is a challenge in a heterogeneous system. As this challenges mainly arise in the COTS systems.

SC2: Interface issues: [S4], [S8]
As the third party software with a defined interface is integrated with the heterogeneous systems with a defined interface, then the interface clash arise between the components. So interface compatibility is a challenge while integrating third-party components.

SC3: Loss of support for older releases: [S4], [S8]
The vendor support for the third party software is a challenging part of the component-based software engineering. When a third party software is integrated into the heterogeneous system and when the life cycle of the third party component is expired, then the support for the heterogeneous systems after the product life cycle of the third party component might be a challenge.

SC4: Cost Factor: [S6], [S14]
Cost factor might be a challenge. By using reusable software components to reduce the development cost but the source of the cost lies in the integration effort for the third party component with the heterogeneous systems. From the study [S6] the integration tax incurs the increment of the cost.

SC5: Vendors Support to Trouble reports/bug fixes: [S8]
The vendor support for the third party software failure is a challenge. When a defect is identified and sent as a report to the vendor, then there is no guarantee that the vendor will fix the issues and provide support. So vendor support to troubleshoot the report might be a challenge.

SC6: Fault tolerant scheduling: [S11]
Due to the wide of range of resource failures in the heterogeneous systems with third-party components, fault tolerant scheduling might be challenge.

SC7: Lack of documentation: [S13], [S26], [S30]
Due to lack of quality documentation of the third party component, the understandability of the component is reduced and it's hard to modify the software by others. So this might be a challenge while using third party component.

SC8: Interoperability Issues: [S13], [S14], [S16], [S28]
When developing a system with a large number of sub-systems with different interfaces the interoperability testing becomes complicated and might be a challenge. Interoperability might be easier when a system is developed with the standardized framework.

SC9: Trust worthiness: [S15], [S1]
As the web services provided by commercial vendors doesn't have the trust on the organization, due to which trustworthiness might be a challenge.

SC10: Licensing issues: [S17], [S29]
If a third party software with a limited license is used in the heterogeneous system, then the systems, which integrate this software, might fall under the violation of intellectual property rights of the open source software. Therefore, licensing issues might fall under challenge.

SC11: Configuration management/ version control system: [S18], [S28]
If the version of the integrated third party software changes then the compatibility of the third party software with the heterogeneous systems becomes complex. Therefore, Configuration management might be a challenge.

SC12: Lack of flexibility: [S19], [S22], [S26]
If a third party software integrated with the heterogeneous product and requirement provided by the customer need to be modified or changed, then the flexibility issues. Therefore, Lack of flexibility to change requirements might be a challenge

6.1.4 Analysis of Literature based on the Testing of heterogeneous systems with third-party components:

As the third party component used in heterogeneous systems and challenges were analyzed in the previous sections, the test design techniques used for testing of heterogeneous systems having third party components.
Upon analyzing the identified primary studies, 18 studies discussed the testing of heterogeneous systems. Anup and Matthew [S1] have proposed an approach to access the robustness of the Win32 application for operating systems failures and have used Software wrappers to improve the robustness of the COTS software. Andre et.al [S14] have performed a case study and have proposed a strategy for web service composition structural integration testing, which evaluates the quality of the test activity through coverage strategy. In [S15], the trustworthiness and non-availability of the source code of the web service oriented architecture were addressed by proposing unit testing techniques: specification based test case generation, collaborative testing, and group testing. In [S16] Ghazi et.al have performed a literature review to identify the testing techniques used for testing of heterogeneous systems and have conducted an empirical survey to perceive the usage of the testing techniques. Maki and Matinlassi [S17] have by performing interviews and have identified that static test design techniques were used for testing of open source components from integration level. Jiang et.al [S20] have proposed an approach for the selection of regression tests of the systems that have

third party software. A case study was conducted to validate the proposed approach.
The test design techniques that are used for testing heterogeneous systems having third party components are: [S1] [S13] [S14] [S15] [S16] [S17] [S18] [S20] [S21] [S22] [S23] [S24] [S25] [S26] [S27] [S29] [S30] [S31]

- **Use case testing:**
- **Behavior Capture testing:**
- **Exploratory testing:**
- **Equivalence class partitioning:**
- **Static test design techniques:**

6.1.5 Discussions and Conclusions for SLR:

A total of 31 studies were obtained from the database search strategy. 31 studies discuss third party components, 26 studies discuss the challenges faced while integrating third-party components and 18 studies discuss the test design techniques used for testing heterogeneous systems having third party components. The heterogeneous systems, which have third party system supply, were considered in this research apart from the other dimensions of the heterogeneous systems. The main reason for conduction of SLR is to identify third party components, challenges faced while integrating and test design techniques.

The study mainly focuses on three aspects of the research questions, identification of third party components used for development, Challenges faced while integrating third-party components and test design techniques used for testing of heterogeneous systems having third party components. The results obtained from the SLR studies were analyzed based on these aspects. It was observed that among 31 studies, 26 studies discussed COTS software systems and 9 studies discussed open source software and 2 studies discussed in-house software. From the literature results, COTS and OSS are the third party software that are mostly used in heterogeneous system development. In-house software was discussed by 2 studies and is less used when compared to other third party software.

This study also focuses on the identification of challenges faced while integrating third-party software with heterogeneous systems. The challenges faced while integrating third-party components were analyzed from the data extraction checklist. 9 studies discussed “non-availability of source code” as a challenge while testing, so it is one of the significant challenges during integration. This problem arises for COTS products. A possible explanation for this statement is that COTS software doesn’t provide source code for modification [59]. From studies [S4], [S8] interface issues, vendors support to trouble reports were discussed as a challenge. Vendor support for trouble reports occur for COTS and OSS software, as these were provided by the third party service providers, and in-house software has the less occurrence of this challenge. The cost factor is one of the challenges that are discussed in studies [S6], [S14]. Staff training and additional testing effort are the possible factors for the occurrence of this challenge. Similarly, the studies were analyzed and challenges faced were listed in the SLR results. These identified challenges were further complemented and supplemented by the interviews.

This study also focused on the test design techniques used for testing of heterogeneous systems having third party components. The test design techniques identified are Use case testing, Equivalence portioning, Behavior Capture testing, Exploratory testing, Equivalence class partitioning. These identified test design

techniques for testing of third party components were complemented and supplemented by the interviews. Maki and Matinlassi [S17] have identified static test design techniques for testing of open source components from integration level by performing interviews. From the studies [S17], [S24] the static techniques or specification based test design techniques proposed were used for testing of OSS and in-house software components. The possible explanation for the statement is that the availability of "source code" of the component. From studies [S20] [S26] [S27] black box test design techniques or dynamic testing is used for testing of COTS software. The possible explanation is that the COTS products do not provide source code for modification. From [S16], the testing of heterogeneous systems in industry was mainly performed using exploratory testing. The possible explanation is that exploratory testing is ease of use and is effective in the detection of critical defects. These identified test design techniques were validated through interviews and the test design between different third party components was investigated.

6.1.6 SLR Validity Threats:

In this section, the threats to validity to this SLR and mitigation strategies are listed. To precise the data obtained from the SLR, the risks obtained must be overcooked and mitigated. The validity threats to this SLR are as follows:

Firstly, the possible threat to this SLR is that the usage of wrong keywords and incorrect formulation of search string leads to missing of relevant and important literature. The mitigation strategy followed for this threat is that the author has taken help from the supervisor and Librarian of BTH library for the formulation of search string. The suggestions given by the supervisor and librarian were taken in improvising the search string and correcting the keywords. A pilot search was also conducted to check whether the relevant literature was identified from the database search.

Another possible threat to this SLR is regarding the quality of the primary studies. The results obtained from the SLR might fall under a possibility of studies that have low-quality standards, improper validation, irrelevant studies and studies that haven't followed a systematic methodology. Quality assessment criteria and data extraction form were formulated to assess the quality of the primary studies in order to mitigate this type of threat. This evaluation helps in understanding the contribution of the primary studies.

Domain applicability might be one of the possible threat this SLR. As the heterogeneous systems development and third party components has applicability in vast domains such as hardware systems, biological sciences, health care systems and so on which leads the search string to return literature from these areas. The search string was formulated by including the keywords related to software domain to mitigate this threat. This strategy includes the studies related to the software field.

Finally, after the completion of SLR process and summarizing the results there might be a threat of missing some relevant and important literature from the results. This scenario occurs when the author has missed some relevant information while documentation of results. To mitigate this threat author has created a spreadsheet where each and every primary study was recorded with its relevant information, and unique identifiers were given. The references to the summarized data were provided using Zotero with IEEE format.

6.2 Interviews Results:

This section talks about the results obtained from the interviews. As mentioned in Table 6, the interviewees selected were well experienced and were working under the organizations that are developing heterogeneous systems, which have third party components as their sub-systems. As mentioned in the research methodology chapter, thematic analysis is used for analyzing the interview data. A step-by-step process is followed to analyze the interview data. The steps followed to analyze the qualitative data are as follows [76]:

- Transcription
- Organization
- Familiarization
- Coding
- Themes

Transcription, Organization, and Familiarization:

Transcription is the process of taking notes in sound recording or video transcripts. In this study, the author has conducted 9 of the interviews online using skype and 3 face to face. The data obtained is collected using recording software. All recorded files were transcribed into a word document using express scribe. This process helps in maintaining each and every data from the interviews, which avoid bias [76].

In the Organization phase, the transcripts obtained from the data were further organized based on the Interview ids. The ids "INTx" for the interviewees shown in table 6 were used. These identifiers help for easy retrieval of data in the analysis phase.

In the familiarization phase, the data from the transcripts was read several times for a clear understanding, which helps in identification of significant and relevant codes from the transcripts.

Codes and Themes:

After the familiarization of the interview data, codes and themes were identified by following the guidelines provided by Braun and Clarke [47]. Transcripts obtained from the interviews were analyzed manually to identify codes. The codes identified were highlighted with different colors based on the categories. The identified codes were further grouped into categories based on the similarity and relevance.

"A theme captures something important about the data in relation to the research question and represent some level of patterned response within the data set"[47]. The similar codes were categorized into themes. The figure 12 shows the identified themes from the codes:

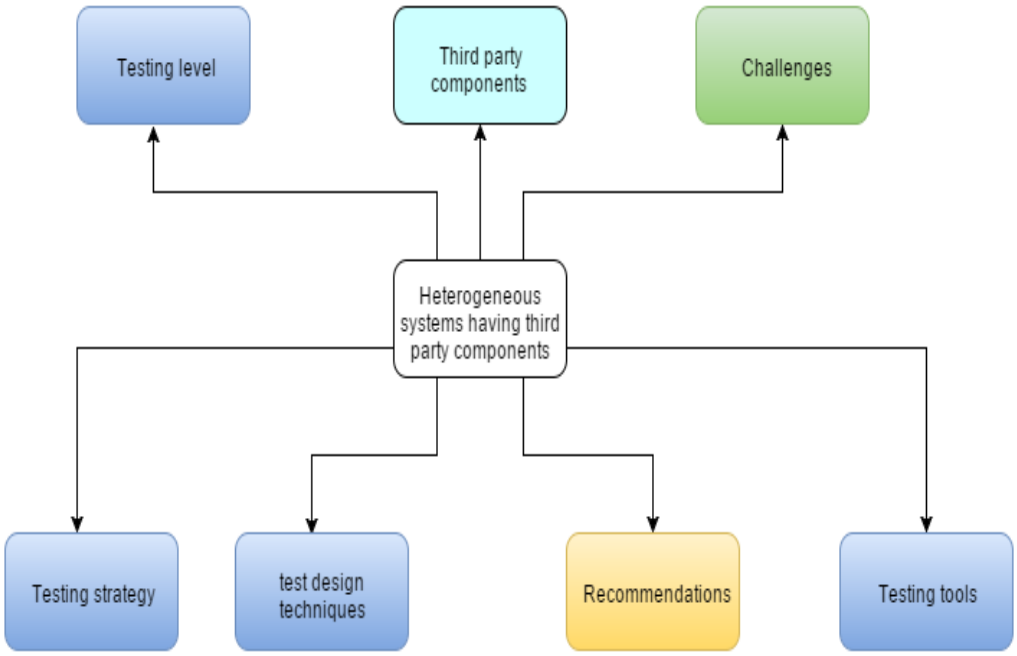

Figure 12 Themes

Codes identified from all the interviews:

Identified Code	**Inclusion Criteria for Codes**	**Theme**
COTS	Frequency, Relevance	Third party component
OSS/FOSS	Frequency, Relevance	Third party component
In-house software	Relevance	Third party component
Version control	Frequency, Relevance	Challenges
Backward compatibility	Frequency, Relevance	Challenges
Debugging of the software	Relevance	Challenges
Dependency on multiple components(Hard to maintain)	Relevance	Challenges
Definition for test scope	Relevance	Challenges
License issues	Frequency, Relevance	Challenges
Lack of documentation	Frequency, Relevance	Challenges
Understandability of the component	Frequency, Relevance	Challenges
Flexibility of the software	Relevance, Frequency	Challenges
Staff training	Frequency, Relevance	Challenges
Vendors Support to Trouble	Frequency,	Challenges

reports/bug fixes	Relevance	
Lack of expertise	Frequency, Relevance	Challenges
interoperability issues	Relevance	Challenges
Lifecycle coverage	Relevance	Challenges
Active community	Relevance	Challenges
Non-availability of source code	Relevance	Challenges
Use case testing	Relevance, Frequency	Test design technique
Equivalence class partitioning	Relevance, Frequency	Test design technique
Boundary value analysis	Relevance, Frequency	Test design technique
Condition coverage	Relevance	Test design technique
Decision coverage	Relevance	Test design technique
Static technique (informal Reviews, walkthroughs)	Relevance	Test design technique
Exploratory testing technique	Relevance, Frequency	Test execution technique
Decision table testing technique	Frequency	Test design technique
Manual testing	Frequency	Testing strategy
Automated testing	Frequency	Testing strategy
Soap UI	Frequency	Testing tools
Selenium	Frequency	Testing tools
In-house developed simulators	Frequency	Testing tools
Sonar	Frequency	Testing tools
Unit testing	Frequency	Testing levels
Integration level	Relevance, Frequency	Testing levels
Component level	Relevance, Frequency	Testing levels
Cover for the lifecycle of your product, not just for the commercial 3pp.	Relevance	Recommendations

Concentrate on the functionality rather than improving the third party software	Relevance	Recommendations
Check for backward compatibility and security issues.	Relevance	Recommendations
More time and resources: If there is more possibility for future use, better to go for development of in-house framework.	Relevance	Recommendations
Less time and more resources: Identify the suitable software in the market and use it. This would help us to focus on the main objective	Relevance	Recommendations
Less time and less resources: Choose most reliable open source software that meets the needs.	Relevance	Recommendations
Good understanding of cons and pros of third party components before use	Relevance	Recommendations
Impact analysis of the third party software should be done before its usage	Relevance	Recommendations
One round of regression testing is recommended to check the functionality of new version of the third-party software	Relevance	Recommendations
Proper handling of the failure scenarios.	Relevance	Recommendations
Dependency of the service should be written in the automated test suite(third party)	Relevance	Recommendations

Table 10 Codes and Themes for interviews

Themes:

- Third party component

- Challenges
- Recommendations
- Testing
  - Testing type
  - Testing level
  - Testing tool
  - Test design technique

6.2.1 Codes and Themes for RQ1:

A total of three codes were identified for third party components used in heterogeneous systems. The codes identified from the transcripts were selected based on the frequency of occurrence and relevance. These identified codes were categorized into themes. The table 11 shows the codes for RQ1 that is third party components used in heterogeneous systems.

Codes	Inclusion criteria	Theme	Frequency
COTS	Frequency, Relevance	Third party component	6
OSS/FOSS	Frequency, Relevance	Third party component	4
In-house software	Relevance	Third party component	2

Table 11 Codes and Themes for RQ1

Based on the identified codes, third party component was identified as a theme. All the codes come under Third party component Theme.

6.2.1.1 Analysis of interview results for RQ1:

Codes related to RQ1 that is Third party components in Heterogeneous systems are analysed in this section. A total of 3 codes were identified from 12 interviews and were categorized into a single theme Third party component.

Third party components: The codes “Commercial of the Shelf”, “open source software”, “in-house” were gathered under this theme. Every interviewee discussed different types of third party components used in heterogeneous systems. One of the interviews mentioned that “COTS software was used by them for product development”. In the other interview it was said that “we use different types of third party components depending on the situation and we mainly use COTS and OSS software for development”. In the other interview the practitioner said that “In-house software were used for development of heterogeneous systems”. 10 among 12 interviewees said that COTS and OSS software were used for the development of heterogeneous systems. 2 interviewees discussed the usage of in-house software in heterogeneous systems.

After the analysis of all interview transcripts, the third party components used in heterogeneous systems are as follows:

- COTS software discussed in 6 interviews
- OSS software discussed in 4 interviews
- In-house software discussed in 2 interviews

6.2.2 Codes and themes for RQ2:

The codes and themes for RQ2 were identified based on the inclusion criteria followed in RQ1. A total of 17 codes were identified from the transcripts. These identified codes were categorized into "challenges" theme. The table 12 shows the codes, themes, inclusion criteria and the frequency. Here the frequency is that the total number of times the code discussed in the interviews.

SNo:	Codes	Inclusion criteria	Theme	Frequency
IC1	Version control	Frequency, Relevance	Challenges	12
IC2	Backward compatibility	Frequency, Relevance	Challenges	3
IC3	Debugging of the software	Relevance	Challenges	1
IC4	Dependency on multiple components(Hard to maintain)	Relevance	Challenges	1
IC5	Definition for test scope	Relevance	Challenges	1
IC6	License issues	Frequency, Relevance	Challenges	3
IC7	Lack of documentation	Frequency, Relevance	Challenges	2
IC8	Understandability of the component	Frequency, Relevance	Challenges	3
IC9	Flexibility of the software	Relevance, Frequency	Challenges	2
IC10	Staff training	Frequency, Relevance	Challenges	2
IC11	Vendors Support to Trouble reports/bug fixes	Frequency, Relevance	Challenges	12
IC12	Lack of expertise	Frequency, Relevance	Challenges	3
IC13	interoperability issues	Relevance	Challenges	1
IC14	Lifecycle coverage	Relevance, Frequency	Challenges	3
IC15	Active community	Relevance	Challenges	1
IC16	Non availability of source code	Relevance, Frequency	Challenges	4

Table 12 Codes and Themes for RQ

6.2.2.1 Analysis of interview results for RQ2:

A total of 16 codes are identified from the questions posed on the challenges faced while integrating third-party components in interviews. All the challenges identified from the transcripts were categorized into challenges theme.

Challenges: The main challenges that were cited by 12 respondents in the interviews were Version control and Vendor support to trouble reports/bug fixes. As version control a general problem for any type of systems, but from the interview results it is observed that it is magnified in the context of heterogeneous systems. 4 interviewees discussed the lack of documentation as a challenge. From one of the interviews: "*one of the biggest problem while testing the heterogeneous systems having third party software is lack of documentation to the third party software*." 3 interviewees discussed that licensing issues as a challenges while

integrating OSS software. From another participant "*I think it's hard to define the scope of testing ahead for the heterogeneous systems*", defining scope of testing at the beginning might be challenge which leads to increase in cost. Lifecycle coverage was stated as a challenge by 3 interviewees. From one of the interview "*You have to make sure that the life cycle of your product should be covered not only the third party software, so that you have vendor support for the product life cycle*", Here the interviewee perspective is that the vendors support for life cycle coverage of the product is a challenge faced while integrating third party to heterogeneous systems. One interview discussed that dependency of third party component on multiple components is a challenge due to the maintenance of heterogeneous systems. From the responses of interviews, the challenges faced while integrating third-party components in heterogeneous systems are backward compatibility, Debugging, licensing issues, flexibility of the software, staff training, lack of expertise, compatibility issues, active community and competence of the component. From 2 interviews staff training is a challenge faced while integration of in-house software due to increase in cost and time factors. Debugging of heterogeneous systems is a challenge due to the unavailability of the source code for the third-party software. One of the interviewee stated that *"When a freeware is integrated into the commercial product, then there is a risk that the commercial product ends up into freeware."* Here licensing is a challenge due to the violation of intellectual property rights when you integrate third party software in your product development.

When author posed a question that the challenges occurred limit to your organization, then all the respondents verbally expressed that these challenges limit to the type of third party component but not to the single group or organization.

6.2.3 Codes and Themes for RQ3:

The codes and themes for RQ3 were identified based on the inclusion criteria followed in RQ1. Total of 14 codes related to RQ3 were identified from the interview transcripts. These identified codes were categorized into four themes, they are as follows:

- Test design techniques
- Testing tools
- Testing strategy
- Testing level

Codes	Inclusion criteria	Theme (Categories)	Frequency
Use case testing	Relevance, Frequency	Test design technique	3
Equivalence class partitioning	Relevance, Frequency	Test design technique	2
Boundary value analysis	Relevance, Frequency	Test design technique	3

Condition coverage	Relevance	Test design technique	1
Decision coverage	Relevance	Test design technique	1
Static technique (informal Reviews, walkthroughs)	Relevance	Test design technique	2
Exploratory testing technique	Relevance, Frequency	Test execution technique	8
Decision table testing technique	Relevance	Test design technique	1
Manual testing	Frequency	Testing strategy	5
Automated testing	Frequency	Testing strategy	4
Soap UI	Frequency	Testing tools	5
Selenium	Frequency	Testing tools	4
In-house developed simulators	Relevance	Testing tools	1
Sonar	Frequency	Testing tools	3
Unit testing	Frequency	Testing levels	6
Integration level	Relevance, Frequency	Testing levels	8
Component level	Relevance, Frequency	Testing levels	7

Table 13 Codes and Themes for RQ3

6.2.3.1 Analysis of interview results based on RQ3

Based on the analysis of the interview responses about testing of heterogeneous systems having third party components, a total of 16 codes were identified. These identified codes were again categorized into 4 themes "test design techniques," "testing tools," "testing level" and "testing strategy."

Test design techniques: From the interview responses a total of 8 codes were identified under the category test design techniques. From one of the interview *"From the requirements and user stories, the use cases are defined to perform user acceptance testing*". Here the black box test design technique that is user acceptance testing is used for testing of systems having a commercial of the shelf components as third party components. As User case testing was discussed as a test design techniques used for testing of systems having COTS products by 3 interviewees. 2 interviewees stated that equivalence class partitioning was used and 3 respondents stated that boundary value analysis is used. But from one of the interview "*test design techniques were selected based kind of issue".* From one of the responses "decision and condition coverage (*Matrices) were used to calculate the complexity of the code based on the development of feature performed by code*

reviewers". Here the interviewee work under the development of systems having open source software (OSS), so the heterogeneous system having OSS components were tested by following white box or specification-based testing. From 8 interviews, exploratory testing is one of the test execution techniques mainly used for testing of heterogeneous systems having third party software. In one of the interview, testing of heterogeneous systems that have COTS components is performed by using dynamic black box test design techniques. Boundary value analysis and equivalence class partitioning were used in testing of their product which have COTS.

Testing levels: From most of the interviews *"Most of the testing is performed at the integration and component level for the systems having commercial of the shelf software".* From one of the interviewees "*Unit testing is mainly used for the testing of systems that provide source code such as open source software systems and in-house software systems*", here the interviewee opinion is that systems that provide source code for testing were performed unit level testing. From one of the interview system level or end to end testing is used for testing of heterogeneous systems having third party components.

Testing tools: The testing tools identified from the interviews results are SOAPUI, Selenium and in-house simulators. From one of the interviewee, in-house simulator is used for testing of in-house software systems. "*Selenium is mainly used for to test the functionality of the web-based applications"* Here the interviewee perspective is that web-based system of systems functionality is validated using automated testing tool (Selenium). *"Sonar is one of the testing tools used to review the code of the software systems (Static automated testing)"* here the coding standards for the software systems which provide source code were examined using static automated testing tools

Testing strategies: In this section the strategy followed for testing of different third party components were identified. Manual exploratory testing and automated testing is followed for testing of third party components. In one of the interview "to make sure the version used is compatible with the system, we start with exploratory testing". Here the interviewee's perception is that manual testing was performed at the beginning of the third party integration. Automation is performed at the end to make sure the system level test.

6.2.4 Codes and themes for RQ4:

The codes and themes for RQ4 were identified based on the inclusion criteria followed in RQ1. A total of 11 codes were identified from the transcripts. These identified codes were categorized into "recommendations" theme.

Codes	Inclusion criteria	Theme
Cover for the lifecycle of your product, not just for the commercial 3pp.	Relevance	Recommendations
Concentrate on the functionality rather than improving the third party software	Relevance	Recommendations
Check for backward compatibility and security issues.	Relevance	Recommendations

More time and resources: If there is more possibility for future use, better to go for development of the in-house framework.	Relevance	Recommendations
Less time and more resources: Identify the suitable software in the market and use it. This would help us to focus on the main objective	Relevance	Recommendations
Less time and less resources: Choose most reliable open source software that meets the needs.	Relevance	Recommendations
Good understanding of cons and pros of third party components before use	Relevance	Recommendations
Impact analysis of the third party software should be done before its usage	Relevance	Recommendations
One round of regression testing is recommended to check the functionality of new version of the third-party software	Relevance	Recommendations
Proper handling of the failure scenarios.	Relevance	Recommendations
Dependency of the service should be written in the automated test suite(third party)	Relevance	Recommendations

Table 14 Codes and Themes for RQ4

6.2.4.1 Analysis of interview results for RQ4:

From the analysis of interview transcripts, a total of 11 codes were identified for RQ4. These codes were categorized under the theme "Recommendations".

***Recommendations**:* One of the interviewee perspectives is "*You need to have cover for the lifecycle of your product, not just for the commercial 3pp. Because you need to provide support for your product whole time"*. "*Don't lock yourself in integrating to hard towards another 3pp so that you can easily swap them out if necessary"*, here the interviewee perspective is that the c*heck for* backward compatibility and security issues of the third party software that are used must be verified before using. From one of the interview "*Have a good understanding of cons and pros of respective third party components before you use*", here the interviewee perspective is that the advantages and disadvantages of the selected third party component must be known which help to avoid delay in vendor support. *"Impact analysis of the third party software should be done before its usage,"* here the interviewee perspective is that the behavior and impact of the third party software must be known earlier, so that most of the mistakes are avoided. From one of the interview, one round of regression testing is recommended to check the functionality of new version of the third party software which helps in reduction of costs related to testing. One of the interviewee perspectives is that the failure case scenarios must be handled with the systems having third party components. From

one of the interview "*Dependency of the service should be written in the automated test suite (third party)",* here the interviewee perspective is that the dependencies of subsystems must be tested. These third-party software dependencies must be written in the test suite for automation. This helps in reduction of testing effort and cost at the end.

7 DISCUSSIONS AND VALIDITY THREATS

7.1 Discussions

In this section, we provide comprehensive discussions of the results obtained from the SLR and Interviews. The analyzed results were further validated if found coherence with the SLR findings. Which helps in strengthening the results and analysis of the research and provides validation. A comparative analysis is used to discuss the results obtained from SLR and interview, which is further used for validation of coherent data in this study.

7.1.1 Discussion for RQ1:

As stated in SLR COTS, OSS and in-house software components were used for development of heterogeneous systems. As 26 studies from SLR discussed the COTS software, 9 studies discussed OSS and 2 studies discussed the in-house software products. From SLR results, most of the focus is on COTS and OSS products when compared to in-house software. From interviews, 6 among 12 interviewees stated that COTS products were mainly used in their product development. 4 of the interviewees said that OSS software is used. 2 interviewees stated that in-house is used in heterogeneous systems. Hence, it is observed that in industry COTS, OSS and in-house software were used in the development of heterogeneous systems. One of the possible explanation for this is from the interviews "selection of third party components is mainly done based on the factors time, quality and cost". This leads to the possible lessons learned

Lessons learned: First from the literature it is observed that COTS and OSS software was mainly used for the development of heterogeneous systems. But from the interviews, it is observed that COTS, OSS and in-house were used for the development. So COTS, OSS and in-house software were employed in the development of heterogeneous systems having third party components. The selection of third party component is mainly done based on the factors time, quality and cost.

7.1.2 Discussion for RQ2:

As stated in SLR a total of 12 challenges were identified and are listed in section 6.1.3 with identifiers "SC". SC1 "Non-availability of source code" were discussed by 9 studies and SC8 "interoperability issues" were discussed 4 studies. The distribution of studies based on challenges was discussed in the section 6.1.3. The challenges obtained from the interviews were 16. Interviews were conducted to validate and complement the SLR results to indicate the potential challenges. The challenges derived from interviews were listed in section 6.2.3 with the identifier "IC". A total of 9 challenges were newly identified from the interviews are as follows: IC2, IC3, IC5, IC8, IC10, IC12, IC14, and IC15. 7 challenges discussed in the SLR results were validated and are as follows: SC1, SC5, SC7, SC8, SC10, SC11, and SC12. The challenges SC2, SC3, SC4, SC6, and SC9 obtained from the literature were not discussed in the industrial context. Finally, a total of 21 challenges were identified from both SLR and Interviews. The table 15 lists the

identified challenges and categorizes them based on the type of third party component.

Id	Challenges	Obtained	Third party component
CH1	Non availability of Source code	SLR, Interviews	COTS
CH2	Interface issues/standards	SLR	COTS, OSS
CH3	Loss of support for older releases	SLR	COTS, OSS
CH4	Cost Factors	SLR	COTS
CH5	Vendors Support to Trouble reports/bug fixes	SLR, Interviews	COTS, OSS
CH6	Fault-tolerant scheduling	SLR	OSS, COTS, in-house
CH7	Lack of Quality documentation	SLR, Interviews	OSS
CH8	Interoperability Issues	SLR, Interviews	OSS, COTS, in-house
CH9	Trust worthiness	SLR	OSS
CH10	Licensing issues	SLR, Interviews	OSS, COTS
CH11	Configuration management/ version control	SLR, Interviews	OSS, COTS, in-house
CH12	Lack of flexibility	SLR, Interviews	COTS
CH13	Backward compatibility	Interviews	COTS, in-house, OSS
CH14	Debugging of the software	Interviews	COTS
Ch15	Maintenance issues	Interviews	COTS, OSS
CH16	Definition for test scope	Interviews	COTS,OSS, in-house
CH17	Staff training	Interviews	COTS, OSS, In-house
CH18	Lack of expertise	Interviews	COTS, in-house, OSS
CH19	Life cycle coverage	Interviews	COTS, OSS
CH20	Active community	Interviews	COTS, OSS
Ch21	Understandability of the component	Interviews	COTS, OSS, in-house s

Table 15 Challenges obtained from SLR and Interviews

The challenges that are discussed in SLR and Interviews were considered as most important. Then the challenges obtained from the interviews and then challenges from the SLR.

Lessons Learned: From the analysis of SLR and Interviews results for RQ2 the possible lessons learned are:
Non availability of the source code, vendors support to trouble report, lack of quality documentation, Interoperability issues, licensing issues, configuration management/version control and lack of flexibility were the main challenges faced by the practitioners. Possible explanation is that these are the challenges

discussed in Interviews and are in-line with the literature. Non-availability of source code, lack of flexibility, debugging of the software were incurred while integrating COTS product. One of the possible explanation for the statement from interview results is "COTS products doesn't provide source code, which is difficult to debug the software and hard to maintain" which are in-line with the literature [S8, S19]. Licensing issue and trust worthiness is one of the challenge that is mainly faced by open source software systems. Possible explanation for the statement from one of the interviewee is "if you integrate open source software to your commercial product then there might be a risk that the commercial product falls under freeware. Which leads to violation of Intellectual property rights and provide incurred costs to the management" which is in-line with the literature [S17]. The challenges such as fault tolerance, backward compatibility, configuration management were the challenges mainly faced while using COTS, OSS, and in-house software.

7.1.3 Discussion for RQ3 and RQ3.1:

From SLR results, author have identified Use case testing, equivalence class portioning, behavior capture testing and exploratory testing for testing of heterogeneous systems having third party components. From interview results, we have identified 8 testing techniques out of which 7 were test design techniques and 1 test execution technique (exploratory testing). Testing strategies manual and automated testing were discussed for testing of systems having third party components. Testing tools such as SOAPUI, selenium, Sonar and in-house development were used in industry. Testing levels mainly concentrated while testing of heterogeneous systems were integration and system level. Form the interview results test design techniques used for testing of heterogeneous systems having COTS are use case testing, equivalence class partitioning, and boundary value analysis. The possible explanation from one of the interview is that "testing design techniques used for testing of systems having COTS product is Black box, due to non-availability of the source code". Here the interview results were in line with the literature [S20]. From one of the interview results testing of systems having open source software is mainly based on the specification based testing, where matrices were used to check the complexity of the code. Static verification of the code is also performed while integration open source software systems which is in line with the literature [S17, S24]. From one of the interview, Black duck tool is identified as a static analysis tool which is used to check the security vulnerabilities and issues in the source code before usage. One interviewee with 10 years' experience stated that "the beginning phase of the testing is performed through exploratory testing and then comes unit testing". From 8 interview results, it is observed that exploratory testing techniques were mainly to test the heterogeneous systems due to the factor "ease of use" and "efficiency in defect detection". These findings are in line with the study [S16].

Lessons learned:
From the SLR and interview results, it is observed that test design for heterogeneous systems with different types of third party components differ. The possible explanation for this is "testing of software having COTS products is done through black box test design". 5 out of 12 interviews said that black box test design is mainly used for testing of COTS systems which is in-line with [S20]. As open source and in-house software provide source code for modification, Specification-based verification and validation techniques are used for testing of third party components, which is in-line with literature [S17, S24, S27]. And from 8 interviews, exploratory testing is used for testing. SO test design for heterogeneous systems having different types of third party component varies.

7.1.4 Discussion for RQ4

From the analysis of interview results for RQ4, the author has proposed recommendations for practitioners using different types of third party components in product development. The table below shows the recommendations:

Type of third party component	Recommendations
COTS	• Impact analysis of the third party software should be done before its usage • Check for backward compatibility and security issues of the third party software used • Check for the lifecycle coverage of your product, not just for the commercial 3pp.
OSS	• Check with the license issues before the integration of third party software. • Check the background of the third party service provider before integration. • The dependency of the third party systems must be in automated test suites for regression. • One round of regression testing is recommended to check the functionality of new version of the third-party software.
In-house software	• The functionality of the product must concentrated rather than the improvement of the third party software
General	• Selection of the component must be done based on the quality, cost and time factors. • More time and resources: If there is more possibility for future use, better to go for development of the in-house framework. • Less time and more resources: Identify the suitable software in the market and use it. This would help us to focus on the primary objective • Less time and less resources: Choose most reliable open source software that meets the needs.

Table 16 Recommendations

In the above table the recommendation for the practitioners were provided by categorizing them into COTS, OSS, in-house and general. Here the COTS section provide instructions for the practitioners using COTS product, OSS section provides guidelines to the practitioner using open source software, in-house section for the practitioners using in-house software. The general section provides the guidance to the practitioner in the selection process using third party component in product development.

7.2 Threats to validity

This section discusses about the validity threats for the interviews.

7.2.1 Internal validity threats:

Internal validity threats refer to other factors that could alter the relationship between applied treatment and the results [77]. The possible threats to internal validity could be an instrument for data collection, selection of interviewee and misinterpretation of extracted data. The instrument for data collection in this research i.e. questions for interview were made very simple and unambiguous for the participant to understand. The questionnaire was improved iteratively through continuous reviews from researcher and supervisor to attain high quality and understandable content. Also feedback from various masters' students who are working in this domain of research helped to mitigate the possibility of this threat.

The selection of participants for the interview was carefully made to ensure that they have considerable experience in testing with heterogeneous components. Which helped in gaining reliable and valid answers and hence avoiding selection bias. The other possible internal validity threat i.e. misinterpretation of extracted data could persist due to researcher bias. The researcher may have interpreted the data in his perception which at times could contradict with the responses of interviewee. To mitigate this risk, the data from transcripts is summarized and sent to the respondent for validation.

7.2.2 External validity threats:

The factors which would limit the applicability of the results to larger sample size i.e. affecting the generalizability of results are external validity threats [44]. To mitigate this threat, the sample was carefully selected so that it would represent the whole population. Also, the respondents chosen were of high experience and from varying backgrounds. The diversity of respondents in their roles, varying size of organizations, various ethnic backgrounds helped to ensure the applicability of the results to larger population beyond the selected sample.

7.2.3 Construct Validity:

Construct validity is about how accurately the test measures the intended construct [44]. In this research, the test implies to questionnaire in the interview study. The threat here would hinder the purpose of questionnaire in obtaining the objective of interview study. So, to mitigate this threat, the questionnaire is made simple and unambiguous for the user to understand. Also, the inputs from SLR study are taken into consideration while designing the questionnaire. Pilot interviews with participants having experience in this research field are conducted. Feedback was taken at end of every interview to enhance the understandability of questionnaire.

The understandability of the term Heterogeneous systems might be one of the threats. So, to mitigate this threat, the author has explained the definition of the term heterogeneous systems at the beginning of the interview and then proceeded with the interview process. This helps to avoid confusion to interviewee about the term heterogeneous system and can expect relevant data from the interview.

7.2.4 Conclusion Validity:

The factors that may avert the correctness of the conclusions are the threats to conclusion validity [78]. Reliability of the data is a key threat to conclusion

validity. To avoid this threat in literature review, a systematic procedure with a quality assessment filter is adapted to gather quality data from literature. The extracted data was put in an excel sheet, followed by rigorous analysis of the data properties, that aptly lead to the relationships in data.

For interview study, data extracted from respondents was investigated using thematic analysis. Iterative execution of thematic analysis to generate codes and themes helped attain insightful relationships. Finally, the conclusions for the research were followed by a comparative analysis of SLR results and survey results.

8 CONCLUSION

This section reports conclusion and future work of the study

8.1 Conclusions

In this study, the author identified different types of third party components used in heterogeneous systems, challenges faced while integrating third party components, test design techniques used for testing of various types of third party components and recommendations to the practitioners. A systematic literature review was performed to identify third party components used for product development; challenges faced while integrating third-party components and provided test design techniques used for testing of systems having third party components. The results obtained from SLR were complemented and supplemented using a qualitative interview approach. Finally, the author has provided guidelines to the practitioner using third party components through interviews.

SLR results indicated three third party components COTS, OSS and in-house software systems, among which COTS and OSS have more focus in literature. Moreover, 12 challenges were identified from the literature shown in SLR results, among which non-availability of source code, lack of flexibility, licensing issues, interoperability issues, lack of quality documentation were found as most faced challenges. In addition to that, 6 testing techniques were identified for testing of heterogeneous systems having third party component from the SLR results.

The results from interviews state that COTS and OSS software were mainly used for development of heterogeneous systems. At the same time, from one of the interviewee, large-scale companies adopt in-house software for the development rather than going for COTS and OSS. Also, a total of 21 challenges were identified from SLR and interviews among which ch1, ch5, ch8, ch10, ch11 and ch12 were found significant. The challenges faced were listed in the discussion section.

In addition, from interviews, author have identified 8 test design techniques for different types of third party components. Testing of systems having OSS or in-house software is mainly done using white box test design such as condition coverage and decision coverage. Test execution techniques such as exploratory testing were also used for testing of these systems. Static testing such as informal reviews, walkthrough was done on OSS software. So, test design differs between different types of third party components.

In addition, from the analysis of interview results, this study proposed 10 recommendations to the practitioners who are using third party components for product development. The recommendations were listed in discussions section. The recommendations proposed were categorized into four sections COTS, OSS, in-house and general.

Thus, this study provides the practitioners and researchers with a clear idea about types of third party software used, challenges faced while using different types of third party components and testing design techniques used for testing of systems having different types of third party systems in the context of heterogeneous systems.

REFERENCES

[1] G. Pour, "Software Component Technologies: JavaBeans and ActiveX," in *tools*, 1998, pp. 398 – 398.

[2] A. N. Ghazi, J. Andersson, R. Torkar, K. Petersen, and J. Börstler, "Information Sources and their Importance to Prioritize Test Cases in Heterogeneous Systems Context," *Communications*.

[3] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," *Engineering*, vol. 2, p. 1051, 2007.

[4] G. Pour, "Component-based software development approach: new opportunities and challenges," in *Technology of Object-Oriented Languages, 1998. TOOLS 26. Proceedings*, 1998, pp. 376–383.

[5] G. Lewis, E. Morris, P. Place, S. Simanta, D. Smith, and L. Wrage, "Engineering systems of systems," in *Systems Conference, 2008 2nd Annual IEEE*, 2008, pp. 1–6.

[6] K. Petersen, K. Rönkkö, and C. Wohlin, "The impact of time controlled reading on software inspection effectiveness and efficiency: a controlled experiment," in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, 2008, pp. 139–148.

[7] Q.-M. Xia, T. Peng, B. Li, and Z. Feng, "Study on Automatic Interoperability Testing for E-Business," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, 2009, pp. 1–4.

[8] A. Forward and T. C. Lethbridge, "A taxonomy of software types to facilitate search and evidence-based software engineering," in *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, 2008, p. 14.

[9] J. Kontio, "A case study in applying a systematic method for COTS selection," in *Software Engineering, 1996., Proceedings of the 18th International Conference on*, 1996, pp. 201–209.

[10] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," in *2007 Future of Software Engineering*, 2007, pp. 188–198.

[11] M. Morisio and N. Sunderhaft, "Commercial-off-the-shelf (cots): A survey," *Univ. Md.*, 2000.

[12] Ø. Hauge, C. Ayala, and R. Conradi, "Adoption of open source software in software-intensive organizations–A systematic literature review," *Inf. Softw. Technol.*, vol. 52, no. 11, pp. 1133–1154, 2010.

[13] A. A. Ssaed, W. Kadir, and S. Z. M. Hashim, "Metaheuristic Search Approach Based on In-house/Out-sourced Strategy to Solve Redundancy Allocation Problem in Component-Based Software Systems," *Int. J. Softw. Eng. Its Appl.*, vol. 6, no. 4, pp. 143–154, 2012.

[14] P. A. Oberndorf, "Facilitating component-based software engineering: COTS and open systems," in *Assessment of Software Tools and Technologies, 1997., Proceedings Fifth International Symposium on*, 1997, pp. 143–148.

[15] A. K. Ghosh and M. Schmid, "An approach to testing COTS software for robustness to operating system exceptions and errors," in *Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on*, 1999, pp. 166–174.

[16] J. M. Haddox and G. M. Kapfhammer, "An approach for understanding and testing third party software components," in *Reliability and Maintainability Symposium, 2002. Proceedings. Annual*, 2002, pp. 293–299.

[17] M. Ruffin and C. Ebert, "Using open source software in product development: A primer," *Softw. IEEE*, vol. 21, no. 1, pp. 82–86, 2004.

[18] P. A. Freeman, "Software Engineering Body of Knowledge (SWEBOK)," in *INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING*, 2001, vol. 23, pp. 693–696.

[19] D. Wang, B. Barnwell, and M. B. Witt, “A cross platform test management system for the SUDAAN statistical software package,” in *Software Engineering Research, Management and Applications, 2009. SERA’09. 7th ACIS International Conference on*, 2009, pp. 237–244.
[20] L. Mariani, “Behavior capture and test for verifying evolving component-based systems,” in *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, 2004, pp. 78–80.
[21] M. Narita, M. Shimamura, K. Iwasa, and T. Yamaguchi, “Interoperability verification for web service based robot communication platforms,” in *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, 2007, pp. 1029–1034.
[22] E. Putrycz and M. Slavescu, “Solving performance issues in COTS-based systems,” in *Commercial-off-the-Shelf (COTS)-Based Software Systems, 2006. Fifth International Conference on*, 2006, p. 10–pp.
[23] A. P. Mathur, *Foundations of Software Testing, 2/e*. Pearson Education India, 2008.
[24] C. Keating, R. Rogers, R. Unal, D. Dryer, A. Sousa-Poza, R. Safford, W. Peterson, and G. Rabadi, “System of systems engineering,” *Eng. Manag. J.*, vol. 15, no. 3, pp. 36–45, 2003.
[25] A. N. Ghazi, K. Petersen, and J. Börstler, “Heterogeneous Systems Testing Techniques: An Exploratory Survey,” in *Software Quality. Software and Systems Quality in Distributed and Mobile Environments*, Springer, 2015, pp. 67–85.
[26] M. A. Rossi and others, “Decoding the‘ free/open Source (F/OSS) Software Puzzle’, a Survey of Theoretical and Empirical Contributions,” 2004.
[27] I. Boivie, C. \AAborg, J. Persson, and M. Löfberg, “Why usability gets lost or usability in in-house software development,” *Interact. Comput.*, vol. 15, no. 4, pp. 623–639, 2003.
[28] A. Ahmed, *Software testing as a service*. CRC Press, 2009.
[29] I. Jovanović, “Software Testing Methods and Techniques,” *IPSI BgD Trans. Internet Res.*, vol. 30, 2006.
[30] D. Graham, E. Van Veenendaal, and I. Evans, *Foundations of software testing: ISTQB certification*. Cengage Learning EMEA, 2008.
[31] W.-T. Tsai, Y. Chen, and R. Paul, “Specification-based verification and validation of web services and service-oriented operating systems,” in *Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on*, 2005, pp. 139–147.
[32] J. Pan, P. Koopman, and D. Siewiorek, “A dimensionality model approach to testing and improving software robustness,” in *AUTOTESTCON’99. IEEE Systems Readiness Technology Conference, 1999. IEEE*, 1999, pp. 493–501.
[33] J. Strecker and A. M. Memon, “Relationships between test suites, faults, and fault detection in gui testing,” in *Software Testing, Verification, and Validation, 2008 1st International Conference on*, 2008, pp. 12–21.
[34] A. Jaaksi, “Experiences on product development with open source software,” in *Open source development, adoption and innovation*, Springer, 2007, pp. 85–96.
[35] L. C. Briand, Y. Labiche, and M. M. Sówka, “Automated, contract-based user testing of commercial-off-the-shelf components,” in *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 92–101.
[36] J. Zheng, B. Robinson, L. Williams, and K. Smiley, “A lightweight process for change identification and regression test selection in using COTS components,” in *Commercial-off-the-Shelf (COTS)-Based Software Systems, 2006. Fifth International Conference on*, 2006, p. 7–pp.
[37] N. P. Kropp, P. J. Koopman, and D. P. Siewiorek, “Automated robustness testing of off-the-shelf software components,” in *Fault-Tolerant Computing, 1998. Digest of Papers. Twenty-Eighth Annual International Symposium on*, 1998, pp. 230–239.
[38] H. Reza, S. Buettner, and V. Krishna, “A method to test component off-the-shelf (COTS) used in safety critical systems,” in *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, 2008, pp. 189–194.

[39] M. de Jonge, “Developing product lines with third-party components,” *Electron. Notes Theor. Comput. Sci.*, vol. 238, no. 5, pp. 63–80, 2009.
[40] A. B. Kajornboon, “Using interviews as research instruments,” *E-J. Res. Teach.*, vol. 2, no. 1, 2005.
[41] D. E. Gray, *Doing research in the real world*. Sage, 2013.
[42] J. W. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2013.
[43] C. W. Dawson, *Projects in computing and information systems: a student’s guide*. Pearson Education, 2005.
[44] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
[45] D. S. Cruzes and T. Dyb\a a, “Research synthesis in software engineering: A tertiary study,” *Inf. Softw. Technol.*, vol. 53, no. 5, pp. 440–455, 2011.
[46] M. Rodgers, A. Sowden, M. Petticrew, L. Arai, H. Roberts, N. Britten, and J. Popay, “Testing methodological guidance on the conduct of narrative synthesis in systematic reviews effectiveness of interventions to promote smoke alarm ownership and function,” *Evaluation*, vol. 15, no. 1, pp. 49–73, 2009.
[47] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qual. Res. Psychol.*, vol. 3, no. 2, pp. 77–101, 2006.
[48] C. Wohlin and A. Aurum, “Towards a decision-making structure for selecting a research design in empirical software engineering,” *Empir. Softw. Eng.*, pp. 1–29, 2014.
[49] L. M. Given, *The Sage encyclopedia of qualitative research methods*. Sage Publications, 2008.
[50] R. Spitzlinger, “Mixed Method Research-Qualitative Comparative Analysis,” 2006.
[51] R. M. Belbin, *Team roles at work*. Routledge, 2012.
[52] L. A. Guion, D. C. Diehl, and D. McDonald, “Conducting an in-depth interview,” 2011.
[53] B. Boehm, W. Brown, and R. Turner, “Spiral development of software-intensive systems of systems,” in *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, 2005, pp. 706–707.
[54] O. Mazhelis, P. Tyrväinen, and L. Frank, “Vertical software industry evolution: The impact of software costs and limited customer base,” *Inf. Softw. Technol.*, vol. 55, no. 4, pp. 690–698, Apr. 2013.
[55] J. Brondum and L. Zhu, “Visualising architectural dependencies,” in *Proceedings of the Third International Workshop on Managing Technical Debt*, 2012, pp. 7–14.
[56] R. Zhou, P. C. Sanza, and N. A. Visnevski, “Open modular embedded instrumentation architecture for test and evaluation,” in *Defense and Security Symposium*, 2006, pp. 624903–624903.
[57] M. Azmoodeh, N. Georgalas, and S. Fisher, “Model-driven systems development and integration environment,” *BT Technol. J.*, vol. 23, no. 3, pp. 96–110, 2005.
[58] T. Østerlie and A. I. Wang, “Debugging integrated systems: An ethnographic study of debugging practice,” in *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*, 2007, pp. 305–314.
[59] S. A. Hissam and D. Carney, “Isolating faults in complex COTS-based systems,” *J. Softw. Maint.*, vol. 11, no. 3, pp. 183–199, 1999.
[60] D. H. Kim, H. S. Son, W. Y. Kim, and R. Y. C. Kim, “Test Case Extraction for Intelligent Power Switch Heterogeneous Vehicles,” *Inf. Technol. Comput. Sci. ITCS*, pp. 82–88, 2012.
[61] J. Tian, “An Emerging Experience Factory to Support High-Quality Applications Based on Software Components and Services,” *J. Softw.*, vol. 6, no. 2, pp. 289–297, 2011.
[62] W. Jing, Z. Wu, H. Liu, and J. Dong, “An Effective Bi-Criteria and Contention Awareness Scheduling in Heterogeneous Distributed Systems,” *Intell. Autom. Soft Comput.*, vol. 17, no. 6, pp. 759–769, 2011.

[63] D. Hu, Z. Li, and Y. Ni, “Integration of Heterogeneous System Based on Business Object,” in *Management and Service Science, 2009. MASS’09. International Conference on*, 2009, pp. 1–4.
[64] K. M. Gee, “An architectural framework for integrating COTS/GOTS/legacy systems,” DTIC Document, 2000.
[65] A. Endo, A. D. S. Sim, S. Souza, and P. Souza, “Web services composition testing: a strategy based on structural testing of parallel programs,” in *Practice and Research Techniques, 2008. TAIC PART’08. Testing: Academic & Industrial Conference*, 2008, pp. 3–12.
[66] P. Mäki-Asiala and M. Matinlassi, “Quality assurance of open source components: integrator point of view,” in *Computer Software and Applications Conference, 2006. COMPSAC’06. 30th Annual International*, 2006, vol. 2, pp. 189–194.
[67] M. Mott, F. Khan, and others, “Integration of commercial-off-the-shelf (COTS) video stimulus instrument with fielded Automated Test System (ATS),” in *AUTOTESTCON Proceedings, 2000 IEEE*, 2000, pp. 539–543.
[68] M. N. Irfan, “Heuristics for Improving Model Learning Based Software Testing,” in *Testing: Academic and Industrial Conference-Practice and Research Techniques, 2009. TAIC PART’09.*, 2009, pp. 127–128.
[69] L. Mariani, S. Papagiannakis, and M. Pezze, “Compatibility and regression testing of COTS-component-based software,” in *Software Engineering, 2007. ICSE 2007. 29th International Conference on*, 2007, pp. 85–95.
[70] S. Beydeda, “Self-metamorphic-testing components,” in *Computer Software and Applications Conference, 2006. COMPSAC’06. 30th Annual International*, 2006, vol. 2, pp. 265–272.
[71] W.-L. Wang, D. Pan, and M.-H. Chen, “Architecture-based software reliability modeling,” *J. Syst. Softw.*, vol. 79, no. 1, pp. 132–146, 2006.
[72] S. Nabi and M. N. A. Khan, “An Analysis of Application Level Security in Service Oriented Architecture,” *Int. J. Mod. Educ. Comput. Sci. IJMECS*, vol. 6, no. 2, p. 27, 2014.
[73] K. C. Foo, Z. M. J. Jiang, B. Adams, A. E. Hassan, Y. Zou, and P. Flora, “An Industrial Case Study on the Automated Detection of Performance Regressions in Heterogeneous Environments.”
[74] S. S. Gokhale and K. S. Trivedi, “Analytical models for architecture-based software reliability prediction: A unification framework,” *Reliab. IEEE Trans. On*, vol. 55, no. 4, pp. 578–590, 2006.
[75] J.-H. Lo, S.-Y. Kuo, M. R. Lyu, and C.-Y. Huang, “Optimal resource allocation and reliability analysis for component-based software applications,” in *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*, 2002, pp. 7–12.
[76] A. Lacey and D. Luff, *Qualitative data analysis*. Trent Focus Sheffield, 2001.
[77] R. Feldt and A. Magazinius, “Validity Threats in Empirical Software Engineering Research-An Initial Survey.,” in *SEKE*, 2010, pp. 374–379.
[78] M. Niazi and M. A. Babar, “De-motivators of software process improvement: An analysis of vietnamese practitioners’ views,” in *Product-Focused Software Process Improvement*, Springer, 2007, pp. 118–131.

APPENDIX A: QUALITY CRITERIA

Id	paper tile	QC1: Does the reader able to understand the aims and objectives of the research?	QC2: Does the study performed clearly describes about third party components used in heterogeneous systems.	QC3: Does the study state the challenges faced while using third-party	QC4: Does the data collection method used in the study clearly described	QC5: Does the data collected relate to the aim of the research	QC6: Does the data collected is analysed based on the method described	QC7: Does the research conducted contribute to the software engineering field	QC8: Is the conclusion clearly stated and related to the aims and objective of the research	Score
S1	An approach to testing	1	1	1	1	1	1	1	1	8
S2	Spiral development of :	1	1	0	0	0	0	0	0	2
S3	Vertical software indus	1	1	0	0	1	1	1	1	6
S4	Visualising architectura	1	1	1	0	1	1	1	1	7
S5	Open modular embedd	1	0	1	0	0	0	1	1	4
S6	Model-driven systems	1	0	1	1	1	0	0	0	4
S7	Debugging Integrated S	1	1	1	1	1	0	1	1	7
S8	Isolating faults in comp	1	1	1	1	1	0	1	1	7
S9	Test Case Extraction f	1	1	0	0	1	1	1	1	6
S10	An Emerging Experienc	1	1	0	0	1	1	1	1	7
S11	An Effective Bi-Criteria	1	0	1	0	1	1	1	1	6
S12	Integration of Heteroge	1	0	0	0	1	0	1	1	4
S13	An architectural frame	1	1	1	1	1	0	1	1	7
S14	Web Services Compos	1	0	1	1	0	0	1	0	4
S15	Specification-Based V	1	1	1	1	1	1	1	1	8
S16	Heterogeneous Syster	1	1	1	1	1	1	1	1	8
S17	Quality Assurance of C	1	1	1	1	1	1	1	1	7
S18	Solving performance is	1	1	1	1	0	0	1	1	6
S19	Integration of commer	1	1	1	1	1	0	1	1	7
S20	A lightweight process f	1	1	1	1	0	0	1	1	6
S21	Heuristics for Improvin	1	1	1	0	1	0	1	0	4
S22	Compatibility and regre	1	1	1	1	1	1	0	1	7
S23	A Dimensionality Mod	1	1	1	1	1	1	0	1	7
S24	Relationships between	1	1	1	1	1	0	1	1	6
S25	Self-Metamorphic-Tes	1	1	1	1	1	0	1	0	6
S26	Behavior Capture and '	1	1	1	1	1	1	1	1	8
S27	Architecture-based sol	1	0	1	1	1	0	1	1	6
S28	An Analysis of Applica	1	1	0	1	1	1	1	1	7
S29	An Industrial Case Stuc	1	1	1	1	1	0	1	1	7
S30	Analytical Models for A	1	1	1	1	1	1	1	1	8
S31	Optimal Resource Allo	1	1	1	1	1	1	1	1	8

APPENDIX B: INTERVIEW QUESTIONNAIRE:

GENERAL

- What is your current role?
- How long you are working in this organization?
- How long it will take to work on required product to develop?
- How long in current role?

SPECIFIC ABOUT PROJET/PRODUCT

- What is the goal of your product being developed?
- Which of the following software development life cycle model is mostly used by your company?
  -Agile
  -Plan driven
  -Hybrid
- How many subsystems are present in the product being developed?
- How many people are involved in developing this product?
- What are the third party components in your product development?

Context:

- Are there any challenges faced while integrating your software product with third party components such as COTS,OSS, In-house software
- Are there any challenges faced while testing third party components in your systems on these levels
  ->System level testing
  ->Integration level testing
- What are the test design techniques used for testing of third party components in your systems at system and integration level of testing.
- Does test design differ while using different third party components(eg: COTS or OSS or in-house)
- What are the test activities used in testing of third party components in your system
- Are there any testing tools used for testing of third party components? If yes can you list some
- Is there any strategy followed while testing. If yes then what would be the difference while testing normal systems (systems without third-party components) and heterogeneous system using third party components.
- What are the recommendation that you can be provide while testing third party components