ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

The impact of requirements in globally distributed software development: An empirical study

Master thesis
263-0800-00L, Fall 2014

Marc Egg

supervised by

Dr. Martin Nordio
Dr. Christian Estler
Prof. Dr. Bertrand Meyer
Chair Of Software Engineering, ETH Zurich, Switzerland

December 2014 – June 2015

Abstract

Requirements describe the functionality and qualities a software product should have. In traditional local software development the quality of requirements impacts on software quality. Today software development is increasingly globally distributed. Are the findings regarding requirements in traditional local software development also true in globally distributed environments?

We carried out an empirical study using 68 globally distributed software development projects and analyzed the impact of requirements documents on software quality. Our findings include no statistically significant differences in software quality between projects with bad requirements and projects with good requirements documents. However authorship of requirements documents results in a statistically significant different amount of contracts being specified. Programmer-written requirements documents seem to result in more contracts being specified than requirements engineer-written requirements documents. As an extra analysis we compared projects with regard to national cultures and found no statistically significant differences in software quality.

Acknowledgments

Bertrand Meyer was the lecturer of my very first lecture as a computer science student. He remained a constant companion throughout my studies both as a lecturer and mentor. I appreciate his constant efforts to educate students in all fields of software engineering and I'm thankful for having had the chance to participate in many of his courses.

The attendance of courses given by Bertrand Meyer entailed getting to know many members of his chair, the *Chair of Software Engineering*. One of the members introduced me to *Martin Nordio* and *Christian Estler* after I mentioned my interest in *requirements* and *good design*. They told me about their idea to research the relationship between requirements and software quality with regard to globally distributed software development. Martin and Christian became the supervisors of my master thesis. The collaboration and discussion with them gave me many insights into software engineering and of course also *requirements* and *good design*. I'm also very grateful for their suggestions and guidance throughout this thesis.

This master thesis concludes my computer science studies. The successful completion is also due to the backing of *my family* which motivated me in challenging situations and cheered every success.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

The impact of requirements in globally distributed software development: an empirical study

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Egg	Marc

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, June 1 2015	M. Egg

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Background . . . 2
- 1.3 Overview . . . 2

2 Research questions and hypotheses **4**
- 2.1 Research questions . . . 4
- 2.2 Hypotheses . . . 5

3 Research methodology **8**
- 3.1 Introduction . . . 8
- 3.2 Origin of data . . . 8
- 3.3 Collecting and choosing data . . . 9
- 3.4 Defining quality of requirements documents . . . 17
- 3.5 Defining quality of software . . . 19
  - 3.5.1 Motivation . . . 19
  - 3.5.2 Coupling . . . 21
  - 3.5.3 Information hiding . . . 21
  - 3.5.4 Contracts . . . 21
  - 3.5.5 Eiffel Inspector . . . 22
  - 3.5.6 Relationship to globally distributed software development . . . 23
- 3.6 Determining quality of requirements documents . . . 23
- 3.7 Determining quality of software quality . . . 24
- 3.8 Summary . . . 27

4 Quantitative analysis and quantitative results **29**
- 4.1 Introduction . . . 29
- 4.2 Quantitative analysis . . . 29
- 4.3 Quantitative results for RQ.1 . . . 30
- 4.4 Quantitative results for RQ.2 . . . 37
- 4.5 Quantitative results for RQ.3 . . . 43
- 4.6 Correlations . . . 49

5 Related work **51**
- 5.1 Empirical studies on software quality . . . 51
- 5.2 Empirical studies relying on DOSE . . . 51
- 5.3 Teaching DOSE . . . 52

6 Conclusion **53**

A Additional quantitative results **54**
A.1 Research question RQ.1 54
A.1.1 Main analysis (2009 - 2012) 54
A.1.2 Main analysis (2013 - 2014) 59
A.1.3 40-20-40 analysis (2009 - 2014) 64
A.1.4 40-20-40 analysis (2009 - 2012) 69
A.1.5 40-20-40 analysis (2013 - 2014) 74
A.1.6 Time zone classification analysis (2009 - 2014) 79
A.1.7 Time zone classification analysis (2009 - 2012) 84
A.1.8 Time zone classification analysis (2013 - 2014) 89
A.2 Research question RQ.2 94
A.2.1 Main analysis (2009 - 2012 vs. 2013) 94
A.2.2 Main analysis (2009 - 2012 vs. 2014) 99
A.3 Research question RQ.3 104
A.3.1 Main analysis (2009 - 2012) 104
A.3.2 Main analysis (2013 - 2014) 109

B Normality tests **114**
B.1 Research question RQ.1 114
B.1.1 Main analysis (2009 - 2014) 114
B.1.2 Main analysis (2009 - 2012) 125
B.1.3 Main analysis (2013 - 2014) 136
B.1.4 40-20-40 analysis (2009 - 2014) 147
B.1.5 40-20-40 analysis (2009 - 2012) 158
B.1.6 40-20-40 analysis (2013 - 2014) 169
B.2 Research question RQ.2 180
B.2.1 Main analysis (2009 - 2012 vs. 2013 - 2014) 180
B.2.2 Main analysis (2009 - 2012 vs. 2013) 191
B.2.3 Main analysis (2009 - 2012 vs. 2014) 202
B.3 Research question RQ.3 213
B.3.1 Main analysis (2009 - 2014) 213
B.3.2 Main analysis (2009 - 2012) 224
B.3.3 Main analysis (2013 - 2014) 235

1 Introduction

1.1 Motivation

Requirements define the functionality and qualities a software product should have and are a much debated topic in the software engineering research community. Back in 1976 Bell et al. [7] provokingly asked whether requirements are really a problem. Their empirical study showed requirements are indeed a problem. The requirements examined in the study were incorrect, ambiguous, inconsistent or missing. Lutz [35] identified misunderstanding of requirements as the main source of safety-related functional faults in software of two NASA spacecrafts. Another study by Basili et al. [5] identified incorrect or misinterpreted functional specifications or requirements as the cause of 48 percent of the faults in medium-scale software projects. Thus, requirements have an impact in software development. The previously mentioned examples originate from the 1970s and 1980s and therefore relate to traditional local software development. Today software development is increasingly globally distributed [38, 33]. This raises a question: Are the previously mentioned findings also true in globally distributed software development?

We present the findings of our empirical study which examined the impact of requirements in globally distributed software development. The focus of our study is on highlighting the impact of requirements on software quality. Our data sample consists of 68 software development projects carried out in a globally distributed environment. The data sample originates from a university course [46, 47, 44] taught on multiple continents. As part of that university course students have to complete a globally distributed software development project which includes writing a requirements document and implementing the previously specified requirements. We assessed both the quality of the requirements and software products to determine the impact of requirements on software quality.

The data samples show evidence that the quality of requirements documents does not influence software quality but the authorship of requirements documents does. In particular, software products emerging from programmer-written requirements documents have a higher percentage of contracts than software products originating from requirements engineer-written requirements documents. As an extra analysis we compared projects with regard to national cultures and found no statistically significant differences in software quality.

1.2 Background

Globalization forces software development organizations to lower their cost to remain competitive [38, 33]. The consequence was the rise of globally distributed software development where the development of software partially or completely takes place in countries with low labor costs and high expertise such as India. These days collaborators do not necessarily sit next door but in another country or even on another continent.

The global distribution introduces new challenges [34, 38, 33] including communication, time zone difference, language and cultures. Separation by distance prevents spontaneous communication and sitting together to resolve merge conflicts [22] in a shared code base or to debug [21] source code. In general distance and be it only even 30 meters results in less frequent communication [2]. The time zone difference between collaborators reduces the amount of time to synchronously communicate which decreases the pace of information exchange [20]. Different native languages of the parties involved in globally distributed software development further increases the communication overhead [43]. Cultural differences occur in two manifestations: Difference in *organizational culture* namely how organizations develop software and differences in *national culture* such as language, values and norms [13]. Hall [31] classifies (national) culture into low-context cultures and high-context cultures. Communication between individuals of high-context cultures heavily relies on their shared cultural background in the sense that values and norms give the exchanged communication a meaning. Individuals of low-context cultures do not refer to their cultural background when communicating with each other and thus are very unmistakable.

The methods used in traditional local software development do not take into account these challenges found in globally distributed software development. For this reason it is necessary to verify the applicability of findings related to traditional local software development with regard to globally distributed software development. The software engineering research community continuously fills the knowledge gap (e.g. [57, 8, 14, 23, 48, 45]) introduced by the rise of globally distributed software development. We contribute our unprecedented empirical study about the impact of requirements in globally distributed software development.

1.3 Overview

The remaining sections have the following topics. *Section 2* explains the research questions along with the hypotheses which our empirical study shall answer. *Section 3* discusses the methodology used to perform the empiri-

cal study. *Section 4* introduces the quantitative analyses used to determine the impact of requirements and presents the quantitative results. *Section 5* highlights related work. *Section 6* summarizes the findings of the empirical study and identifies future work. *Appendix A* provides additional quantitative results. *Appendix B* contains all normality tests used in the quantitative analysis.

2 Research questions and hypotheses

2.1 Research questions

In order to learn more about the impact of requirements on software quality we defined the following three research questions which relate to different aspects of the potential impact of requirements on software quality.

RQ.1 What is the impact of bad and good requirements documents on the average coupling ratio, the median coupling ratio, the information hiding ratio, the percentage of routines with preconditions, the percentage of routines with postconditions, the percentage of classes with class invariants, the average percentage of contracts, the number of Eiffel Inspector warnings per 1000 lines of code (LOC), the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC?

RQ.2 What is the impact of programmer-written and requirements engineer-written requirements documents on the average coupling ratio, the median coupling ratio, the information hiding ratio, the percentage of routines with preconditions, the percentage of routines with postconditions, the percentage of classes with class invariants, the average percentage of contracts, the number of Eiffel Inspector warnings per 1000 LOC, the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC?

RQ.3 What is the impact of high- and mixed-context culture groups on the average coupling ratio, the median coupling ratio, the information hiding ratio, the percentage of routines with preconditions, the percentage of routines with postconditions, the percentage of classes with class invariants, the average percentage of contracts, the number of Eiffel Inspector warnings per 1000 LOC, the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC?

The third research question RQ.3 is not concerned with the impact of requirements on software quality. Given the availability of both graded requirements documents and software products we decided to run a quantitative analysis looking into the impact of culture on software quality as well. Subsection 4.5 explains why we compare projects with high- and mixed-context culture groups instead of projects with high- and low-context culture groups.

2.2 Hypotheses

Since each research question includes multiple subordinate questions we split the research questions into multiple hypotheses. To answer the research questions we deploy statistical tests which require the formulation of the hypotheses as null-hypotheses. A null-hypothesis assumes no statistically significant difference between the different observations. If a statistical test does not support the null-hypothesis, then it supports the alternative hypothesis. In our empirical study the alternative hypothesis is that there is a statistically significant difference.

Hypotheses H.1 to H.10 refer to the first research question RQ.1 which looks into the impact of requirements quality on software quality.

H.1 There is no difference in the average coupling ratio between projects with bad and good requirements documents.

H.2 There is no difference in the median coupling ratio between projects with bad and good requirements documents.

H.3 There is no difference in the information hiding ratio between projects with bad and good requirements documents.

H.4 There is no difference in the percentage of routines with preconditions between projects with bad and good requirements documents.

H.5 There is no difference in the percentage of routines with postconditions between projects with bad and good requirements documents.

H.6 There is no difference in the percentage of classes with class invariants between projects with bad and good requirements documents.

H.7 There is no difference in the average percentage of contracts between projects with bad and good requirements documents.

H.8 There is no difference in the number of Eiffel Inspector warnings per 1000 LOC between projects with bad and good requirements documents.

H.9 There is no difference in the number of Eiffel Inspector suggestions per 1000 LOC between projects with bad and good requirements documents.

H.10 There is no difference in the average number of Eiffel Inspector rule violations per 1000 LOC between projects with bad and good requirements documents.

Hypotheses H.11 to H.20 refer to the second research question RQ.2 which looks into the impact of requirements authorship on software quality.

H.11 There is no difference in the average coupling ratio between projects with programmer- and requirements engineer-written requirements documents.

H.12 There is no difference in the median coupling ratio between projects with programmer- and requirements engineer-written requirements documents.

H.13 There is no difference in the information hiding ratio between projects with programmer- and requirements engineer-written requirements documents.

H.14 There is no difference in the percentage of routines with preconditions between projects with programmer- and requirements engineer-written requirements documents.

H.15 There is no difference in the percentage of routines with postconditions between projects with programmer- and requirements engineer-written requirements documents.

H.16 There is no difference in the percentage of classes with class invariants between projects with programmer- and requirements engineer-written requirements documents.

H.17 There is no difference in the average percentage of contracts between projects with programmer- and requirements engineer-written requirements documents.

H.18 There is no difference in the number of Eiffel Inspector warnings per 1000 LOC between projects with programmer- and requirements engineer-written requirements documents.

H.19 There is no difference in the number of Eiffel Inspector suggestions per 1000 LOC between projects with programmer- and requirements engineer-written requirements documents.

H.20 There is no difference in the average number of Eiffel Inspector rule violations per 1000 LOC between projects with programmer- and requirements engineer-written requirements documents.

Hypotheses H.21 to H.30 refer to the third research question RQ.3 which looks into the the impact of culture on software quality.

H.21 There is no difference in the average coupling ratio between groups with only high- and mixed-context culture teams.

H.22 There is no difference in the median coupling ratio between groups with only high- and mixed-context culture teams.

H.23 There is no difference in the information hiding ratio between groups with only high- and mixed-context culture teams.

H.24 There is no difference in the percentage of routines with preconditions between groups with only high- and mixed-context culture teams.

H.25 There is no difference in the percentage of routines with postconditions between groups with only high- and mixed-context culture teams.

H.26 There is no difference in the percentage of classes with class invariants between groups with only high- and mixed-context culture teams.

H.27 There is no difference in the average percentage of contracts between groups with only high- and mixed-context culture teams.

H.28 There is no difference in the number of Eiffel Inspector warnings per 1000 LOC between groups with only high- and mixed-context culture teams.

H.29 There is no difference in the number of Eiffel Inspector suggestions per 1000 LOC between groups with only high- and mixed-context culture teams.

H.30 There is no difference in the average number of Eiffel Inspector rule violations per 1000 LOC between groups with only high- and mixed-context culture teams.

3 Research methodology

3.1 Introduction

Our empirical study shall answer the three previously defined research questions along with the 30 hypotheses capturing the impact on individual software qualities. In such a study the research methodology plays an import role.

Our first step was to collect all data necessary to perform the study. This includes information about the project setup, the people involved and the deliverables, namely the requirements documents and the software products. The next step was to clean the data in order to only look into projects with comparable settings. The third step was the most time consuming because we had to assess the quality of both the requirements documents and the software products. Afterwards, the quantitative analysis through statistical tests provided results which answer the three research questions.

The remainder of this chapter describes the research methodology in more detail and motivates the decisions we made.

3.2 Origin of data

The requirements documents and software products used in the empirical study originate from a university course named *Distributed and Outsourced Software Engineering* (*DOSE*). Members of the *Chair of Software Engineering* at the *Swiss Federal Institute of Technology Zurich* educate the enrolled students theoretically and practically in distributed and outsourced software engineering. The practical education takes place through a globally distributed software development project where the participating students work together with students from other universities all over the world that joined DOSE.

The globally distributed software development project the students participate in spans 13 weeks and includes several non-movable deadlines. The project follows a structured process defined by the teaching staff in advance. Among other tasks the process includes writing a requirements document for a given project description as well as implementing the requirements to create a software product. Each software product consists of various *subcomponents*. Table 1 summarizes for every edition of DOSE the topic as well as the individual subcomponents.

Throughout the years the subcomponents were almost always the same. *Logic* keeps track of the software product's state, *GUI* is the graphical user interface the human user interacts with, *Net* takes care of the communi-

Year	Topic	Subc. 1	Subc. 2	Subc. 3
2009	Card games	Logic	GUI & Net	
2010	Language learning tool	Logic	GUI	Store
2011	Card & board games	Logic	GUI & Net	AI
2012	Card & board games	Logic	GUI & Net	AI
2013	Card & board games	Logic	GUI & Net	AI
2014	Project-management tool	Back end	Front end	

Table 1: Topics and subcomponents of projects per year.

cation through networks, *Store* is responsible for storing data persistently on the hard drive, *AI* provides means to play the card and board games against computer opponents relying on artificial intelligence, *back end* is the server-part of the project-management tool while *front end* is the part of the project-management application the human user manipulates. The students implement the subcomponents in *Eiffel*. An exception is the *front end* in 2014 which the students had to implement using web technologies.

At least two students from the same university work together in a *team* and every team is responsible for one subcomponent. *Groups* consist of at least two teams and produce the overall software product formed by the individual subcomponents. Thus every group work on their own *project*. We use groups and projects interchangeably.

Between 2009 and 2012 the process expected every team to write the requirements document for their assigned subcomponent and to implement the requirements defined in their own requirements document. In the 2013 and 2014 editions of DOSE the process changed and was slightly different. Groups still consist of teams but teams had *roles*. One team of a group had the role of the *requirements engineers* who write the requirements document for the whole software product and thus all subcomponents. The remaining teams in the group had the role of the *programmers* who implement the requirements defined by the requirements engineers for their assigned subcomponent.

3.3 Collecting and choosing data

The students had to use versioning systems to share their work with the other students and to submit their deliverables to the teaching staff. Wikis stored the administrative information such as the composition of teams, the composition of groups and the mapping of teams to subcomponents.

First we retrieved all data available from these two previously mentioned

Year	Groups	Selected groups		Excluded groups	
		#	%	#	%
2009	8	8	100	0	0
2010	11	9	81.82	2	18.18
2011	12	10	83.33	2	16.67
2012	21	21	100	0	0
2013	12	12	100	0	0
2014	8	8	100	0	0
All	72	68	94.44	4	5.56

Table 2: The number (#) and percentage (%) of selected and excluded groups per year. In 2010 the author participated in a group. We excluded this group to avoid any bias. The other exclusions relate to unstable groups.

sources and then collected the information regarding the participating universities, teams and groups along with team sizes, group sizes and mappings of teams to subcomponents. Every group which participated in DOSE received a unique identifier in the form of a natural number denoting the global group number. Then we collected the available requirements documents and software products.

The previously compiled information made us aware of the problems some groups suffered. As a result we defined the criterion both requirements documents and software products had to fulfill for the participation in the study. The criterion is that the deliverables have to originate from *stable groups*. A group is stable if and only if the individual teams of the group worked together for the *full* duration of the project and submitted *all* required deliverables namely requirements documents and software products. More insights into the number and percentage of selected and excluded groups per year gives Table 2. In 2010 we excluded two groups. The first group did not provide an implementation for the Store subcomponent. The second group had the author as a member. Another two excluded groups originate from the 2011 edition of DOSE. One group dissolved and the other group did not submit a software product. From 72 available projects we excluded four projects for the previously mentioned reasons. This leaves us with 68 projects usable for our empirical study. We decided to include in our empirical study only the *back end* of the projects done in 2014 because the *front end* uses web technologies which are not comparable to subcomponents written in Eiffel.

Having now selected the projects used for the study we are able to report the characteristics of our data sample with regard to the participating universities, the number of groups, the number of teams, the number of teams

per group, the number of students per team and the time zone difference between teams involved in our study. Table 3 lists the universities which participated in DOSE. Most universities originate from Europe (11 universities), followed by Asia (4 universities), South America (2 universities), Africa (1 university) and Australia (1 university). Detailed information about the number of groups, number of teams and the number of students gives Table 4. Over the years the number of groups oscillates around ten with the exception of 2012 whereas the number of teams and students is subject to large changes. Table 5 reports the detailed numbers related to the group size and shows that groups usually consist of two or three teams. Table 6 summarizes the detailed numbers related to the team size and shows that most frequently two to four students form a team. For the 2011 edition of DOSE Table 5 shows there are two groups with two teams although Table 1 indicates three subcomponents per projects for the 2011 edition of DOSE. Our investigation revealed that one group split *GUI & Net* between the two teams and in another group a team implemented both *Logic* and *AI*. It was not possible to recover the reasons for these decisions but both groups comply with our definition of stable groups. This is why we selected them as well for our study.

Previous tables listed the participating universities per year and the details of the groups' and the teams' compositions. Table 7 summarizes per year the affiliation of students to universities. The numbers in parentheses indicate the number of teams originating from the corresponding university. As can be seen in Table 7 we report the number of students affiliated with the *Swiss Federal Institute of Technology Zurich* and *University of Zurich* in the same row since students from University of Zurich join teams from the Swiss Federal Institute of Technology Zurich. Major suppliers of participants are the universities regularly participating in DOSE. These include National University of Rio Cuarto, Polytechnic University of Milan, Swiss Federal Institute of Technology Zurich and University of Zurich, Pontifical Catholic University of Rio Grande do Sul, Lobachevsky State University of Nizhny Novgorod, University of Debrecen and Technical University of Madrid. Both IT University of Copenhagen and University of Crete supply a large number of students one time only.

One of the challenges in globally distributed software development is the difference in time zones. To measure the geographical distribution we classified groups into three classes according to the maximum time zone difference between teams of a group: Small, medium and large. A *small* time zone difference is one where the maximum time zone differences between teams of a group is less than or equal to three hours. A *medium* time zone difference is one where the maximum time zone differences between teams of

University	City	Country	2009	2010	2011	2012	2013	2014
Polytechnic University of Milan	Milan	Italy	✓	✓	✓	✓	✓	✓
Swiss Federal Institute of Technology Zurich	Zurich	Switzerland	✓	✓	✓	✓	✓	✓
University of Zurich	Zurich	Switzerland	✓	✓	✓	✓	✓	✓
Lobachevsky State University of Nizhny Novgorod	Nizhny Novgorod	Russia	✓	✓	✓	✓	✓	
National University of Rio Cuarto	Rio Cuarto	Argentina		✓	✓	✓	✓	✓
Odessa National Polytechnic University	Odessa	Ukraine	✓	✓	✓	✓		
Technical University of Madrid	Madrid	Spain			✓	✓	✓	✓
University of Debrecen	Debrecen	Hungary	✓	✓	✓	✓		
Hanoi University of Science and Technology	Hanoi	Vietnam	✓	✓	✓			
Pontifical Catholic University of Rio Grande do Sul	Porto Alegre	Brazil				✓	✓	✓
Cairo University	Cairo	Egypt				✓	✓	
ITMO University	St. Petersburg	Russia			✓	✓		
IT University of Copenhagen	Copenhagen	Denmark			✓		✓	
Innopolis University	Kazan	Russia						✓
Korea Advanced Institute of Science and Technology	Daejeon	South Korea		✓				
University of Adelaide	Adelaide	Australia					✓	
University of Crete	Heraklion	Greece				✓		
University of Delhi	Delhi	India		✓				
Wuhan University	Wuhan	China		✓				

Table 3: Universities participating in the different editions of DOSE.

Year	#Groups	#Teams	#Students
2009	8	16	52
2010	9	27	86
2011	10	28	89
2012	21	50	146
2013	12	48	171
2014	8	24	70
All	68	193	614

Table 4: Number of groups, number of teams and number of students participating in the different editions of DOSE. The numbers for 2014 include the teams and students working on the front end as well.

Year	#Groups	#Groups with		
		2 teams	3 teams	4 teams
2009	8	8		
2010	9		9	
2011	10	2	8	
2012	21	13	8	
2013	12			12
2014	8		8	
All	68	23	33	12

Table 5: The number of groups and the group sizes participating in the different editions of DOSE. The numbers for 2014 include the teams working on the front end as well.

Year	#Teams	#Teams with			
		2 students	3 students	4 students	5 students
2009	16		12	4	
2010	27	2	20	3	2
2011	28	4	16	7	1
2012	50	12	30	8	
2013	48	8	14	17	9
2014	24	5	16	3	
All	193	31	108	42	12

Table 6: The number of teams and the team sizes participating in the different editions of DOSE. The numbers for 2014 include the students working on the front end as well.

a group is greater than three hours but less than or equal to five hours. A *large* time zone difference is one where the maximum time zone differences between teams of a group is more than five hours. We report the groups' time zone classifications in Table 8. In general the time zone classifications are almost even. The largest difference in time zones occurred in 2013 where two groups with teams from Rio Cuarto (Argentina) and Adelaide (Australia) had a maximum time zone difference of 13.5 hours. Other large time zone differences occurred between teams in Rio Cuarto (Argentina) and Daejeon (South Korea), Hanoi (Vietnam) and Wuhan (China).

University	All years	2009	2010	2011	2012	2013		2014	
						Req.	Prog.	Req.	Prog.
National University of Rio Cuarto	104 (29)		12 (4)	18 (5)	32 (9)		21 (5)		21 (6)
Polytechnic University of Milan	82 (28)	14 (4)	9 (3)	19 (6)	21 (8)		10 (4)		9 (3)
IT University of Copenhagen	71 (18)			9 (4)			62 (14)		
Swiss Federal Institute of Technology Zurich / University of Zurich	70 (25)	15 (5)	20 (7)	11 (4)	7 (3)		13 (4)		4 (2)
Pontifical Catholic University of Rio Grande do Sul	51 (19)				17 (6)	21 (8)		13 (5)	
Lobachevsky State University of Nizhny Novgorod	34 (10)	6 (2)	5 (2)	7 (2)	8 (2)		8 (2)		
University of Crete	33 (12)				33 (12)				
University of Debrecen	30 (10)	6 (2)	9 (3)	3 (1)	12 (4)				
Technical University of Madrid	24 (9)			6 (2)	4 (2)		5 (2)		9 (3)
Hanoi University of Science and Technology	24 (6)	8 (2)	8 (2)	8 (2)					
Cairo University	20 (7)				5 (2)		15 (5)		
University of Adelaide	16 (4)					16 (4)			
Innopolis University	14 (5)							9 (3)	5 (2)
Odessa National Polytechnic University	12 (3)	3 (1)		5 (1)	4 (1)				
Wuhan University	10 (3)		10 (3)						
Korea Advanced Institute of Science and Technology	10 (2)		10 (2)						
ITMO University	6 (2)			3 (1)	3 (1)				
University of Delhi	3 (1)		3 (1)						

Table 7: Affiliation of students and teams with universities in the different editions of DOSE. The numbers in parentheses represent the number of teams.

Year	#Groups	Time zone classification		
		#Small	#Medium	#Large
2009	8	6		2
2010	9	1	1	7
2011	10	5	3	2
2012	21	10	7	4
2013	12		6	6
2014	8	1	3	4
All	68	23	20	25

Table 8: Number of groups classified by the maximum time zone difference between the teams (small: 0 - 3 hours, medium: 3 - 5 hours, large: greater than 5 hours).

3.4 Defining quality of requirements documents

When writing the requirements documents students had to follow *IEEE Std 830-1998* [1] which is the *IEEE Recommended Practice for Software Requirements Specifications*. The standard suggests several suitable structures for requirements documents along with recommendations what each section of the requirements document should cover. The standard also defines the qualities a *good* requirements document should have. These qualities [1] are correctness, unambiguity, completeness, consistency, prioritization, verifiability, modifiability and traceability.

We graded the requirements documents following a grading scheme developed by the Chair of Software Engineering to grade requirements documents written as part of mandatory course work. The grading scheme builds upon the quality attributes defined by IEEE Std 830-1998 and is subject to frequent refinement. The used grading scheme awards points for the following qualities which a requirements document should have:

- Completeness of functional requirements (8 points)
- Relevance of functional requirements (8 points)
- Description of a useful system by functional requirements (8 points)
- Existence and quality of GUI mock ups (8 points)
- Level of detail of requirements (8 points)
- Level of abstraction of requirements (8 points)
- Prioritization of requirements (8 points)
- Consistency of requirements document (8 points)
- Verifiability of functional requirements (8 points)
- Specification of verifiable, justified and relevant quality measures for non-functional requirements (8 points)
- Formatting of requirements document (4 points)
- Readability of requirements document (4 points)
- Scope of glossary (4 points)
- Precision and understandability of language (4 points)

- Length of document (4 points)

Hereinafter we briefly explain the individual graded qualities. Functional requirements should be complete in the sense that they describe all customer required functionality. Relevant functional requirements describe only mandatory functionality and ignore optional functionality if not marked as optional. The grading scheme awards points for describing a useful system despite the potential incompleteness of functional requirements. If the requirements describe a graphical user interface, then the grading scheme awards points for the existence and quality of mock ups visualizing the planned GUI. Then each requirement should be detailed enough to be self-contained. At the same time the requirements should be abstract and not describe implementation details. The priorities of requirements define the order of implementation and thus also the level of importance within the requirements. The requirements document should be consistent in the sense that the document uses terminology in a consistent and understandable way. A verifiable functional requirement allows the judgement if the software product contains the functionality described by it. Non-functional requirements define the qualities of the software product to be developed and therefore they should relate to the appropriate quality measures which allow the judgement if the software product has the specified quality. The formatting of the requirements document should facilitate reading by appropriately structuring the document and requirements. The requirements document is readable when its orthography and grammar are correct. A glossary should define the terminology used in the requirements documents and allows readers to read the document despite their potential lack of used terminology. The language in the document should be precise and understandable such that it allows the verification of requirements. The document should have a reasonable scope but of course it should not contain irrelevant topics.

The total score a requirements document can have is 92 or 100 points. Requirements document describing the whole project, the subcomponents *GUI* or the subcomponent *GUI & Net* can have a maximum score of 100 points. Otherwise the requirements document can have a maximum score of 92 points.

Awarding points using the grading scheme in a straightforward way results in inappropriately high total scores. If the requirements of a requirements document are very incomplete but the requirements document is of excellent quality, then the straightforward application of the grading scheme results in a high total score. In fact the requirements document does not deserve such a high total score. This is why we scale the maximum number of points awarded by the qualities with 8 points and 4 points depending on

Completeness (#Points)	Maximum #Points	
	8 points qualities	4 points qualities
8	8	4
7		
6		
5	6	3
4		
3	4	2
2		
1		
0	2	1

Table 9: Scaling of maximum number of points for qualities with 8 points and 4 points depending on number of points given for completeness.

the number of points the requirements document receives for *completeness*. Table 9 shows the exact scaling. If for example a requirements document is not very complete and receives 3 points for completeness, then it can receive a maximum number of 4 points for the qualities being worth of 8 points respectively 2 points for qualities being of worth 4 points. This approach ensures meaningful total scores.

3.5 Defining quality of software

3.5.1 Motivation

The definition of software quality is another much debated topic in the software engineering research community. Early contributors to the discussion were McCall et al. [36] and Boehm et al. [10]. In 1977 McCall et al. [36] defined quality factors along with criteria to be fulfilled. According to them there exist 11 quality factors namely correctness, reliability, efficiency, integrity, usability, maintainability, flexibility, testability, portability, reusability and interoperability. There are too many criteria to list but for example the criteria related to correctness are traceability, consistency and completeness. One year later in 1978 Boehm et al. [10] proposed a *software quality characteristics tree* with similar qualities as McCall et al. [36]. Despite their age Pressmann [52] notes that the software quality factors suggested by McCall et al. [36] are still valid today.

However, both Boehm et al. [9] and McCall et al. [15] acknowledge the difficulties in measuring some of their proposed software qualities. For example McCall et al. [36] define usability as the "*effort required to learn, operate,*

prepare input, and interpret output of a program" but usability is subject to the perception of individuals [41]. The customer of a software product wants the product to be portable between different operating environments while the programmer of the software product desires extensive documentation of its source code. First is the external view on software quality and the later is the internal view on software quality [41]. As we have seen external software quality is difficult to measure [9, 15]. To overcome the difficulties in measuring external software qualities Kitchenham [55] suggests to measure internal software qualities in order to predict external software qualities. One popular contribution related to the measurement of (internal) software quality is the metrics suite by Chidamber et al. [16]. The metrics suite consists of six metrics namely weighted methods per class, depth of inheritance tree, number of children, coupling between object classes, response for a class and lack of cohesion in methods.

We decided to focus our empirical study on internal software quality. To assess the (internal) quality of the software products we chose the following static metrics which are easy to understand and easy to compute:

- Average coupling ratio
- Median coupling ratio
- Information hiding ratio
- Percentage of routines with preconditions
- Percentage of routines with postconditions
- Percentage of classes with class invariants
- Average percentage of contracts
- Number of Eiffel Inspector Warnings per 1000 LOC
- Number of Eiffel Inspector Suggestions per 1000 LOC
- Average number of Eiffel Inspector rule violations per 1000 LOC

Hereinafter, we motivate the choice of the previously listed metrics and explain their meaning.

3.5.2 Coupling

Coupling refers to the reliance of a class on other classes. A class C is coupled to class C' when class C relies on features offered by class C'. The more class C relies on features offered by class C', the higher the coupling of class C is. Due to their reliance on other classes highly coupled classes are difficult to analyze, maintain, modify, reuse, test and understand [26, 25]. The community agrees that low coupling of classes is highly desirable [26, 42, 25, 41, 52, 51]. We use the definition of Trudel et al. [59] to measure coupling. Trudel et al. defined the coupling ratio of a class C as the number of *out accesses* to the number of *in accesses*. Out accesses are accesses to features defined in a class other than C while in accesses are accesses to features defined inside C. The definition ignores repeated accesses to the same feature within a routine's body. The coupling ratio is a positive real number greater than or equal to zero. We measure both the average and median coupling ratio of classes per project.

3.5.3 Information hiding

Parnas [50] proposes to design modules in such a way that modules hide design decisions from other modules. Parnas calls this approach *information hiding*. The benefit [50] of hiding design decisions results in the possibility to revisit design decisions without affecting clients of the module. Information hiding also reduces the amount of information a client needs to understand when using a interface[42] since the client just needs to understand how to use the interface while at the same time the client can ignore the implementation details. As with coupling the community strongly suggests to apply information hiding [52, 42, 40, 41, 26]. In Eiffel programmers can apply information hiding by restricting the export status of features [40]. Aside from the coupling ratio Trudel et al. [59] gave a definition of the information hiding ratio as well: It is the ratio of *non-exported features* to *exported features*. A feature is *exported* if and only if its export status is not constrained [40]. The information hiding ratio is a positive real number greater than or equal to zero. Note that above definition of the information hiding ratio refers to projects and not classes.

3.5.4 Contracts

The Eiffel programming language allows programmers to specify contracts, that is preconditions, postconditions and class invariants. For Meyer [40] contracts are "*a key element of software development in Eiffel*" and play an important role in *Design by contract* [39, 41]. Design by contract is a

collection of methodological principles to develop reliable software by seeing contracts as business contracts between contracting parties. Each contracting party has rights and obligations. With regard to software it means that the caller and supplier of a routine enter into a contract. If the caller fulfills the routines' preconditions, then the supplier guarantees the fulfillment of the postconditions. The benefit of writing contracts, that is preconditions, postconditions and invariants, is manifold. Contracts serve not only as formal specifications [42] but also document interfaces [42, 40], help with testing [40] and ease debugging [42, 40]. With contracts playing an important role in Eiffel we decided to use the percentage of routines with preconditions, the percentage of routines with postconditions, the percentage of classes with class invariants and the average percentage of contracts as metrics. The average percentage of contracts is the average of the three other metrics. Since these are all percentages the values lie between zero and 100 and are real numbers.

3.5.5 Eiffel Inspector

Continuous exposure to source code shapes a programmer's intuition of problematic implementation patterns. A typical example of such a problematic implementation pattern is the comparison of a Boolean variable to a Boolean constant (i.e. `is_empty = True`). Fowler et al. [24] call these problematic implementation patterns "*bad smells*". Among other examples they list duplicate code, long features and large classes as bad smells. Problematic implementation patterns entail a number of problems [24]: They reduce the quality of the software's design, complicate the understanding of source code and slow down development.

Eiffel Inspector [60] is a rule-based static code analysis tool which finds such problematic implementation patterns in Eiffel source code. Rules define the problematic implementation patterns and the Eiffel Inspector checks the source code against the rules. Once the Eiffel Inspector is able to match source code against a rule, that is an occurrence of such a problematic implementation pattern, it reports a rule violation indicating the source code which violates the rule. Every rule has a severity level. The current collection of rules comprises two levels of severity: Warnings and suggestions. We use the number of Eiffel Inspector warnings per 1000 LOC, the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC. The average number of Eiffel Inspector rule violations per 1000 LOC is the average of the other two metrics. The values from these three metrics are positive real numbers greater than or equal to zero.

3.5.6 Relationship to globally distributed software development

Our justification of the metrics' choice relies on fundamental software engineering principles. But do these metrics relate to globally distributed software development? Yes, they do!

Battin et al. [6] report their issues in a globally distributed software development project and how they dealed with these issues. The project included teams located in China, United States of America, Japan, India, Singapore and Australia. Their key principles to cope with the complexity of the projects' software architecture were *low coupling* and *well-defined interfaces* [6]. Our metrics related to coupling include the average coupling ratio and the median coupling ratio. Our metrics related to well-defined interfaces are the information hiding ratio, the percentage of routines with preconditions, the percentage of routines with postconditions, the percentage of classes with class invariants and the average percentage of contracts. The information hiding ratio states how well-defined interfaces are and the metrics related to contracts do the same as well since contracts are a helpful tool to document interfaces [42, 40].

Thus our choosen metrics do not only rely on fundamental principles of software engineering but also play an important role in globally distributed software development.

3.6 Determining quality of requirements documents

Since every human judgement is subject to a certain bias we had to take precautions to reduce the possibility for bias. The previously mentioned global group number is a unique identifier for every group and thus also project. We permuted the global group numbers and stored the mapping in a document unavailable to the author. The next steps were to print all requirements documents, to write the permuted global group numbers onto the title page of every single requirements document and to remove all occurrences related to year, country, city, university, group numbers, team names and student names in the requirements documents to make sure these do not influence the grading of the requirements documents. The author was never part of any of these anonymization activities. We then graded the documents from 2009 to 2012 per subcomponent using multiple iterations. Then we graded the documents from 2013 followed by the ones fro 2014. This approach prevented the author from establishing any connection between documents.

Table 10 summarizes the number of requirements documents and gives the statistics for the number of pages. Table 11 summarizes the number of

Year	#Docs.	#Pages					
		Min	**Median**	**Mean**	**Max**	**Total**	**SD**
2009	16	22	47	41.25	55	330	12.89
2010	27	44	68	70.11	105	631	21.38
2011	28	31	43.5	52	128	520	27.76
2012	50	22	36	39.29	76	825	15.08
2013	12	12	28	34	75	408	20.15
2014	8	16	22	26.38	41	211	9.96
Total	141	12	39	43.01	128	2925	21.99

Table 10: Number of requirements documents and number of pages described through the minimum number of pages (Min), the median number of pages, the mean number of pages, the maximum number of pages (Max), the total number of pages and the standard deviation of the number of pages (SD) for every edition of DOSE.

requirements documents and gives the statistics for the number of requirements. We graded a total number of 141 requirements documents with a total number of 2925 pages containing 4769 requirements. The number of requirements documents per year is coupled to the number of teams (2009 - 2012) respectively the number of groups (2013 & 2014). This explains the varying number of requirements documents per year. The topics used explain the differences in the number of requirements and pages per year.

3.7 Determining quality of software quality

Determining the quality of the software products was the next step in our grading activities. We used EiffelStudio 15.01.9.6535 to calculate the static metrics. Due to the ongoing refinement of the Eiffel programming language as well as Eiffel core libraries such as *EiffelBase* and *Gobo Eiffel Project* it was not possible to compile the software products without making any adjustments to the source code. Typical compiler errors of the unedited source code of projects include among others outdated paths in project files, missing calls to specific creation procedures, missing renamings of inherited features which resulted in multiple features of the same name being defined in a class, missing select clauses due to the diamond problem introduced by multiple inheritance, typing problems due to missing string conversions and usage of unsupported characters in source code. Simple and straightforward changes in the source code solved the failing compilation processes. Since every group produced their own software product we measure software quality

Year	#Docs.	#Reqs.					
		Min	**Median**	**Mean**	**Max**	**Total**	**SD**
2009	16	36	77.5	74.5	115	596	25.76
2010	27	84	101	104.89	144	944	19.61
2011	28	51	72.5	92	246	920	56.47
2012	50	16	66	65.57	121	1377	28.88
2013	12	20	49.5	63.75	145	765	42.03
2014	8	8	18	20.88	36	167	9.31
Total	141	8	69	70.13	246	4769	40.39

Table 11: Number of requirements documents and number of requirements described through the minimum number of requirements (Min), the median number of requirements, the mean number of requirements, the maximum number of requirements (Max), the total number of requirements and the standard deviation of the number of requirements (SD) for every edition of DOSE.

per group.

Three tools measure the metrics which we have chosen. We implemented a tool to measure the coupling ratio. The tool examines abstract syntax trees of classes using the visitor pattern [25] to count the out and in accesses. The other two tools are part of EiffelStudio. The integrated metric tool allowed us to compute the information hiding ratio, the percentage of routines with preconditions, the percentage of routines with postconditions and the percentage of classes with class invariants. The metric tool ignored source code not written by the students. An earlier subsection introduces the Eiffel Inspector which also is part of EiffelStudio. Using the Eiffel Inspector we were able to compute the number of Eiffel Inspector warnings per 1000 LOC and the number of Eiffel Inspector suggestions per 1000 LOC. We activated all predefined Eiffel Inspector rules. The metrics calculated with the metric tool and the Eiffel Inspector allow us to compute both the average percentage of contracts and the average number of Eiffel Inspector rule violations per 1000 LOC.

We continue with the discussion of the source code statistics. Table 12 summarizes the number of software products and gives the statistics for the number of LOC. Table 13 summarizes the number of software products and gives the statistics for the number of classes.

As before with the number of requirements documents per year the number of projects correlates with the number of groups per year. Our study includes 68 projects consisting of a total number of 3161 classes contain-

Year	#Prod.	#LOC					
		Min	**Median**	**Mean**	**Max**	**Total**	**SD**
2009	8	4545	6393.5	6566.38	8386	52531	1317.16
2010	9	8733	14514	13259.67	15909	119337	2458.12
2011	10	2342	9428.5	11197.4	20465	111974	5882.87
2012	21	1969	5578	6011.05	14692	126232	3070.5
2013	12	6119	10803.5	11421.17	20112	137054	4017.07
2014	8	1050	3332	3648.38	7236	29187	1972.57
All	68	1050	7644.5	8475.22	20465	576315	4719.88

Table 12: Number of software products and number of lines of code (LOC) described through the minimum number of LOC (Min), the median number of LOC, the mean number of LOC, the maximum number of LOC (Max), the total number of LOC and the standard deviation of the number of LOC (SD) for every edition of DOSE.

Year	#Prod.	#Classes					
		Min	**Median**	**Mean**	**Max**	**Total**	**SD**
2009	8	21	39.5	43.62	79	349	17.49
2010	9	39	59	64.78	105	583	18.69
2011	10	16	41	52.5	121	525	33.37
2012	21	18	30	37.71	88	792	18.75
2013	12	40	61.5	63.33	96	760	15.77
2014	8	10	13	19	42	152	12.51
All	68	10	43.5	46.49	121	3161	24.53

Table 13: Number of software products and number of classes described through the minimum number of classes (Min), the median number of classes, the mean number of classes, the maximum number of classes (Max), the total number of classes and the standard deviation of the number of classes (SD) for every edition of DOSE.

ing a total number of 576315 lines of code. Regarding both the number of lines of code and classes we observe variations between both the individual years and within years. The project setting, namely topics and number of subcomponents, influences these metrics.

3.8 Summary

Our study uses data gathered from a globally taught university course where students carried out a globally distributed software development project. These projects included writing a requirements document based on IEEE Std 830-1998 and implementing the requirements to create a software product using the Eiffel programming language. We collected these two deliverables and graded both. For the requirements document we anonymized the requirements document to prevent any bias originating from the knowledge of year and universities involved and then graded the anonymized requirements documents following a grading scheme influenced by IEEE Std 830-1998. The usage of straightforward source code modifications to the students' source code allowed us to use tools to calculate the quality metrics we have defined earlier.

Then we reported basic statistics about the people, requirements documents and implementations involved in our study. Table 14 summarizes previously presented numbers. The study comprises of students being affiliated with 19 different universities. The 614 students worked on 68 projects in 68 groups consisting of 193 teams. Teams from two, three or four locations worked together. The 614 students wrote 2925 pages of requirements documents containing 4769 requirements whose implementations result in 3161 classes consisting of 576315 LOC. The scope of all our data makes us confident to be able to answer our research questions.

	All years	2009	2010	2011	2012	2013	2014
#Universities		19 7	11	11	12	10	7
#Groups / #Projects	68	8	9	10	21	12	8
#Teams	193	16	27	28	50	48	24
#Students	614	52	86	89	146	171	70
#Locations per project	2 / 3 / 4	2	3	2 / 3	2 / 3	3 / 4	3
#Pages (average)	43.01	41.25	70.11	52	39.29	34	26.38
#Pages (total)	2925	330	631	520	825	408	211
#Requirements (average)	70.13	74.5	104.89	92	65.57	63.75	20.88
#Requirements (total)	4769	596	944	920	1377	765	167
#LOC (average)	8475.22	6566.38	13259.67	11197.4	6011.05	11421.17	3648.38
#LOC (total)	576315	52531	119337	111974	126232	137054	29187
#Classes (average)	46.49	43.62	64.78	52.5	37.71	63.33	19
#Classes (total)	3161	349	583	525	792	760	152

Table 14: Statistics of the scope of our empirical study.

4 Quantitative analysis and quantitative results

4.1 Introduction

Now with the research methodology being described we can move on to answer the reason questions RQ.1, RQ.2 and RQ.3 through the quantitative analysis of the hypotheses H.1 to H.30. First the focus is on the description of the quantitative analysis and then in the remainder of this section we present and discuss for each research question the quantitative results.

4.2 Quantitative analysis

All of our previously defined research questions RQ.1, RQ.2 and RQ.3 compare one individual characteristic of the projects which has two manifestations. Research question RQ.1 compares software quality of projects with bad requirements documents to those projects with good requirements documents. The second research question RQ.2 looks into the differences in software quality between projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents. The last research question RQ.3 compares software quality of projects with groups consisting of only high-context culture teams to projects with groups consisting of mixed-context culture teams.

The appropriate quantitative analysis [3] for these three research questions are two particular statistical tests: The *Mann-Whitney U-test* and the *t-test*. Hereinafter we summarize the basic properties [3] of these tests. The *Mann-Whitney U-test* or simply *U-test* is a *non-parametric* statistical test to compare the difference between two data samples for statistical significance. Non-parametric statistical tests do not impose any restriction on the distribution of the data sample whereas *parametric* statistical tests do. The t-test is a parametric statistical test and requires the data sample to follow a normal distribution and have same variance. Despite the limited applicability the t-test is more powerful than the U-test and even tolerates outliers in a data sample following a normal distribution. *Welch's t-test* in contrast to the t-test does not assume same variance of the data samples. In our quantitative analysis we use both the *U-test* and *Welch's t-test* to find statistically significant differences. For the sake of simplicity and readability we call Welch's t-test simply t-test throughout the remainder of this document.

Given the limited applicability of the t-test we run normality tests before the quantitative analysis to find the metrics which follow a normal distribution. Appendix B contains all results of the normality tests. In order

#	Min	Q_1	Median	Mean	Q_3	Max
68	9	35.75	50	48.51	59	85

Table 15: Summary of scores of requirements documents using the number of documents (#), minimum (Min), first quartile (Q_1), median, mean, third quartile (Q_3) and maximum (Max) score.

to test the metrics for normality we compare the density function with the plotted normal distribution having the same mean and standard deviation as the data sample, inspect the corresponding quantile-quantile plot (Q-Q plot) and run a Shapiro-Wilk normality test. In the tables summarizing the results of the statistical tests a dagger (†) marks normally distributed metrics.

As previously described we use statistical tests to find statistically significant differences in software quality. In our study we use a significance level α of $\alpha = 0.05$. A statistical test rejects the hypothesis if the test approves the hypothesis with a probability $p < \alpha$. In this case the statistical test finds a statistically significant difference. We mark statistically significant results with an asterisk ($*$).

For the quantitative analysis we used R 2.13.1 and RStudio 0.98.1087.

4.3 Quantitative results for RQ.1

The first research question looks into the impact of the quality of requirements on software quality.

RQ.1 What is the impact of bad and good requirements documents on the average coupling ratio, the median coupling ratio, the information hiding ratio, the percentage of routines with preconditions, the percentage of routines with postconditions, the percentage of classes with invariants, the average percentage of contracts, the number of Eiffel Inspector warnings per 1000 LOC, the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC?

The research question requires the classification of the requirements documents as either bad or good. As previously described in section 3.2 for each project between 2009 and 2012 one requirements document exists per subcomponent. In these cases we combined the individual scores of the requirements documents into a single score by averaging the individual scores.

Taking into account the number of projects, the distribution of the average scores (Figure 1) and the proximity of the median and mean score (Table

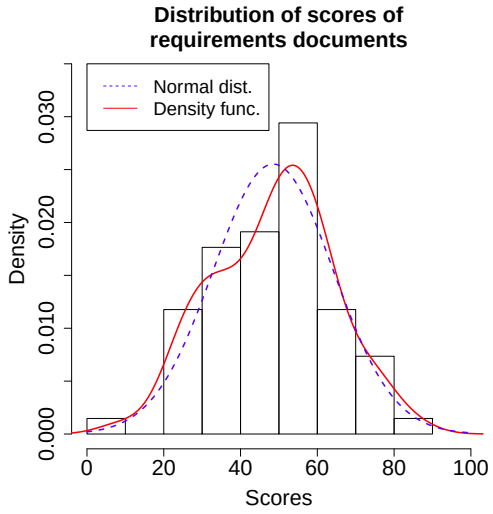


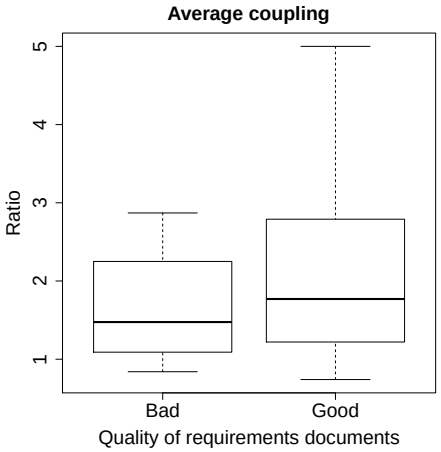
Figure 1: Histogram of all scores of requirements documents.

15) we decided to use the score 50 as a threshold to split the projects in two, namely projects with bad requirements documents and projects with good requirements documents. Doing so classifies 34 projects as projects with bad requirements documents and another 34 projects as projects with good requirements documents (Table 16). The visualization of all metrics using box plots in Figure 2, 3 and 4 does not suggest any difference in software quality between projects with bad requirements documents and projects with good requirements documents.

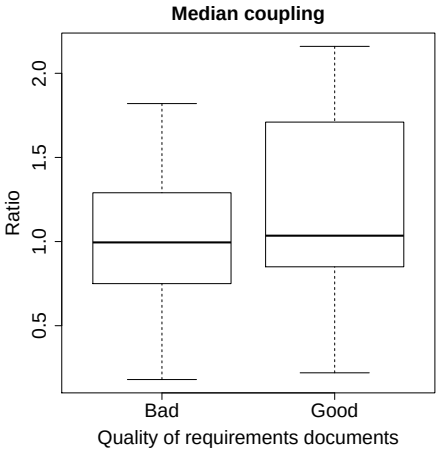
The results of the quantitative analysis (Table 16) support the conclusion drawn from the box plots (Figure 2, 3 and 4). There does not seem to be a statistically significant difference in software quality between projects with bad requirements documents and projects with good requirements documents.

There are some threats to validity which have the potential to reduce the credibility of the results. We discuss these threats in the remainder of this subsection. For the sake of readability we include further figures and tables in Appendix A.

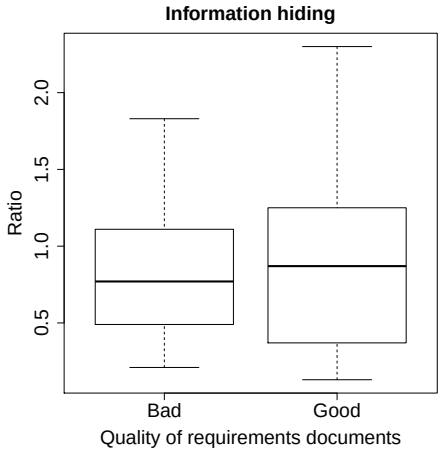
One threat to validity is the change of the setting in 2013. Before 2013 every team wrote the requirements document for their assigned subcomponent. From 2013 on dedicated requirements engineer wrote the requirements document for all subcomponents. Using again a score of 50 as a threshold to split the projects we ran the exactly same quantitative analysis for projects belonging to the *old setting* before 2013 and for projects belonging to the *new setting* from 2013 on. The box plots (Figure 11, 12 and 13 in Appendix



(a) Average coupling ratio (H.1)

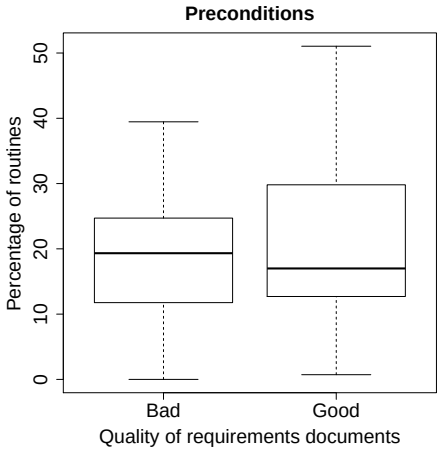


(b) Median coupling ratio (H.2)

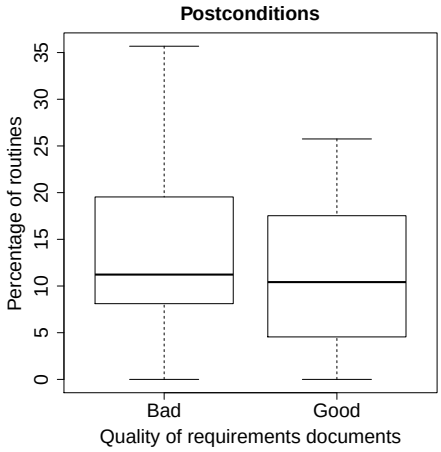


(c) Information hiding ratio (H.3)

Figure 2: Box plots of metrics related to ratios (H.1 to H.3) for research question RQ.1 using the average score 50 as a threshold to split the projects into projects with bad requirements documents and projects with good requirements documents.

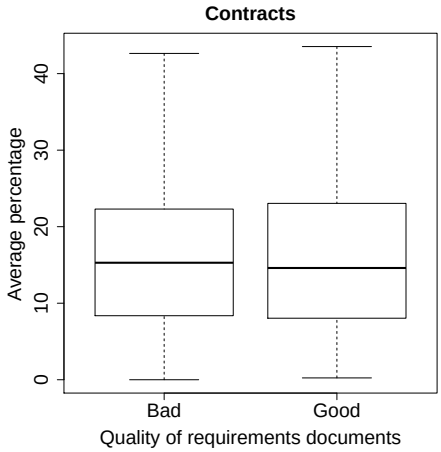


(a) Percentage of routines with preconditions (H.4)



(b) Percentage of routines with postconditions (H.5)

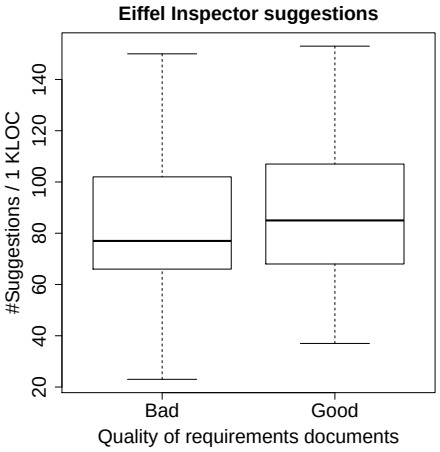
(c) Percentage of classes with class invariants (H.6)



(d) Average percentage of contracts (H.7)

Figure 3: Box plots of metrics related to contracts (H.4 to H.7) for research question RQ.1 using the average score 50 as a threshold to split the projects into projects with bad requirements documents and projects with good requirements documents.

(a) Number of Eiffel Inspector warnings per 1000 LOC (H.8)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.9)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.10)

Figure 4: Box plots of metrics related to Eiffel Inspector (H.8 to H.10) for research question RQ.1 using the average score 50 as a threshold to split the projects into projects with bad requirements documents and projects with good requirements documents.

Metric	Bad			Good			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.1)	34	1.841	1.162	34	2.12	1.249	502	0.354	-0.955	65.659	0.343
Median coupling (H.2)		1.143	0.957		1.434	1.201	488.5	0.275	-1.105	62.861	0.274
Information hiding (H.3)		0.991	0.762		1.434	2.388	590	0.888	-1.029	39.65	0.31
Preconditions (H.4)		19.147	11.517		22.606	15.997	544	0.681	-1.023	59.963	0.31
Postconditions (H.5)		13.845	10.026		14.478	15.009	619	0.619	-0.204	57.559	0.839
Invariants (H.6)		15.621	17.548		16.543	16.835	537.5	0.621	-0.221	65.887	0.826
Contracts (H.7)		16.204	11.098		17.876	12.722	560	0.831	-0.577	64.805	0.566
Warnings (H.8)		34.324	12.853		32.588	9.576	614.5	0.659	0.631	61.007	0.53
Suggestions† (H.9)		86.382	28.67		88.471	31.53	549.5	0.731	-0.286	65.412	0.776
Rule violations† (H.10)		60.353	18.058		60.529	17.891	570.5	0.932	-0.04	65.994	0.968

Table 16: Quantitative results for research question RQ.1 using the average score 50 as a threshold to split the projects into projects with bad requirements documents and projects with good requirements documents. A metric marked with a dagger (†) follows a normal distribution (Figure 47 to 56 in Appendix B.1.1 for the corresponding normality tests). An asterisk (∗) marks statistically significant results that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.1.1) visualizing the quality metrics in the old setting suggest no difference. Again the quantitative results (Table 20 in Appendix A.1.1) support this observation. Figure 14, 15 and 16 in Appendix A.1.2 showing the box plots indicate average number of Eiffel Inspector rule violations per 1000 LOC (H.10) might differ in the new setting. But the quantitative results (Table 21 in Appendix A.1.2) do not indicate a statistically significant difference. Thus the different settings do not influence the quantitative results our main analysis.

The choice of the threshold based on the histogram, mean and median is another threat to validity. Is a document with a score just a little above the threshold really a good document compared to a bad one with a score just a little below the threshold? We ran the previously discussed three analyses again and removed approximately 20 percent of the documents around the chosen threshold such that approximately 40 percent of the projects have bad requirements documents and approximately 40 percent of the projects have good requirements documents. For convenience we call this analysis the *40-20-40 analysis*. Doing so results in projects with requirements documents having a score less than 48 being projects with bad requirements documents and projects with requirements documents having a score of greater than 54 being projects with good requirements documents.

The first 40-20-40 analysis is again one using all projects. Now the box plots in Figure 17, 18 and 19 in Appendix A.1.3 indicate a difference in the average coupling ratio (H.1). However the quantitative results in Table 22 in Appendix A.1.3 do not show a statistically significant differences for any metric. The results do not at all show a statistically significant difference. The result of the t-test for the median coupling ratio (H.2) is statistically significant but misleading. The box plot of the median coupling ratio (Figure 17b) does not suggest a difference and the median coupling ratio (H.2) does not follow a normal distribution (Figure 78 in Appendix B.1.4).

We cannot report any statistically significant differences for the projects done in the old setting (2009 - 2012) because there is none given visualization of the metrics (Figure 20, 21 and 22 in Appendix A.1.4) and the quantitative results (Table 23 in Appendix A.1.4).

The visualization (Figure 23, 24 and 25 in Appendix A.1.5) of the metrics in the new setting (2013 and 2014) reveal striking differences. In particular the average coupling ratio (H.1) (Figure 23a), median coupling ratio (H.1) (Figure 23b), number of Eiffel Inspector suggestions per 1000 LOC (H.9) (Figure 25b) and the average number of Eiffel Inspector rule violations per 1000 LOC (H.10) (Figure 25c) are very different. The quantitative results (Table 24 in Appendix A.1.5) report only one statistically significant difference. It is related to the average number of Eiffel Inspector rule violations

per 1000 LOC (H.10) which follows a normal distribution (Figure 106 in Appendix B.1.6). It is important to mention the distribution since it is the t-test which indicates a statistically significant difference. The p-value resulting from the analysis of the average coupling ratio (H.1) is exactly the significance level. In fact the p-value rounded to 5 digits is 0.04993. Thus, there is a barely statistically significant difference in the average coupling ratio which follows a normal distribution (Figure 77 in Appendix B.1.6).

Another threat to validity is the influence of time zone differences. We classified projects into three classes according to the maximum time zone difference between teams of a group: Small, medium and large. A *small* time zone difference is one where the maximum time zone differences between teams of a group is less than or equal to three hours. A *medium* time zone difference is one where the maximum time zone differences between teams of a group is greater than three hours but less than or equal to five hours. A *large* time zone difference is one where the maximum time zone differences between teams of a group is more than five hours. Then we quantitatively analyzed the impact of the time zone classification on software quality using the *Kruskal-Wallis test* and *analysis of variance (ANOVA)* [53]. We ran the quantitative analysis for all projects (Figure 26, 27 and 28 and Table 25 in Appendix A.1.6), the projects between 2009 and 2012 ((Figure 29, 30 and 31 and Table 26 in Appendix A.1.7)) and the projects from 2013 and 2014 (Figure 32, 33 and 34 and Table 27 in Appendix A.1.8). The quantitative results show no statistically significant difference in software quality between the different time zone classifications.

The quantitative results of the quantitative analysis show evidence that the quality of requirements documents does not influence software quality.

4.4 Quantitative results for RQ.2

The second research question looks into the impact of programmer- and requirements engineer-written requirements documents on software quality.

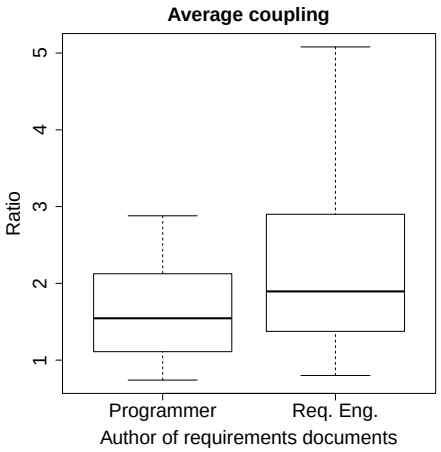
RQ.2 What is the impact of programmer-written and requirements engineer-written requirements documents on the average coupling ratio, the median coupling ratio, the information hiding ratio, the percentage of routines with preconditions, the percentage of routines with postconditions, the percentage of classes with invariants, the average percentage of contracts, the number of Eiffel Inspector warnings per 1000 LOC, the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC?

Between 2008 and 2012 every team had to write their own requirements document for their assigned subcomponent whereas teams consisting of only requirements engineers wrote the requirements documents for the projects in 2013 and 2014. Thus, we split the projects in projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents. There exist 48 projects with programmer-written requirements documents and 20 projects with requirements engineer-written requirements documents (Table 17). The box plots visualizing the metrics (Figure 5, 6 and 7) show a salient difference in the percentage of routines with preconditions (H.14) (Figure 6a), the percentage of routines with postconditions (H.14) (Figure 6b), the percentage of classes with class invariants (H.16) (Figure 6c) and average percentage of contracts (H.17) (Figure 6d) being specified. When programmer write the requirements themselves they seem to specify more contracts.

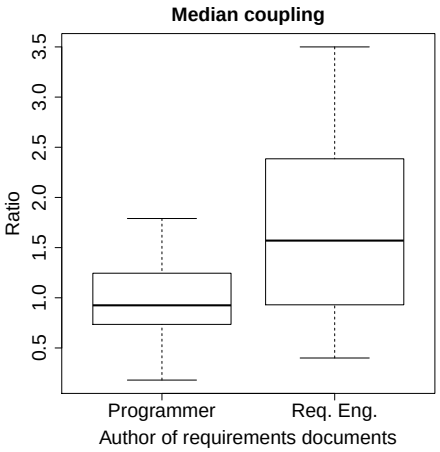
Indeed, the U-test agrees that these differences are statistically significant (Table 17). The U-test also reports a statistically significant difference in the median coupling ratio (H.12). The result suggests that writing the requirements and implementing them leads to a smaller median coupling ratio (H.12).

The very different topic in the 2014 edition of DOSE is a threat to validity. As before we ran the same quantitative analysis another two times. The first time we compare the software quality between projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents from the 2013 edition. In 2013 the topic was card & board games and thus comparable to the topics between 2009 and 2012. Then we compare the software quality between projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents from the 2014 edition. The topic in 2014 was a project-management tool which the students had to program using Eiffel for the *back end* and web technologies for the *front end*.

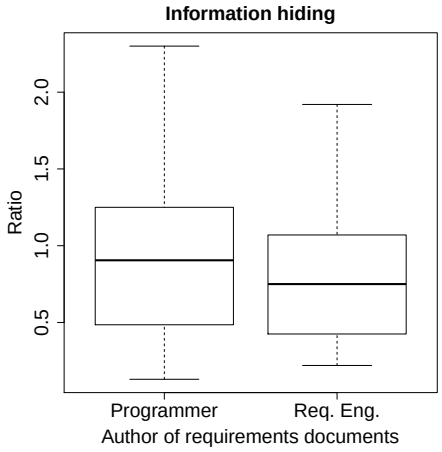
With regard to the 2013 edition the visual analysis of the metrics' box plots (Figure 35, 36 and 37 in Appendix A.2.1) seems to confirm the results of the main analysis related to contracts (H.14 to H.17) (Figure 36) but the median coupling ratio (H.12) (Figure 35b) seems not to differ. The U-test supports this interpretation regarding the percentage of routines with precondtions (H.14), the percentage of routines with postconditions (H.15) and the average percentage of contracts (H.17) being specified. In contrast to the main analysis there is no statistically significant difference in the percentage of classes with class invariants (H.16) and in the median coupling ratio (H.12). None of all metrics follows a normal distribution which makes the (statistically significant) results of the t-test difficult to interpret.



(a) Average coupling ratio (H.11)

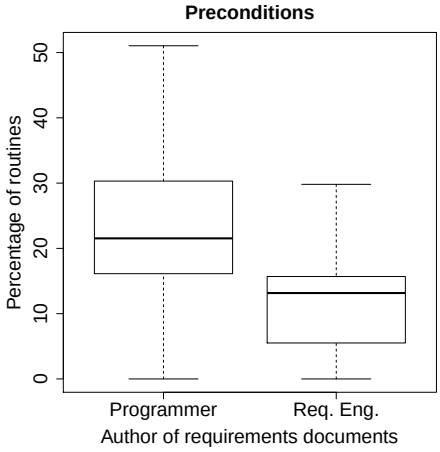


(b) Median coupling ratio (H.12)

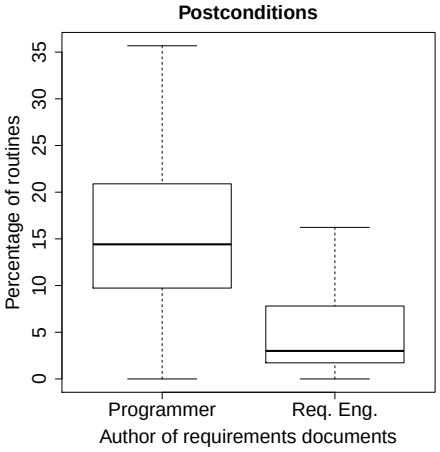


(c) Information hiding ratio (H.13)

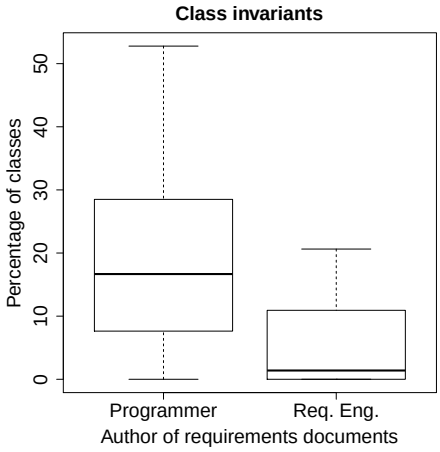
Figure 5: Box plots of metrics related to ratios (H.11 to H.13) for research question RQ.2.



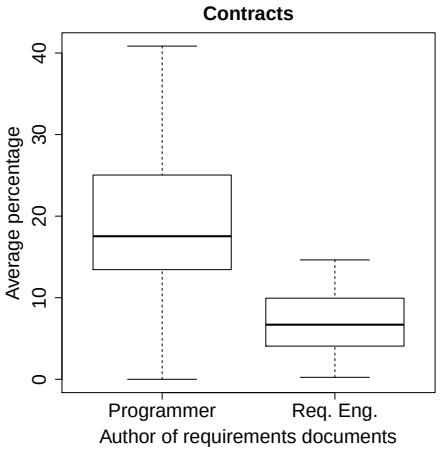
(a) Percentage of routines with preconditions (H.14)



(b) Percentage of routines with postconditions (H.15)

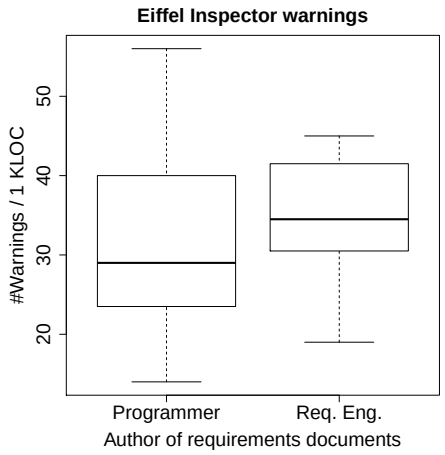


(c) Percentage of classes with class invariants (H.16)

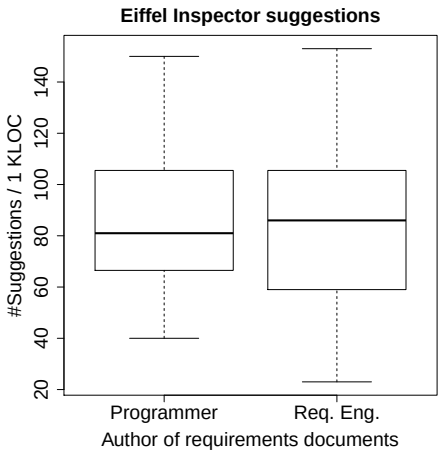


(d) Average percentage of contracts (H.17)

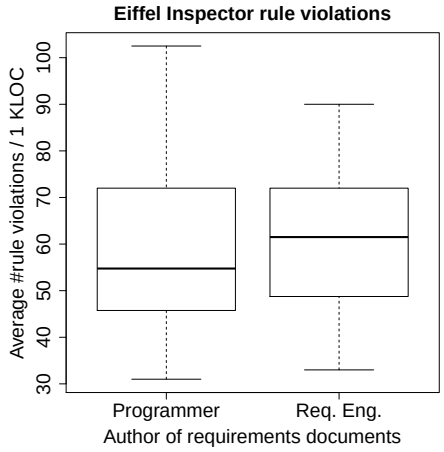
Figure 6: Box plots of metrics related to contracts (H.14 to H.17) for research question RQ.2.



(a) Number of Eiffel Inspector warnings per 1000 LOC (H.18)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.19)



(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.20)

Figure 7: Box plots of metrics related to Eiffel Inspector (H.18 to H.20) for research question RQ.2.

Metric	Programmer			Req. Eng.			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.11)	48	1.839	1.112	20	2.321	1.376	379.5	0.178	-1.39	29.854	0.175
Median coupling (H.12)		0.959	0.42		2.079	1.674	254	0.002*	-2.954	20.007	0.008*
Information hiding (H.13)		1.347	2.044		0.888	0.774	551	0.343	1.344	65.672	0.184
Preconditions (H.14)		23.542	11.908		14.48	16.546	750.5	0*	2.221	27.568	0.035*
Postconditions (H.15)		17.061	11.271		7.205	13.407	816	0*	2.89	30.76	0.007*
Invariants (H.16)		20.026	17.952		6.615	9.848	735	0.001*	3.944	60.868	0*
Contracts (H.17)		20.21	10.991		9.434	10.602	796	0*	3.778	36.839	0.001*
Warnings (H.18)		32.562	12.088		35.6	8.976	356.5	0.098	-1.142	47.582	0.259
Suggestions (H.19)		88.438	28.516		85	33.733	497.5	0.819	0.4	30.892	0.692
Rule violations (H.20)		60.5	18.593		60.3	16.344	456.5	0.757	0.044	40.279	0.965

Table 17: Quantitative results for research question RQ.2. A metric marked with a dagger (†) follows a normal distribution (Figure 107 to 116 in Appendix B.2.1 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

The second visual analysis (Figure 38, 39 and 40 in Appendix A.2.2) reveals an obvious difference in the average coupling ratio (H.11), median coupling ratio (H.12) and the four metrics related to contracts (H.14 to H.17). The U-test agrees (Table 29 in Appendix A.2.2) but also reports a statistically significant difference for the number of Eiffel Inspector suggestions per 1000 LOC (H.19). The only metric following a normal distribution is the median coupling ratio (H.12) and the t-test finds a statistically significant differences as well. It seems that the statistically significant difference in the median coupling ratio (H.12) and the percentage of classes with class invariants (H.16) of the main analysis including all projects is due to the very different topic in the 2014 edition of DOSE.

The quantitative results of the quantitative analysis show evidence that programmer-written requirements documents result in more preconditions and postconditions being specified than requirements engineer-written requirements documents. This complies with the finding that projects with programmer-written requirements have a higher average percentage of contracts. The conclusion regarding the percentage of classes with class invariants is difficult to make. There seems to be a trend that programmer-written requirements documents result in more class invariants being specified.

4.5 Quantitative results for RQ.3

The third research question looks into the impact of high- and mixed-context culture groups on software quality. First we had to classify every team as either having a low- or high-context culture. Copeland et al. [17] classified cultures as either low- or high-context cultures. Due to the unavailability of a copy of their book [17] we had to use a secondary source [4] to access their classification. This allowed us to classify the teams from Switzerland, Denmark and Australia as teams with low-context cultures. All other teams are teams with high-context cultures. The next step was to classify groups based on the cultures of the teams. We classified groups with both low- and high-context cultures as mixed culture groups. The classification of groups made us realize that not a single group with teams having only low-context cultures exists. This is why we had to formulate research RQ.3 using only high-context cultures and mixed-context cultures. The wording of research question RQ.3 is as follows.

RQ.3 What is the impact of high- and mixed-context culture groups on the average coupling ratio, the median coupling ratio, the information hiding ratio, the percentage of routines with preconditions, the percentage

of routines with postconditions, the percentage of classes with invariants, the average percentage of contracts, the number of Eiffel Inspector warnings per 1000 LOC, the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC?

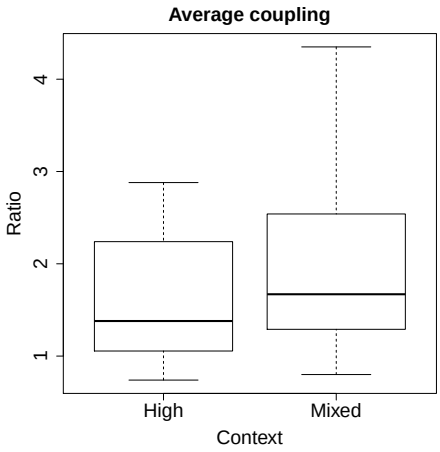
Splitting the data sample in two results in 31 projects with high-context culture groups and 37 projects with mixed-context culture groups (Table 18). The visualizations of the metrics using box plots (Figure 8, 9 and 10) suggests the absence of differences between projects with high- and mixed-context culture groups.

The visual inspection complies with the quantitative results of the U-test (Table 18) which report no statistically significant differences.

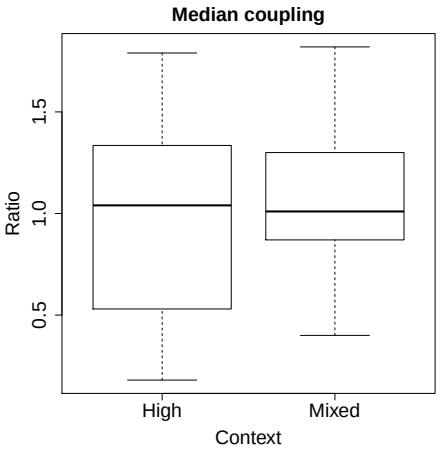
Another two times we run the same analysis again for each setting of the projects to reduce any threat to validity.

First we look into the projects done between 2009 and 2012. The box plots of the metrics (Figure 41, 42 and 43 in Appendix A.3.1) seem to indicate a difference in both the average coupling ratio (H.21) and mean coupling ratio (H.22) as well as the percentage of classes with class invariants (H.26). The quantitative results (Table 30) of the U-test support the observation of the difference in the average coupling ratio (H.21) but the U-test does not support the observation of the difference in the median coupling ratio (H.22) and the percentage of classes with class invariants (H.26). The t-test reports a statistically significant difference for both the average coupling ratio (H.21) and the average number of Eiffel Inspector rule violations per 1000 LOC (H.30) but these metrics do not follow a normal distribution which makes the results hard to interpret.

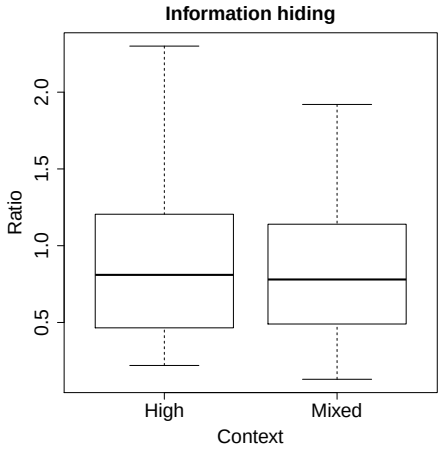
The remaining analysis researches the projects done in the new setting in 2013 and 2014. The metrics' visualization (Figure 44, 45 and 46 in Appendix A.3.2) show some striking differences. The most notable differences are related to the metrics average coupling ratio (H.21), median coupling ratio (H.22), the information hiding ratio (H.23) and the percentage of classe with class invariants (H.26). The results of the quantitative analysis (Table 31 in Appendix A.3.2) show that the U-test finds statistically significant differences in the average coupling ration (H.21), the median coupling ratio (H.22) and the percentage of classes with class invariants (H.26). This complies with the results of the t-test but only the average coupling ratio (H.21) and the median coupling ratio (H.22) follow a normal distribution. The result of the t-test with respect to the information hiding ratio (H.23) is exactly the significance level. The p-value rounded to five digits is 0.05035. The information hiding ratio (H.23) is not normally distributed but the normality tests



(a) Average coupling ratio (H.21)

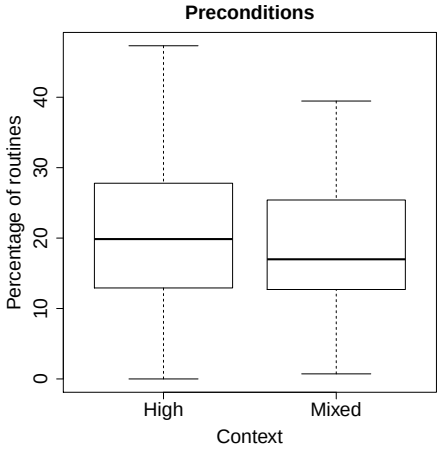


(b) Median coupling ratio (H.22)

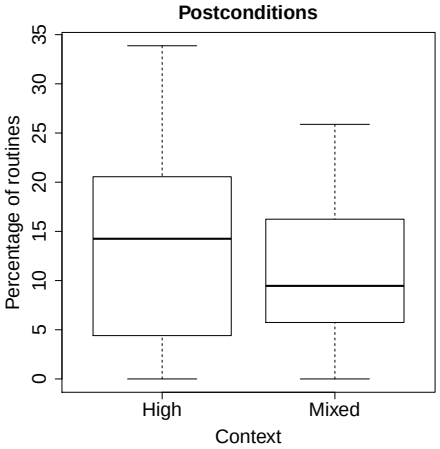


(c) Information hiding ratio (H.23)

Figure 8: Box plots of ratio metrics (H.21 to H.23) for research question RQ.3.

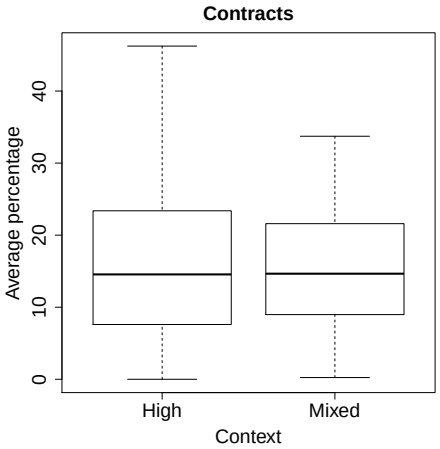


(a) Percentage of routines with preconditions (H.24)



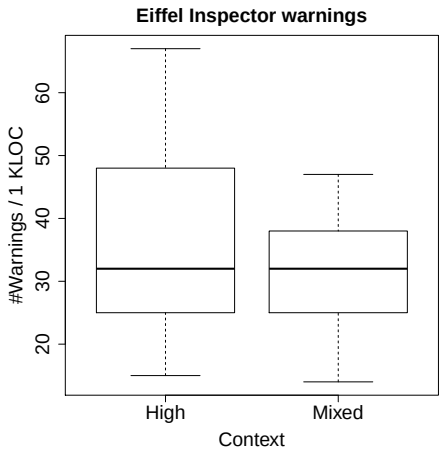
(b) Percentage of routines with postconditions (H.25)

(c) Percentage of classes with class invariants (H.26)

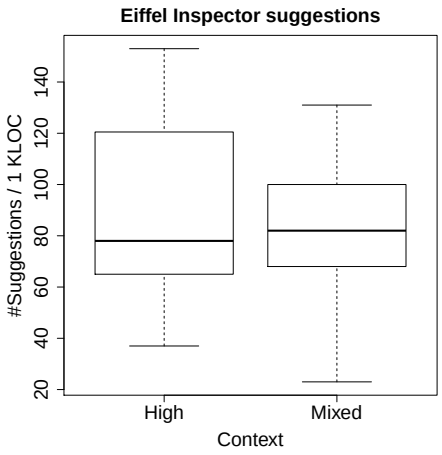


(d) Average percentage of contracts (H.27)

Figure 9: Box plots of metrics related to contracts (H.24 to H.27) for research question RQ.3.



(a) Number of Eiffel Inspector warnings per 1000 LOC (H.28)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.29)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.30)

Figure 10: Box plots of metrics related to Eiffel Inspector (H.28 to H.30) for research question RQ.3.

Metric	High			Mixed			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.21)	31	1.932	1.252	37	2.022	1.18	500.5	0.372	-0.303	62.446	0.763
Median coupling (H.22)		1.478	1.522		1.129	0.461	537.5	0.662	1.229	34.626	0.228
Information hiding (H.23)		1.137	1.209		1.276	2.15	593	0.815	-0.335	58.327	0.739
Preconditions (H.24)		22.097	15.774		19.854	12.334	620	0.571	0.644	56.297	0.522
Postconditions (H.25)		15.714	14.194		12.862	11.275	653.5	0.328	0.905	56.86	0.369
Invariants (H.26)		12.582	13.717		19.015	19.138	443.5	0.108	-1.61	64.557	0.112
Contracts (H.27)		16.797	11.775		17.244	12.122	565	0.922	-0.154	64.531	0.878
Warnings (H.28)		35.613	14.078		31.649	8.025	624.5	0.534	1.39	45.735	0.171
Suggestions (H.29)		91.742	36.422		83.811	23.061	611	0.649	1.049	48.937	0.299
Rule violations (H.30)		63.677	22.589		57.73	12.27	613.5	0.627	1.313	44.406	0.196

Table 18: Quantitative results for research question RQ.3. A metric marked with a dagger (†) follows a normal distribution (Figure 137 to 146 in Appendix B.3.1 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

(Figure 159 in Appendix B.3.3) show few outliers from normality. Since the t-test is robust against outliers it shows some evidence that the information hiding ratio (H.23) might be statistically significant different.

The quantitative results show evidence that the composition of groups with teams consisting of only high-context cultures or mixed-context cultures does not influence software quality. However the quantitative results regarding the average coupling ratio are not consistent but contradict each other. This shows the need for further analyses.

4.6 Correlations

Apart from the previous analyses we also researched correlations between the chosen metrics. To analyze the correlations between the metrics we use Kendall's τ rank correlation coefficient. The computation of the correlation coefficient did not reveal surprising correlations between metrics. Table 19 reports the significant correlations ($\tau \geq 0.4$ and $p < 0.01$ or $\tau \leq -0.4$ and $p < 0.01$) revealed by Kendall's τ rank correlation coefficient. Some of the found correlations are not unexpected. The average and median coupling ratio are closely related to each other. Other significant correlations relate to both the average percentage of contracts and the average number of Eiffel Inspector rule violations are defined using other metrics. This explains the correlations between the percentage of routines with preconditions and the average percentage of contracts, the percentage of routines with postconditions and the average percentage of contracts, the percentage of classes with class invariants and the average percentage of contracts, the number of Eiffel Inspector warnings per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC as well as the number of Eiffel Inspector suggestions per 1000 LOC and the average number of Eiffel Inspector rule violations per 1000 LOC.

The only really interesting but not surprising significant correlation is the one between the percentage of routines with preconditions and the percentage of routines with postconditions. Thus whenever a feature has preconditions, it is likely to have postconditions or vice versa.

Pair of metrics	τ
Average coupling ratio / Median coupling ratio	0.462
Preconditions (%) / Postconditions (%)	0.591
Preconditions (%) / Contracts (%)	0.675
Postconditions (%) / Contracts (%)	0.667
Invariants (%) / Contracts (%)	0.633
#Warnings / #Rule violations	0.411
#Suggestions / #Rule violations	0.828

Table 19: Significant correlations between pair of metrics using Kendall's τ correlation coefficient. All reported pairs of metrics have a correlation coefficient $\tau \geq 0.4$ or $\tau \leq -0.4$ with significance $p \ll 0.001$.

5 Related work

5.1 Empirical studies on software quality

Both Spinellis [57] and Bird et al. [8] looked into software quality of operating systems developed in a globally distributed environment.

FreeBSD [37] is an open-source operating system which Spinellis [57] used to research the impact of global development on productivity, quality and cooperation between developers. His findings include that global distribution does neither affect productivity nor quality (source code and faults) nor developer cooperation. Regarding latter it however seems the mentoring of new contributors is sometimes easier to establish between close by contributors.

Bird et al. [8] used faults found after the release of the operating system *Windows Vista* to learn more about the differences in quality between globally distributed and traditional local software development. The differences found are minor.

Another study [14] is concerned with the impact of process maturity and global distribution on software quality in terms of faults. Cataldo et al. [14] provided evidence for the impact. Increasing the maturity results in better software quality. However increasing distribution reduces the benefits of improving process maturity.

5.2 Empirical studies relying on DOSE

DOSE is a rich resource for empirical studies related to globally distributed software development. Our study used requirements documents and software products. Other empirical studies relying on DOSE looked into awareness and merge conflicts [23], usage of contracts [48] as well as effects of distribution and time zone differences [45].

With regard to shared code bases Estler et al. [23] examined awareness and merge conflicts. Their study shows that the absence of awareness, that is the knowledge of other programmers changing possibly relevant source code, decreases the programmer's performance more than merge conflicts. This insight is important because their study also shows that the absence of awareness occurs more frequently than merge conflicts.

Nordio et al. [48] describe the failure of the students in the 2007 edition of DOSE to create a working software product. Investigations revealed that the primary reason relates to specification issues. The organizers enforced the usage of contracts in the following edition and the quality of contracts correlated with success to deploy the created software product.

The data originating from the 2009 and 2010 editions of DOSE allowed

Nordio et al. [45] to look into the effects of distribution and time zone differences. Distribution, that is the number of locations per projects, does not influence the amount of time spent for communication. Though the study of projects with two and three locations showed a not statistically significant trend for higher communication in projects with two locations. The statistical analysis of the amount of time spent for communication showed no significant difference between groups with zero to three hours, five to seven hours and more than nine hours time zone difference. The data shows a trend that groups with zero to three hours time zone difference spend more time communicating with each other. The study also shows that students of a group being only zero to three hours apart reply e-mails faster than those students of a group being nine hours away from each other.

5.3 Teaching DOSE

The rise of globally distributed software development requires universities to adjust their efforts in teaching. Additional topics [32] related to globally distributed software development should complement the ones of traditional local software development [27]. Though teaching [18, 19] globally distributed software development practically is difficult. Nordio et al. describe the challenges [46, 47] the lecturers of DOSE faced when organizing the course. Such a challenge is to ensure that problems occurring within a team do not spread to the other teams in a group. This is a crucial aspect since teams in a group rely on each other. Another contribution by Nordio et al. [44] presents the efforts undertaken to coach the students for the to them unknown global setting of the course and project. The preparation takes place in the form of a contest where the students from a group have to successfully collaborate and communicate to solve an assignment in as little time as possible.

Similar courses as DOSE exist and lecturers of these courses reported their findings and the challenges faced as well [29, 30, 56, 12, 54, 11, 49, 28, 58].

6 Conclusion

The motivation for our study was the well-known impact [7, 35, 5] of requirements in traditional local software development and the lack of knowledge with regard to globally distributed software development.

We used 68 software development projects done in a globally distributed environment to carry out the empirical study. The goal of the study was to answer three questions. 1. Does the quality of requirements documents influence software quality? 2. Does the authorship of requirements documents influence software quality? 3. Do multiple cultures in groups influence software quality?

We assessed both the quality of requirements documents and software to deploy statistical tests as part of the quantitative analysis. The quantitative results show evidence that neither the quality of requirements nor the culture influence software quality. However, the quantitative analysis suggests that authorship does. Software emerging from programmer-written requirements have more contracts namely preconditions, postconditions and class invariants than software originating from requirements engineer-written requirements.

This raises new questions with regard to previous research. Both Lutz [35] and Basili et al. [5] identified misunderstanding of requirements as a primary source of software faults. Do programmers write less contracts due to misunderstanding the requirements written by requirements engineers? Future work includes studying both requirements and contracts to explain the difference in the amount of contracts being specified based on authorship of requirements.

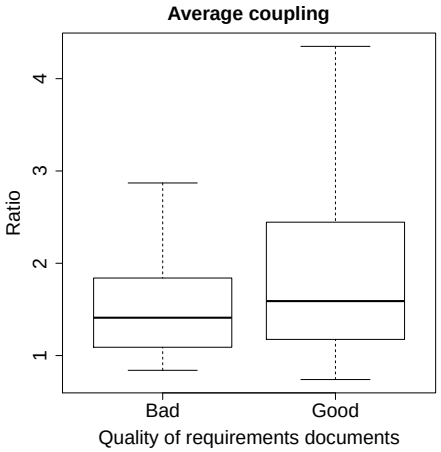
Our goal was to investigate the impact of requirements in globally distributed software development. The results of the empirical study are very interesting and raise new questions. This shows the need for further research to better understand the impact of requirements in globally distributed software development.

A Additional quantitative results

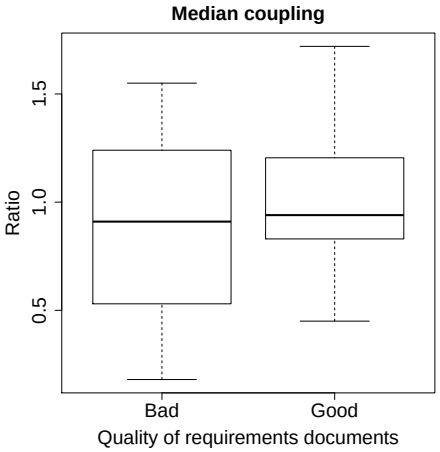
A.1 Research question RQ.1

A.1.1 Main analysis (2009 - 2012)

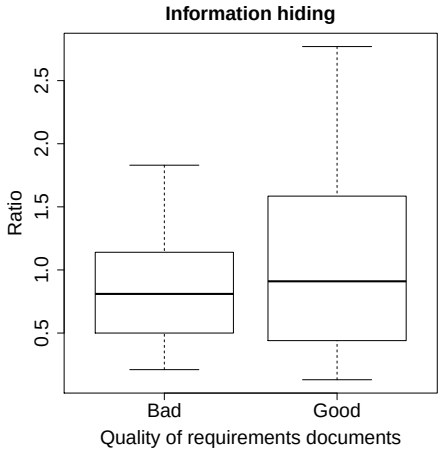
Figure 11 to 13 show the box plots for metrics of the projects done between 2009 and 2012 which the main analysis uses for research question RQ.1. Table 20 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.1)

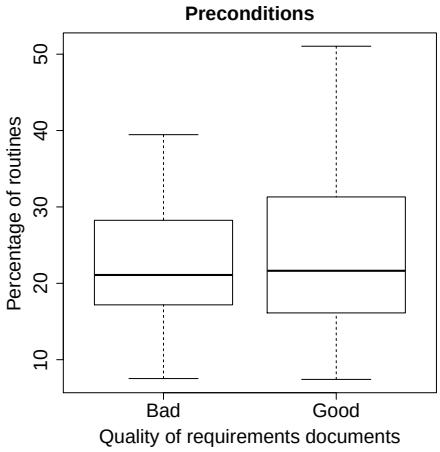


(b) Median coupling ratio (H.2)

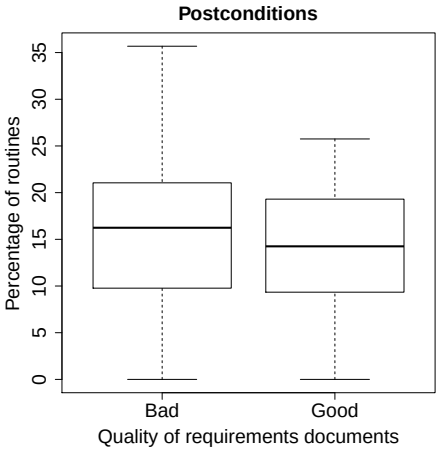


(c) Information hiding ratio (H.3)

Figure 11: Box plots of metrics related to ratios (H.1 to H.3) for research question RQ.1 using the average score 50 as a threshold to split the projects from 2009 to 2012 into projects with bad requirements documents and projects with good requirements documents.

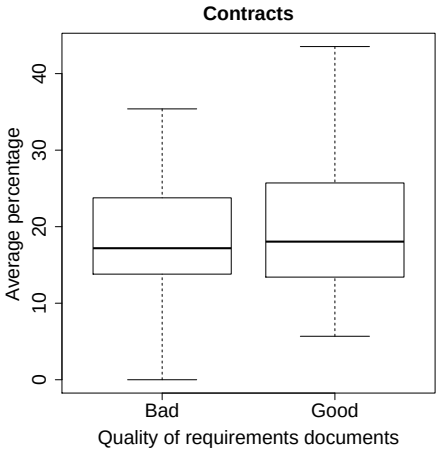


(a) Percentage of routines with preconditions (H.4)



(b) Percentage of routines with postconditions (H.5)

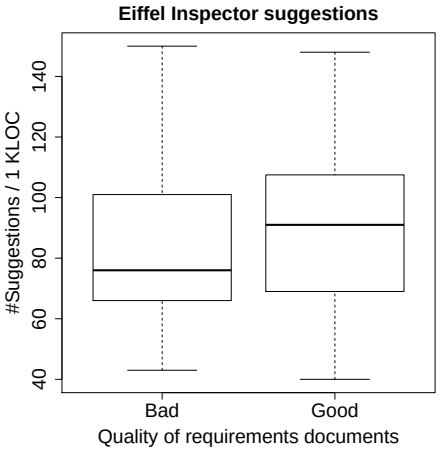
(c) Percentage of classes with class invariants (H.6)



(d) Average percentage of contracts (H.7)

Figure 12: Box plots of metrics related to contracts (H.4 to H.7) for research question RQ.1 using the average score 50 as a threshold to split the projects from 2009 to 2012 into projects with bad requirements documents and projects with good requirements documents.

(a) Number of Eiffel Inspector warnings per 1000 LOC (H.8)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.9)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.10)

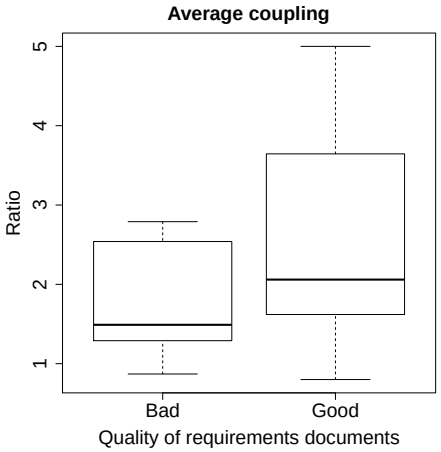
Figure 13: Box plots of metrics related to Eiffel Inspector (H.8 to H.10) for research question RQ.1 using the average score 50 as a threshold to split the projects from 2009 to 2012 into projects with bad requirements documents and projects with good requirements documents.

Metric	Bad			Good			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.1)	25	1.753	1.117	23	1.932	1.124	261	0.591	-0.552	45.618	0.584
Median coupling† (H.2)		0.895	0.388		1.029	0.452	255.5	0.516	-1.098	43.568	0.278
Information hiding (H.3)		0.978	0.67		1.749	2.848	280	0.885	-1.267	24.238	0.217
Preconditions† (H.4)		22.427	11.014		24.753	12.949	272	0.759	-0.668	43.407	0.508
Postconditions (H.5)		17.282	9.307		16.82	13.296	319.5	0.516	0.138	39.034	0.891
Invariants (H.6)		18.791	18.148		21.37	18.044	256.5	0.529	-0.493	45.711	0.624
Contracts (H.7)		19.5	10.453		20.981	11.734	276	0.822	-0.46	44.236	0.648
Warnings (H.8)		32.6	13.714		32.522	10.344	279	0.869	0.022	44.355	0.982
Suggestions (H.9)		84.56	27.286		92.652	29.822	231	0.248	-0.978	44.656	0.333
Rule violations (H.10)		58.58	19.027		62.587	18.299	238	0.312	-0.744	45.902	0.461

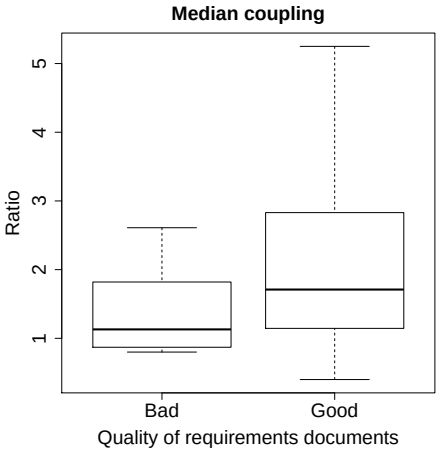
Table 20: Quantitative results for research question RQ.1 using the average score 50 as a threshold to split the projects from 2009 to 2012 into projects with bad requirements documents and projects with good requirements documents. A metric marked with a dagger (†) follows a normal distribution (Figures 57 to 66 in Appendix B.1.2 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.1.2 Main analysis (2013 - 2014)

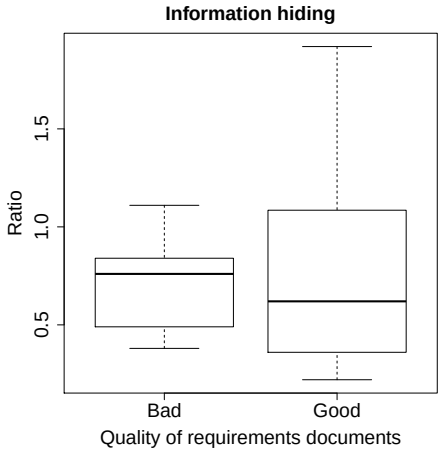
Figure 14 to 13 show the box plots for metrics of the projects done in 2013 and 2014 which the main analysis uses for research question RQ.1. Table 21 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.1)

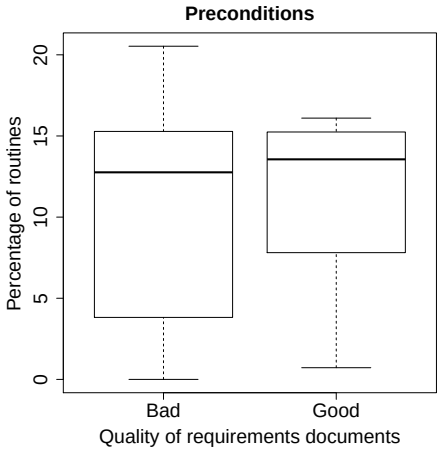


(b) Median coupling ratio (H.2)

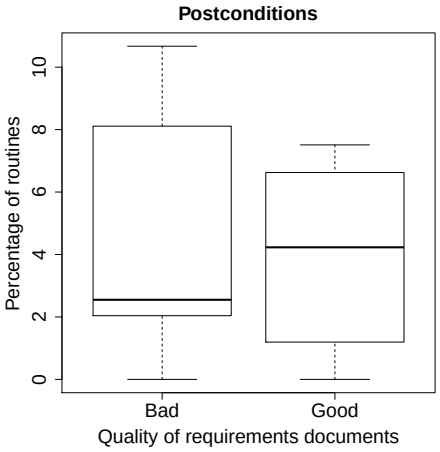


(c) Information hiding ratio (H.3)

Figure 14: Box plots of metrics related to ratios (H.1 to H.3) for research question RQ.1 using the average score 50 as a threshold to split the projects done in 2013 and 2014 into projects with bad requirements documents and projects with good requirements documents.

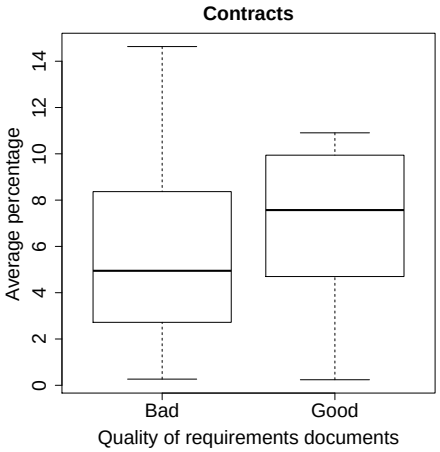


(a) Percentage of routines with preconditions (H.4)



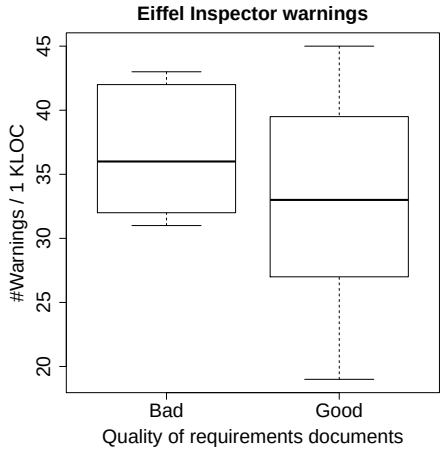
(b) Percentage of routines with postconditions (H.5)

(c) Percentage of classes with class invariants (H.6)

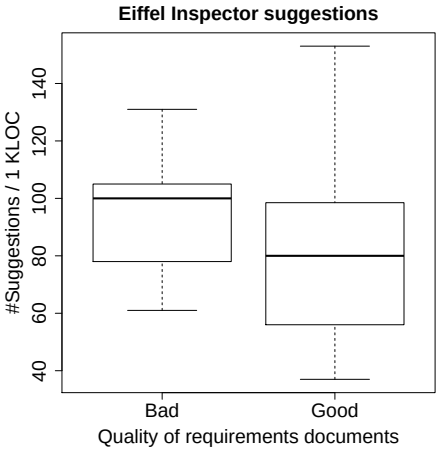


(d) Average percentage of contracts (H.7)

Figure 15: Box plots of metrics related to contracts (H.4 to H.7) for research question RQ.1 using the average score 50 as a threshold to split the projects done in 2013 and 2014 into projects with bad requirements documents and projects with good requirements documents.



(a) Number of Eiffel Inspector warnings per 1000 LOC (H.8)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.9)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.10)

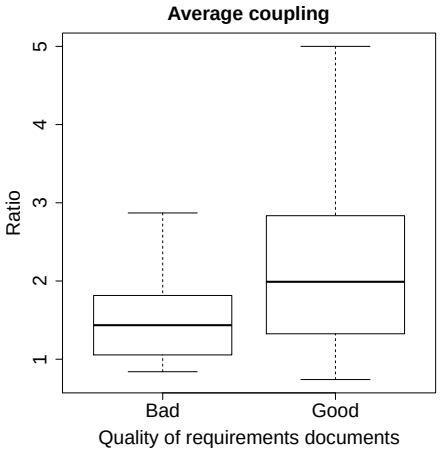
Figure 16: Box plots of metrics related to Eiffel Inspector (H.8 to H.10) for research question RQ.1 using the average score 50 as a threshold to split the projects done in 2013 and 2014 into projects with bad requirements documents and projects with good requirements documents.

Metric	Bad			Good			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling† (H.1)	9	2.084	1.317	11	2.515	1.454	40	0.503	-0.693	17.772	0.497
Median coupling (H.2)		1.832	1.612		2.281	1.773	41	0.552	-0.592	17.757	0.561
Information hiding (H.3)		1.028	1.023		0.774	0.517	54.5	0.732	0.678	11.296	0.512
Preconditions (H.4)		10.037	7.495		18.116	21.045	41	0.552	-1.185	12.951	0.257
Postconditions (H.5)		4.299	3.911		9.582	17.761	45	0.761	-0.959	11.171	0.358
Invariants (H.6)		6.816	12.785		6.452	7.309	40	0.465	0.076	12.155	0.941
Contracts (H.7)		7.05	7.213		11.383	12.754	37	0.37	-0.955	16.244	0.353
Warnings (H.8)		39.111	9.062		32.727	8.199	68	0.171	1.636	16.414	0.121
Suggestions† (H.9)		91.444	33.433		79.727	34.638	61	0.412	0.767	17.453	0.453
Rule violations† (H.10)		65.278	14.896		56.227	17.015	68.5	0.159	1.268	17.889	0.221

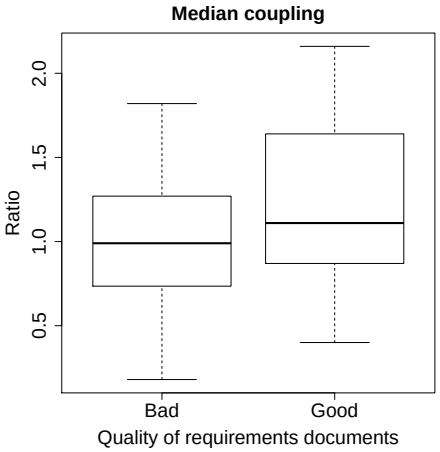
Table 21: Quantitative results for research question RQ.1 using the average score 50 as a threshold to split the projects done in 2013 and 2014 into projects with bad requirements documents and projects with good requirements documents. A metric marked with a dagger (†) follows a normal distribution (Figures 67 to 76 in Appendix B.1.3 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.1.3 40-20-40 analysis (2009 - 2014)

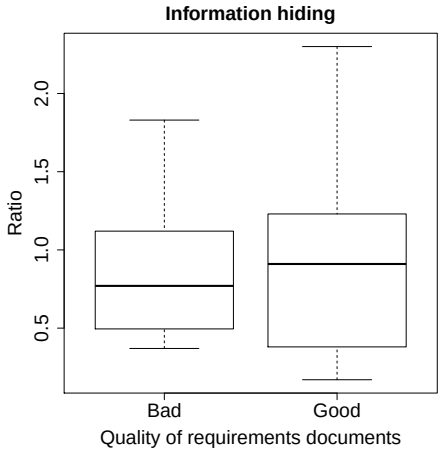
Figure 17 to 19 show the box plots for metrics of all projects which the 40-20-40 analysis uses for research question RQ.1. Table 22 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.1)

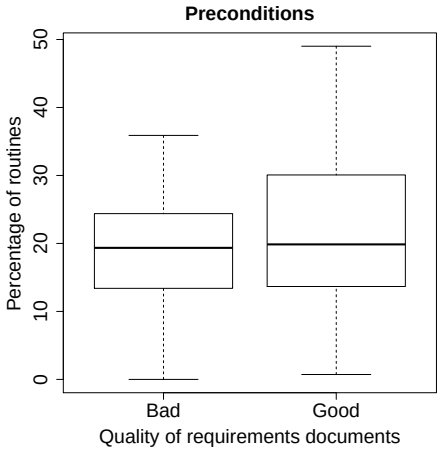


(b) Median coupling ratio (H.2)

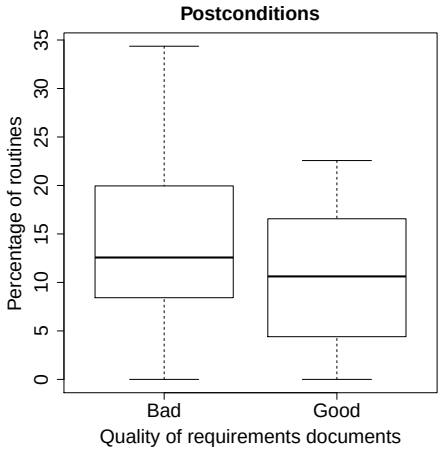


(c) Information hiding ratio (H.3)

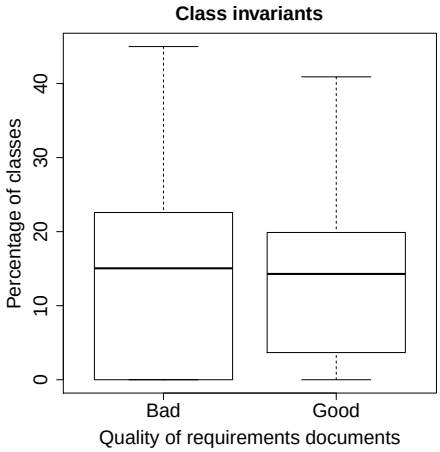
Figure 17: Box plots of metrics related to ratios (H.1 to H.3) for research question RQ.1 using the average score 50 as a threshold to split the projects from into projects with bad requirements documents and projects with good requirements documents.



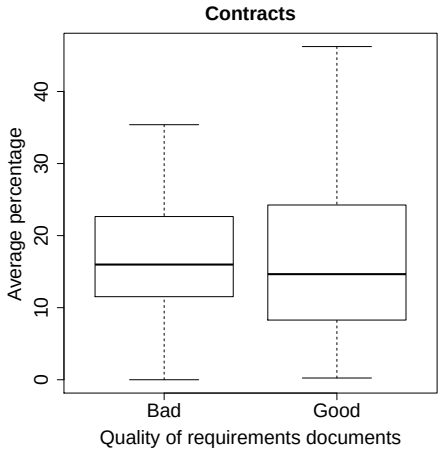
(a) Percentage of routines with preconditions (H.4)



(b) Percentage of routines with postconditions (H.5)



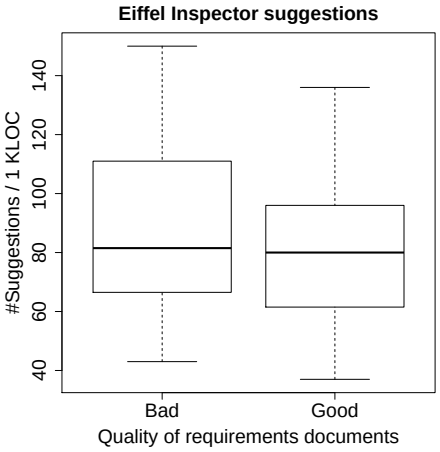
(c) Percentage of classes with class invariants (H.6)



(d) Average percentage of contracts (H.7)

Figure 18: Box plots of metrics related to contracts (H.4 to H.7) for research question RQ.1 using the average score 50 as a threshold to split the projects from into projects with bad requirements documents and projects with good requirements documents.

(a) Number of Eiffel Inspector warnings per 1000 LOC (H.8)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.9)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.10)

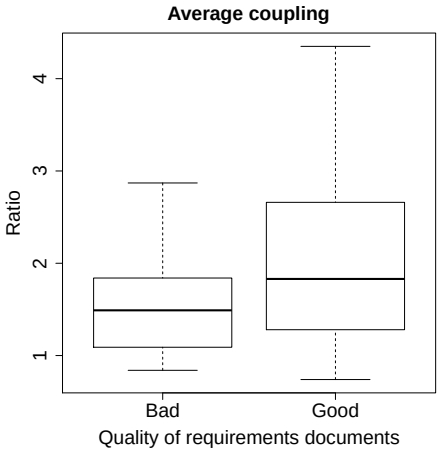
Figure 19: Box plots of metrics related to Eiffel Inspector (H.8 to H.10) for research question RQ.1 using the average score 50 as a threshold to split the projects from into projects with bad requirements documents and projects with good requirements documents.

Metric	Bad			Good			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.1)	28	1.726	1.075	27	2.267	1.332	284	0.115	-1.653	49.937	0.105
Median coupling (H.2)		0.968	0.413		1.53	1.3	285	0.119	-2.145	31.03	0.04*
Information hiding (H.3)		0.951	0.632		1.521	2.641	387	0.886	-1.091	28.864	0.284
Preconditions (H.4)		19.813	10.461		23.356	16.283	354	0.695	-0.956	44.089	0.344
Postconditions (H.5)		14.38	9.22		14.274	15.092	428	0.405	0.031	42.749	0.975
Invariants (H.6)		16.641	17.035		16.043	17.448	387	0.885	0.129	52.804	0.898
Contracts (H.7)		16.945	9.903		17.891	12.559	387	0.887	-0.31	49.418	0.758
Warnings† (H.8)		33.286	12.918		30.667	8.134	415.5	0.533	0.903	45.736	0.371
Suggestions† (H.9)		89.929	28.115		80.667	28.396	439.5	0.304	1.215	52.883	0.23
Rule violations† (H.10)		61.607	19.006		55.667	15.402	437	0.325	1.276	51.496	0.208

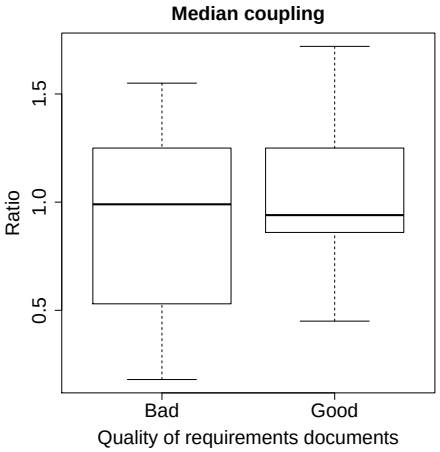
Table 22: Quantitative results of the 40-20-40 analysis for research question RQ.1 using all projects. A metric marked with a dagger (†) follows a normal distribution (Figures 77 to 86 in Appendix B.1.4 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.1.4 40-20-40 analysis (2009 - 2012)

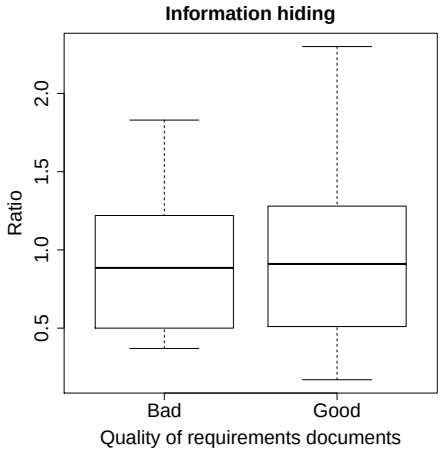
Figure 20 to 22 show the box plots for metrics of the projects done between 2009 and 2012 which the 40-20-40 analysis uses for research question RQ.1. Table 23 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.1)

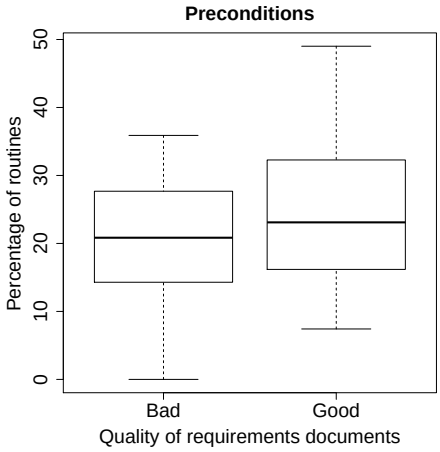


(b) Median coupling ratio (H.2)

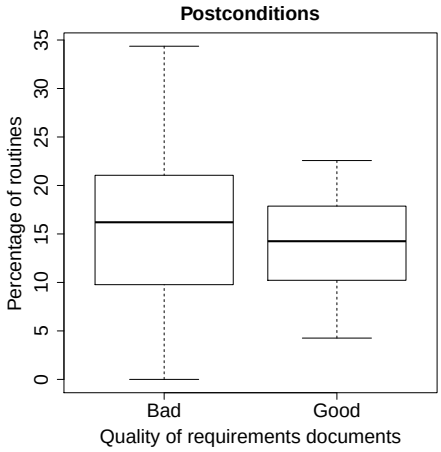


(c) Information hiding ratio (H.3)

Figure 20: Box plots of metrics related to ratios (H.1 to H.3) for research question RQ.1 using the average score 50 as a threshold to split the projects from 2009 to 2012 into projects with bad requirements documents and projects with good requirements documents.

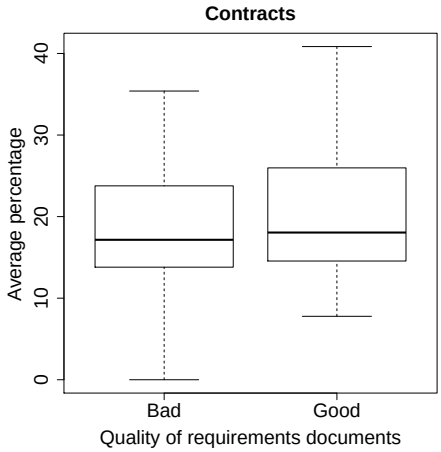


(a) Percentage of routines with preconditions (H.4)



(b) Percentage of routines with postconditions (H.5)

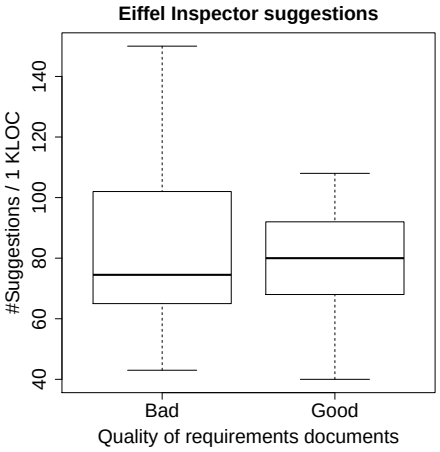
(c) Percentage of classes with class invariants (H.6)



(d) Average percentage of contracts (H.7)

Figure 21: Box plots of metrics related to contracts (H.4 to H.7) for research question RQ.1 using the average score 50 as a threshold to split the projects from 2009 to 2012 into projects with bad requirements documents and projects with good requirements documents.

(a) Number of Eiffel Inspector warnings per 1000 LOC (H.8)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.9)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.10)

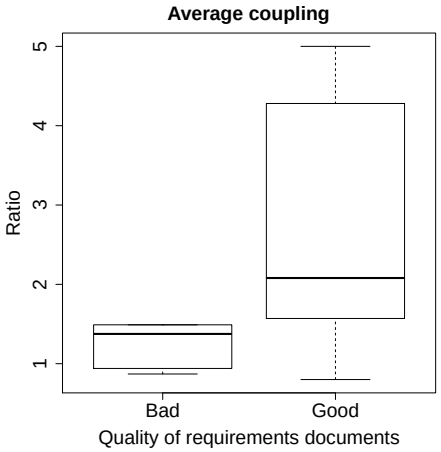
Figure 22: Box plots of metrics related to Eiffel Inspector (H.8 to H.10) for research question RQ.1 using the average score 50 as a threshold to split the projects from 2009 to 2012 into projects with bad requirements documents and projects with good requirements documents.

Metric	Bad			Good			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.1)	22	1.807	1.169	17	2.075	1.224	161.5	0.479	-0.693	33.738	0.493
Median coupling† (H.2)		0.911	0.4		1.072	0.429	160	0.453	-1.196	33.286	0.24
Information hiding (H.3)		1.029	0.689		1.951	3.261	185.5	0.977	-1.146	17.108	0.267
Preconditions† (H.4)		21.425	10.975		25.661	12.281	153	0.347	-1.118	32.432	0.272
Postconditions (H.5)		16.7	8.909		16.741	12.537	202	0.681	-0.012	27.709	0.991
Invariants (H.6)		18.391	17.488		22.082	18.968	162.5	0.496	-0.623	33.056	0.537
Contracts† (H.7)		18.839	9.817		21.495	10.881	167	0.585	-0.789	32.624	0.436
Warnings (H.8)		32.5	14.375		30.176	8.669	198	0.766	0.625	35.189	0.536
Suggestions (H.9)		85.455	28.766		81.235	23.653	189	0.966	0.502	36.824	0.618
Rule violations (H.10)		58.977	20.175		55.706	14.395	187	1	0.591	36.813	0.558

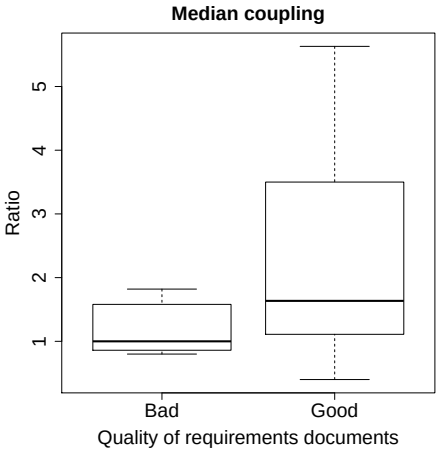
Table 23: Quantitative results of the 40-20-40 analysis for research question RQ.1 using projects from 2009 to 2012. A metric marked with a dagger (†) follows a normal distribution (Figures 87 to 96 in Appendix B.1.5 for the corresponding normality tests). An asterisk ($*$) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.1.5 40-20-40 analysis (2013 - 2014)

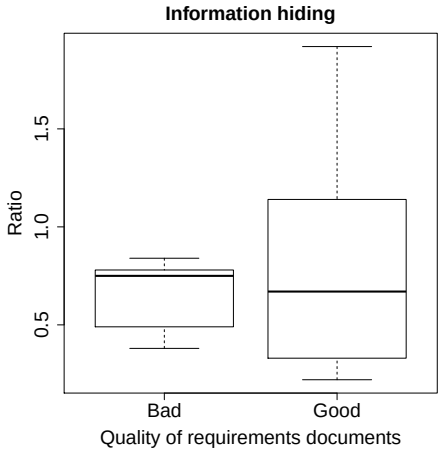
Figure 23 to 25 show the box plots for metrics of the projects done in 2013 and 2014 which the 40-20-40 analysis uses for research question RQ.1. Table 24 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.1)

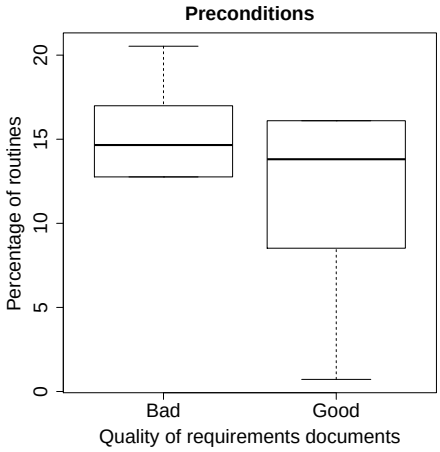


(b) Median coupling ratio (H.2)

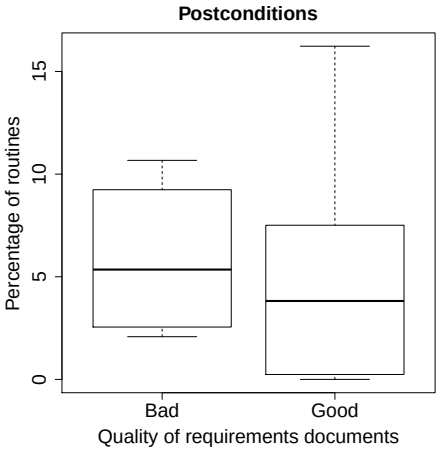


(c) Information hiding ratio (H.3)

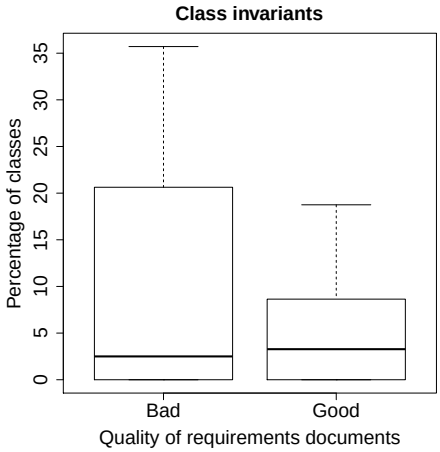
Figure 23: Box plots of metrics related to ratios (H.1 to H.3) for research question RQ.1 using the average score 50 as a threshold to split the projects done in 2013 and 2014 into projects with bad requirements documents and projects with good requirements documents.



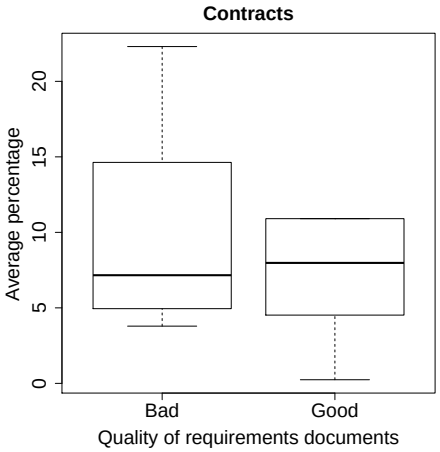
(a) Percentage of routines with preconditions (H.4)



(b) Percentage of routines with postconditions (H.5)



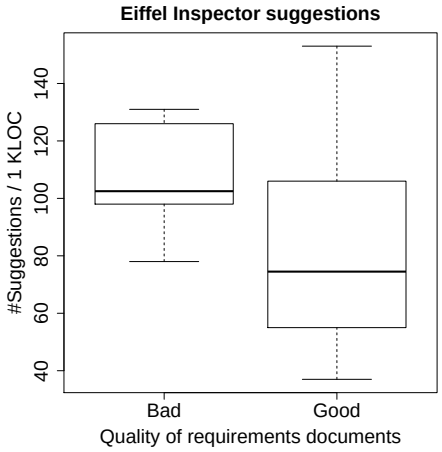
(c) Percentage of classes with class invariants (H.6)



(d) Average percentage of contracts (H.7)

Figure 24: Box plots of metrics related to contracts (H.4 to H.7) for research question RQ.1 using the average score 50 as a threshold to split the projects done in 2013 and 2014 into projects with bad requirements documents and projects with good requirements documents.

(a) Number of Eiffel Inspector warnings per 1000 LOC (H.8)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.9)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.10)

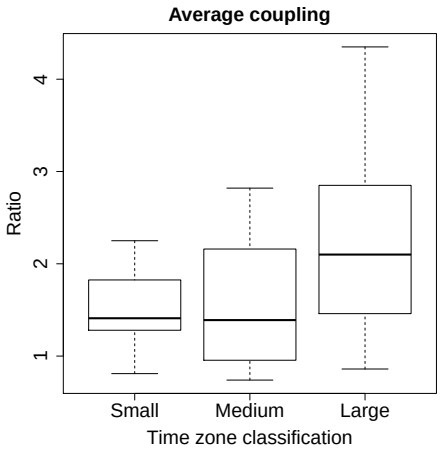
Figure 25: Box plots of metrics related to Eiffel Inspector (H.8 to H.10) for research question RQ.1 using the average score 50 as a threshold to split the projects done in 2013 and 2014 into projects with bad requirements documents and projects with good requirements documents.

Metric	Bad			Good			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling† (H.1)	6	1.432	0.602	10	2.593	1.508	14	0.093	-2.165	12.788	0.05
Median coupling† (H.2)		1.177	0.428		2.309	1.866	20	0.313	-1.84	10.5	0.094
Information hiding† (H.3)		0.665	0.185		0.789	0.542	27.5	0.828	-0.662	12.005	0.52
Preconditions (H.4)		13.902	5.627		19.438	21.697	33	0.792	-0.765	10.885	0.46
Postconditions (H.5)		5.873	3.887		10.08	18.641	35	0.625	-0.689	10.255	0.506
Invariants (H.6)		10.223	14.824		5.776	7.334	32	0.865	0.686	6.498	0.516
Contracts (H.7)		9.999	7.155		11.765	13.378	31	0.958	-0.343	13.929	0.736
Warnings† (H.8)		36.167	4.535		31.5	7.502	42	0.211	1.551	13.972	0.143
Suggestions† (H.9)		106.333	19.541		79.7	36.512	44	0.147	1.898	13.93	0.079
Rule violations† (H.10)		71.25	9.949		55.6	17.801	48	0.057	2.255	13.987	0.041*

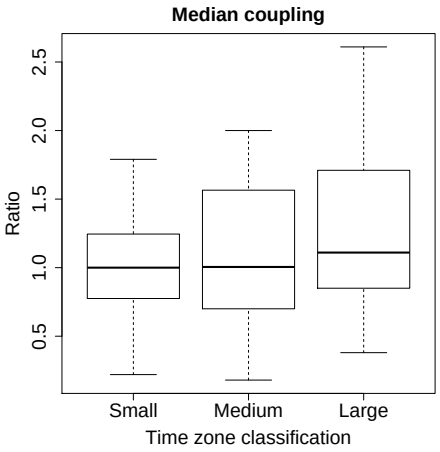
Table 24: Quantitative results of the 40-20-40 analysis for research question RQ.1 using projects done in 2013 and 2014. A metric marked with a dagger (†) follows a normal distribution (Figures 97 to 106 in Appendix B.1.6 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.1.6 Time zone classification analysis (2009 - 2014)

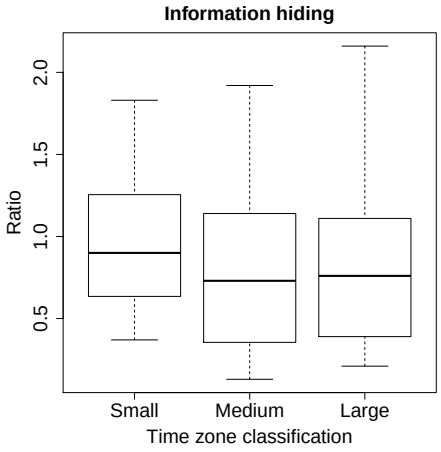
Figure 26 to 28 show the box plots visualizing metrics of all projects which the time zone classification analysis uses. Table 25 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio

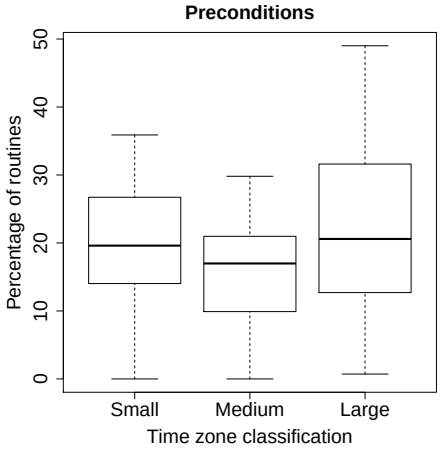


(b) Median coupling ratio

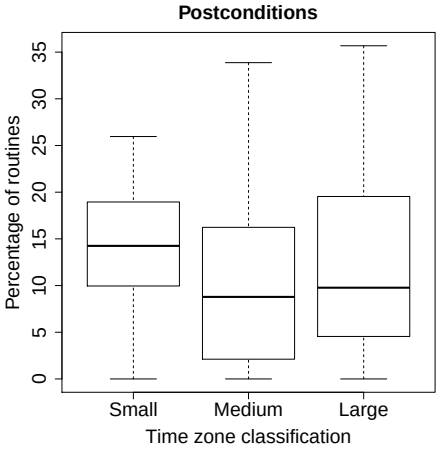


(c) Information hiding ratio

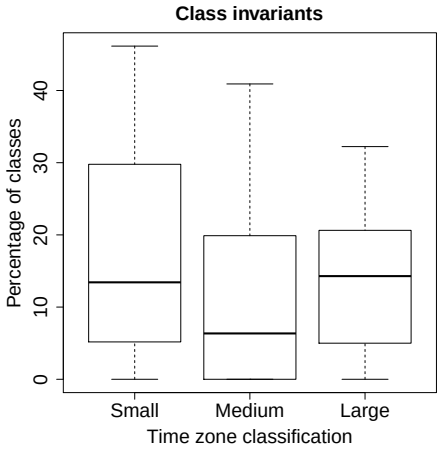
Figure 26: Box plots of metrics related to ratios for time zone difference analysis.



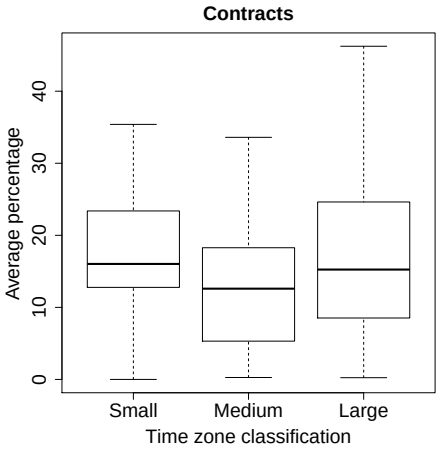
(a) Percentage of routines with preconditions



(b) Percentage of routines with postconditions

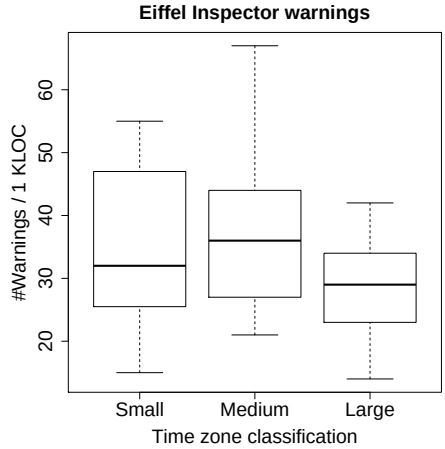


(c) Percentage of classes with class invariants

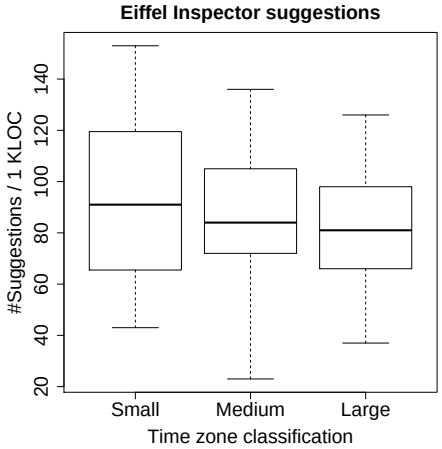


(d) Average percentage of contracts

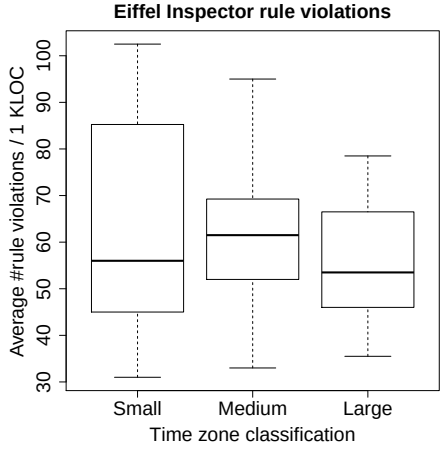
Figure 27: Box plots of metrics related to contracts for time zone difference analysis.



(a) Number of Eiffel Inspector warnings per 1000 LOC



(b) Number of Eiffel Inspector suggestions per 1000 LOC



(c) Average number of Eiffel Inspector rule violations per 1000 LOC

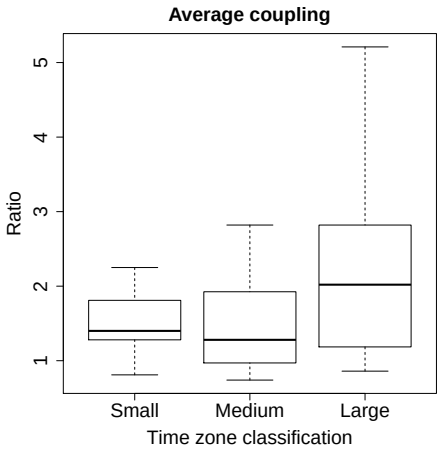
Figure 28: Box plots of metrics related to Eiffel Inspector for time zone difference analysis.

Metric	Small			Medium			Large			ANOVA			Kruskal-Wallis test		
	#	Mean	SD	#	Mean	SD	#	Mean	SD	df	F	p	χ^2	df	p
Average coupling	23	1.586	0.548	20	1.971	1.543	25	2.351	1.266	2	2.525	0.088	4.678	2	0.096
Median coupling		0.983	0.377		1.448	1.54		1.442	1.076	2	1.39	0.256	1.396	2	0.498
Information hiding		1.617	2.625		0.884	0.807		1.102	1.298	2	0.991	0.377	2.969	2	0.227
Preconditions		22.08	12.489		16.42	9.693		23.335	17.353	2	1.519	0.227	2.348	2	0.309
Postconditions		15.635	10.269		10.198	9.63		15.977	16.111	2	1.407	0.252	3.951	2	0.139
Invariants		20.682	22.789		11.883	13.247		15.21	12.766	2	1.494	0.232	1.692	2	0.429
Contracts		19.465	12.07		12.834	8.904		18.174	13.261	2	1.897	0.158	3.453	2	0.178
Warnings		34.435	12.225		37.1	12.806		29.64	7.836	2	2.689	0.076	3.713	2	0.156
Suggestions		94.826	35.183		87.7	28.559		80.4	24.816	2	1.41	0.251	1.367	2	0.505
Rule violations		64.63	22.151		62.4	16.403		55.02	13.233	2	1.964	0.149	2.42	2	0.298

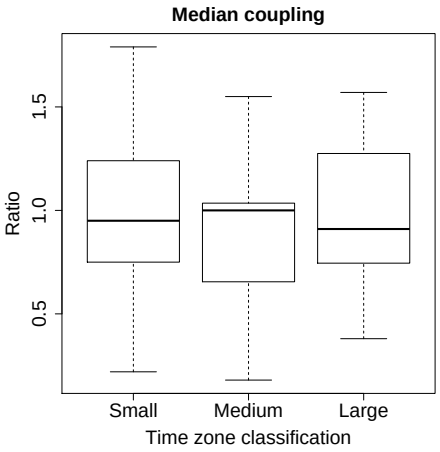
Table 25: Quantitative results for time zone difference analysis using all projects. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.1.7 Time zone classification analysis (2009 - 2012)

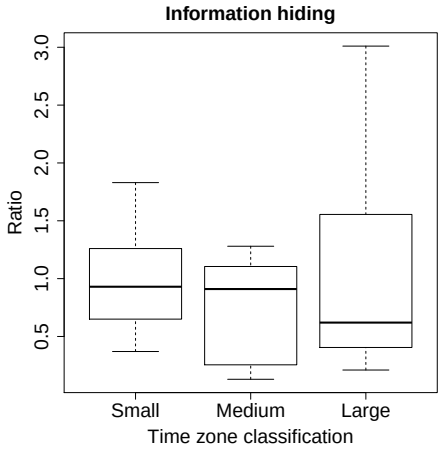
Figure 29 to 31 show the box plots visualizing metrics of projects from 2009 to 2012 which the time zone classification analysis uses. Table 26 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio

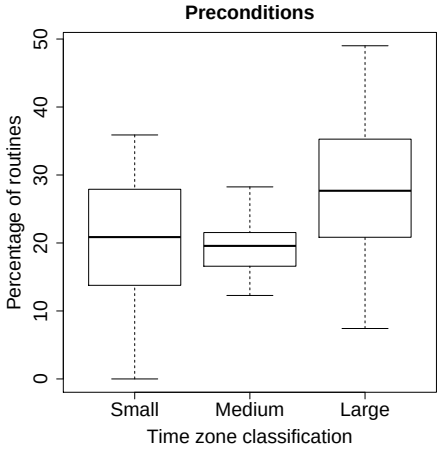


(b) Median coupling ratio

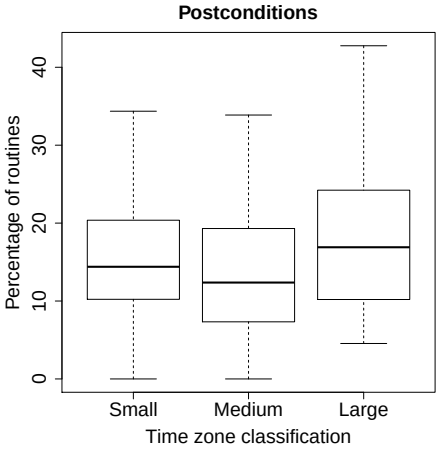


(c) Information hiding ratio

Figure 29: Box plots of metrics related to ratios for time zone difference analysis.

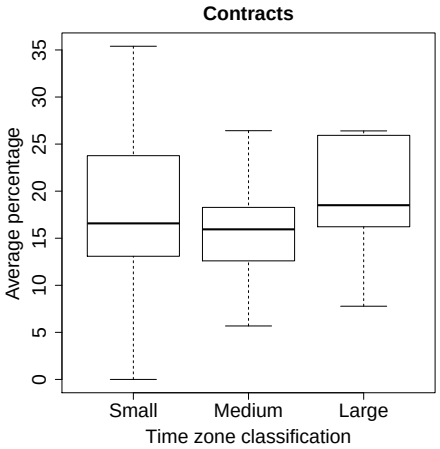


(a) Percentage of routines with preconditions



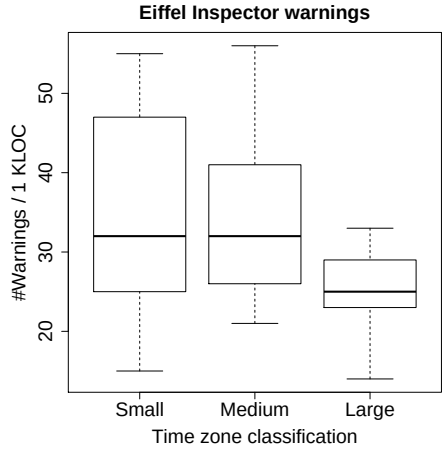
(b) Percentage of routines with postconditions

(c) Percentage of classes with class invariants

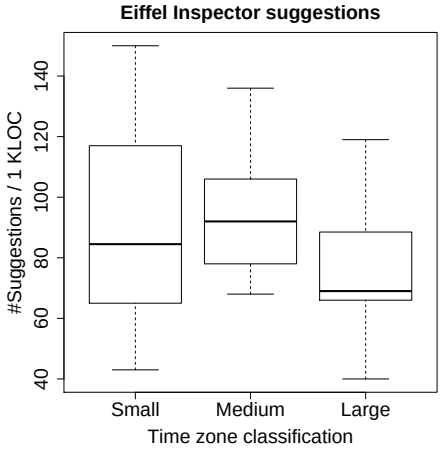


(d) Average percentage of contracts

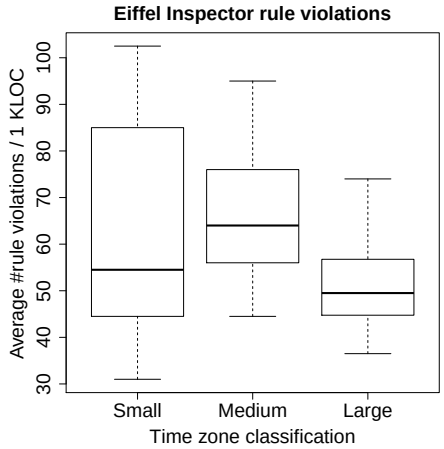
Figure 30: Box plots of metrics related to contracts for time zone difference analysis.



(a) Number of Eiffel Inspector warnings per 1000 LOC



(b) Number of Eiffel Inspector suggestions per 1000 LOC



(c) Average number of Eiffel Inspector rule violations per 1000 LOC

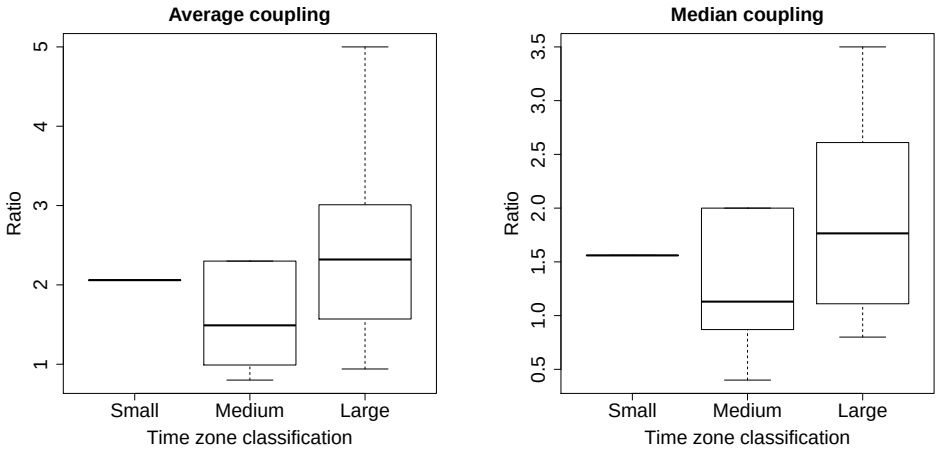
Figure 31: Box plots of metrics related to Eiffel Inspector for time zone difference analysis.

Metric	Small			Medium			Large			ANOVA			Kruskal-Wallis test		
	#	Mean	SD	#	Mean	SD	#	Mean	SD	df	F	p	χ^2	df	p
Average coupling	22	1.565	0.551	11	1.856	1.576	15	2.228	1.28	2	1.633	0.207	2.173	2	0.337
Median coupling		0.956	0.363		0.905	0.483		1.003	0.474	2	0.167	0.846	0.136	2	0.934
Information hiding		1.668	2.675		0.71	0.455		1.345	1.629	2	0.799	0.456	2.816	2	0.245
Preconditions		22.429	12.668		20.177	8.238		27.641	12.572	2	1.451	0.245	2.98	2	0.225
Postconditions		16.335	9.933		14.429	10.378		20.055	13.627	2	0.87	0.426	1.048	2	0.592
Invariants		21.622	22.864		15.111	13.532		21.291	11.93	2	0.526	0.595	1.394	2	0.498
Contracts		20.128	11.917		16.572	7.901		22.996	11.367	2	1.089	0.345	2.735	2	0.255
Warnings		34.773	12.402		36	14.711		26.8	7.262	2	2.699	0.078	4.764	2	0.092
Suggestions		92.182	33.591		96.636	22.787		76.933	21.208	2	1.94	0.156	3.687	2	0.158
Rule violations		63.477	21.954		66.318	16.801		51.867	10.751	2	2.604	0.085	4.337	2	0.114

Table 26: Quantitative results for time zone difference analysis using projects from 2009 to 2012. # stands for *number of projects* and *SD* stands for *standard deviation*.

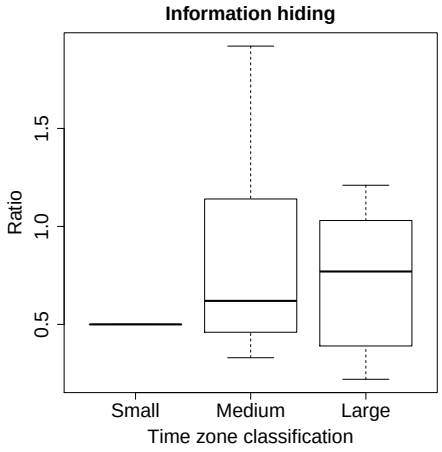
A.1.8 Time zone classification analysis (2013 - 2014)

Figure 32 to 34 show the box plots visualizing metrics of projects done in 2013 and 2014 which the time zone classification analysis uses. Table 27 shows the results of the corresponding quantitative analysis.



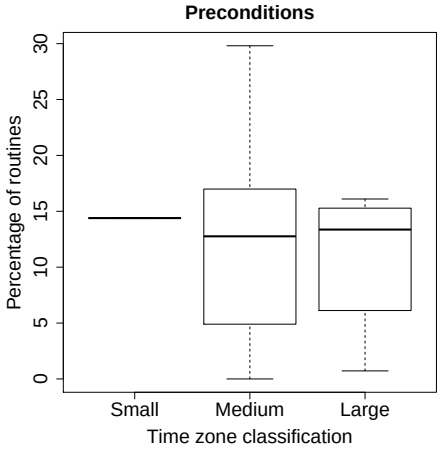
(a) Average coupling ratio

(b) Median coupling ratio

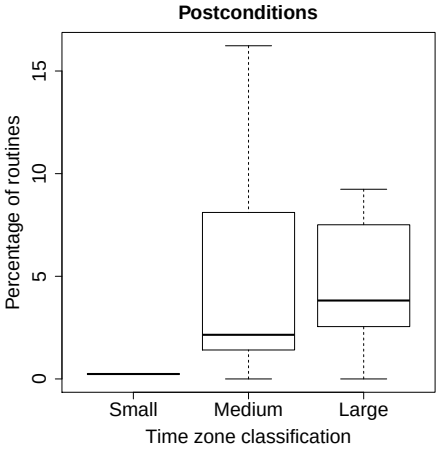


(c) Information hiding ratio

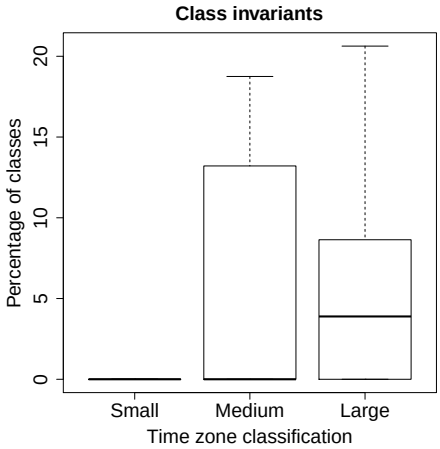
Figure 32: Box plots of metrics related to ratios for time zone difference analysis.



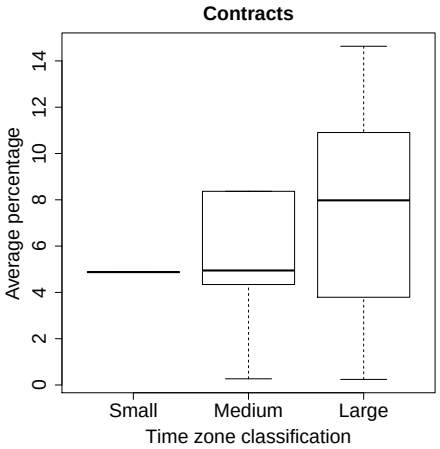
(a) Percentage of routines with preconditions



(b) Percentage of routines with postconditions

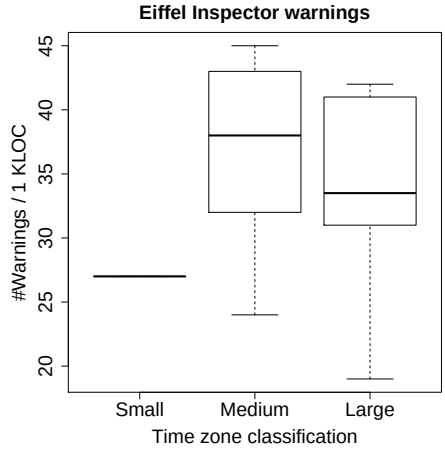


(c) Percentage of classes with class invariants

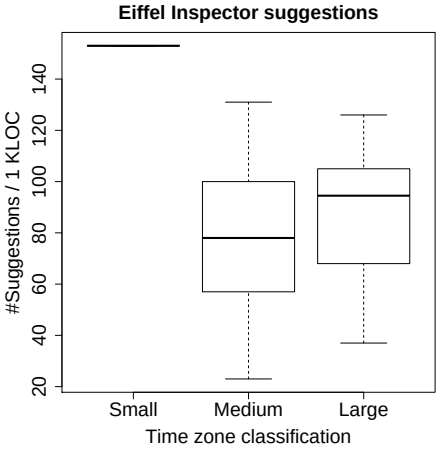


(d) Average percentage of contracts

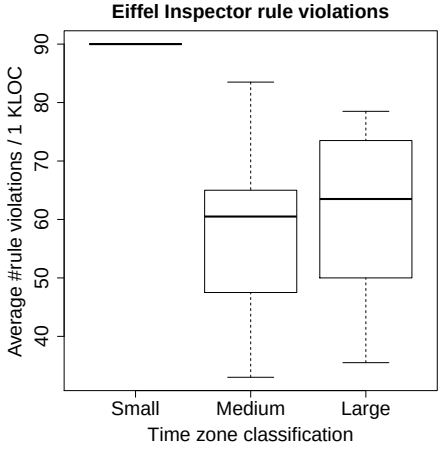
Figure 33: Box plots of metrics related to contracts for time zone difference analysis.



(a) Number of Eiffel Inspector warnings per 1000 LOC



(b) Number of Eiffel Inspector suggestions per 1000 LOC



(c) Average number of Eiffel Inspector rule violations per 1000 LOC

Figure 34: Box plots of metrics related to Eiffel Inspector for time zone difference analysis.

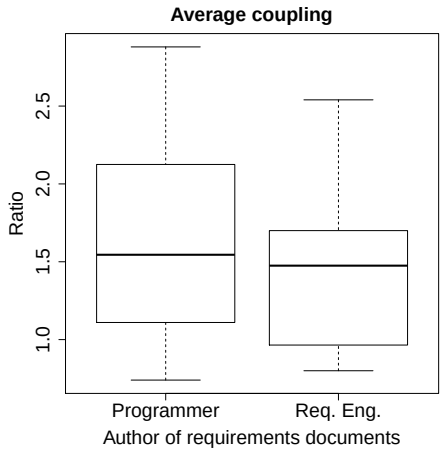
Metric	Small			Medium			Large			ANOVA			Kruskal-Wallis test		
	#	Mean	SD	#	Mean	SD	#	Mean	SD	df	F	p	χ^2	df	p
Average coupling	1	2.06	—	9	2.111	1.583	10	2.536	1.29	2	0.225	0.801	1.235	2	0.539
Median coupling		1.56	—		2.113	2.107		2.1	1.392	2	0.046	0.955	0.468	2	0.791
Information hiding		0.5	—		1.097	1.094		0.739	0.339	2	0.613	0.553	0.236	2	0.889
Preconditions		14.39	—		11.828	9.756		16.877	21.907	2	0.202	0.819	0.389	2	0.823
Postconditions		0.24	—		5.028	5.564		9.86	18.275	2	0.422	0.662	1.622	2	0.444
Invariants		0	—		7.938	12.492		6.087	7.688	2	0.297	0.747	0.936	2	0.626
Contracts		4.877	—		8.264	8.219		10.941	13.079	2	0.228	0.798	0.457	2	0.796
Warnings		27	—		38.444	10.737		33.9	6.935	2	1.102	0.355	2.304	2	0.316
Suggestions		153	—		76.778	32.31		85.6	29.875	2	2.716	0.095	3.203	2	0.202
Rule violations		90	—		57.611	15.467		59.75	15.676	2	1.958	0.172	2.903	2	0.234

Table 27: Quantitative results for time zone difference analysis using projects done in 2013 and 2014. # stands for *number of projects* and *SD* stands for *standard deviation*.

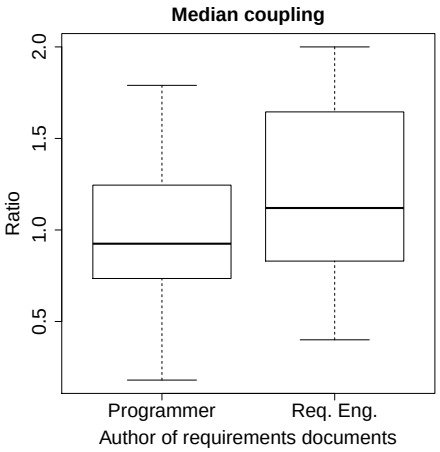
A.2 Research question RQ.2

A.2.1 Main analysis (2009 - 2012 vs. 2013)

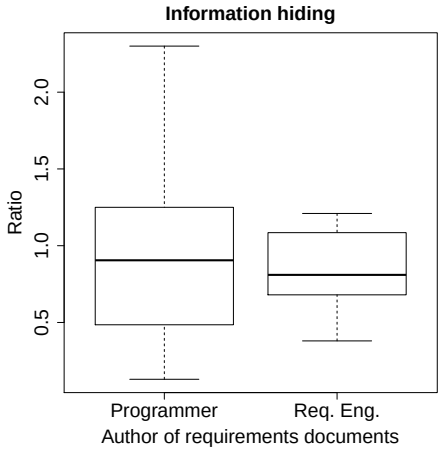
Figure 35 to 37 show the box plots for metrics comparing the projects done between 2009 and 2012 and the projects done in 2013 which the main analysis uses for research question RQ.2. Table 28 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.11)

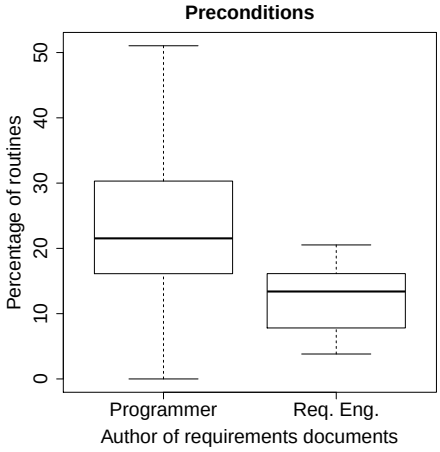


(b) Median coupling ratio (H.12)

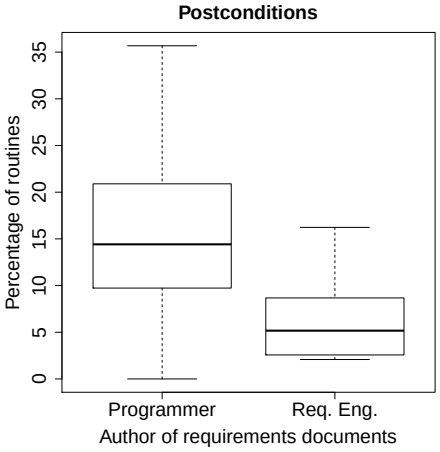


(c) Information hiding ratio (H.13)

Figure 35: Box plots of metrics related to ratios (H.11 to H.13) for research question RQ.2.

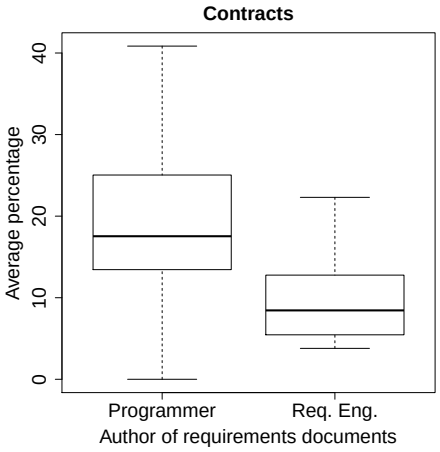


(a) Percentage of routines with preconditions (H.14)



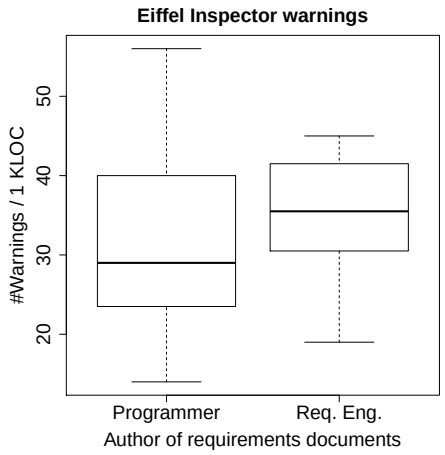
(b) Percentage of routines with postconditions (H.15)

(c) Percentage of classes with class invariants (H.16)

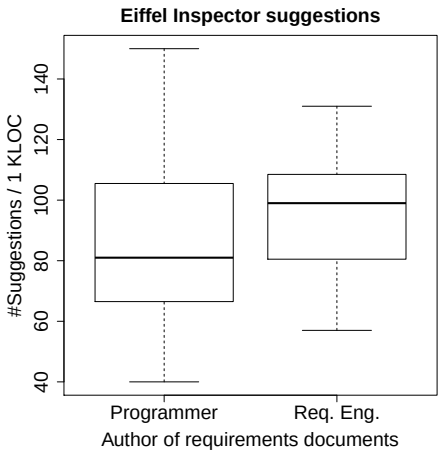


(d) Average percentage of contracts (H.17)

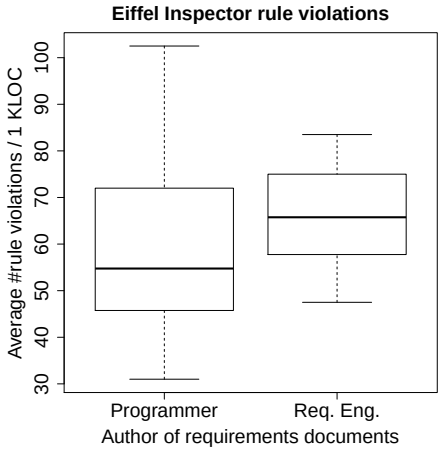
Figure 36: Box plots of metrics related to contracts (H.14 to H.17) for research question RQ.2.



(a) Number of Eiffel Inspector warnings per 1000 LOC (H.18)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.19)



(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.20)

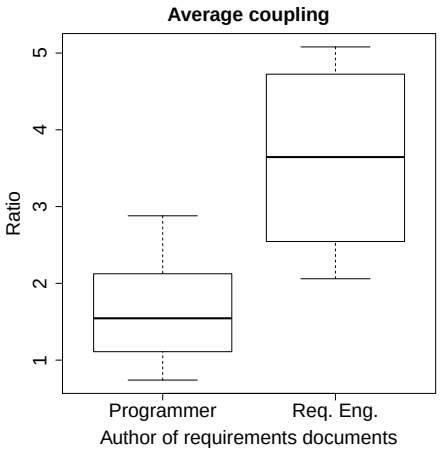
Figure 37: Box plots of metrics related to Eiffel Inspector (H.18 to H.20) for research question RQ.2.

Metric	Programmer			Req. Eng.			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.11)	48	1.839	1.112	12	1.454	0.523	337	0.37	1.746	38.405	0.089
Median coupling† (H.12)		0.959	0.42		1.171	0.507	227	0.263	-1.337	15.002	0.201
Information hiding (H.13)		1.347	2.044		0.896	0.404	293	0.934	1.424	56.9	0.16
Preconditions† (H.14)		23.542	11.908		13.375	7.172	450	0.002*	3.778	28.248	0.001*
Postconditions (H.15)		17.061	11.271		6.308	4.279	491	0*	5.264	48.266	0*
Invariants (H.16)		20.026	17.952		10.794	10.921	378	0.098	2.262	27.9	0.032*
Contracts (H.17)		20.21	10.991		10.159	6.268	467	0.001*	4.177	30.232	0*
Warnings (H.18)		32.562	12.088		34.583	7.798	225.5	0.251	-0.71	25.991	0.484
Suggestions (H.19)		88.438	28.516		97	21.084	218	0.199	-1.165	22.271	0.256
Rule violations (H.20)		60.5	18.593		65.792	11.329	207	0.137	-1.251	27.848	0.221

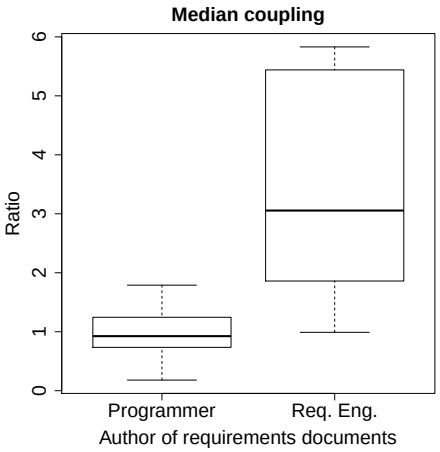
Table 28: Quantitative results for research question RQ.2 comparing projects from 2009 to 2012 with projects from 2013. A metric marked with a dagger (†) follows a normal distribution (Figures 107 to 116 in Appendix B.2.1 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.2.2 Main analysis (2009 - 2012 vs. 2014)

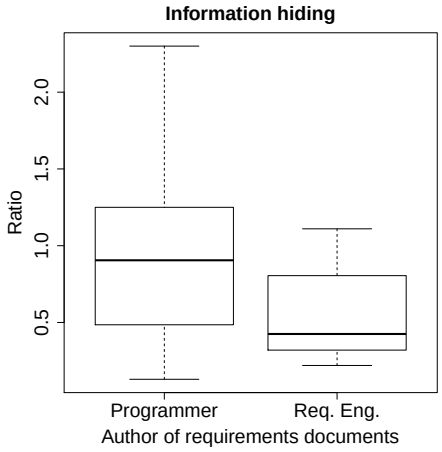
Figure 38 to 40 show the box plots for metrics comparing the projects done between 2009 and 2012 and the projects done in 2014 which the main analysis uses for research question RQ.2. Table 29 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.11)

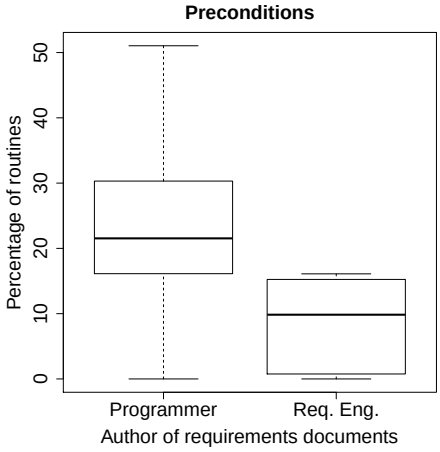


(b) Median coupling ratio (H.12)

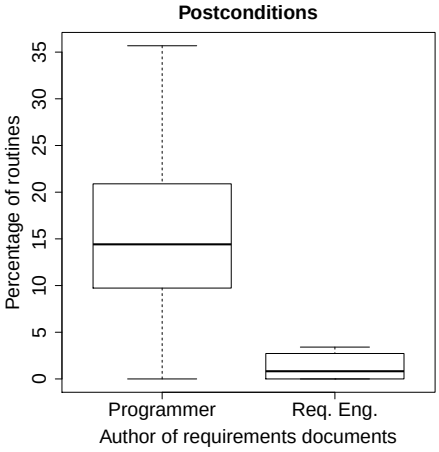


(c) Information hiding ratio (H.13)

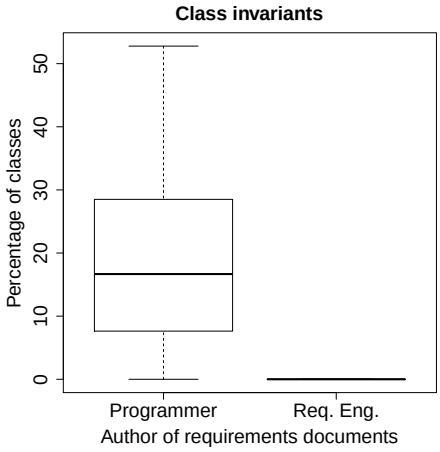
Figure 38: Box plots of metrics related to ratios (H.11 to H.13) for research question RQ.2.



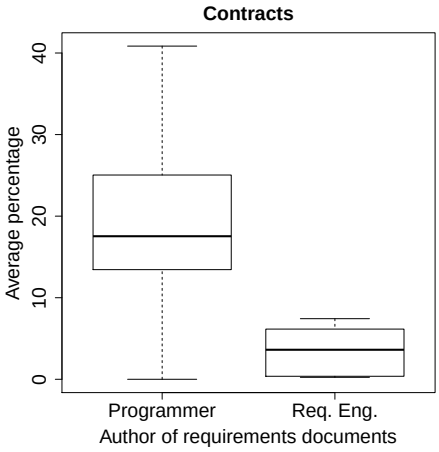
(a) Percentage of routines with preconditions (H.14)



(b) Percentage of routines with postconditions (H.15)

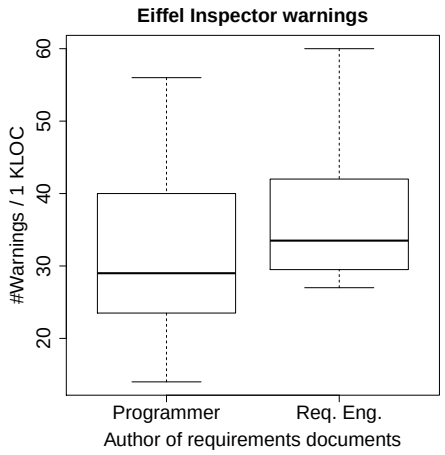


(c) Percentage of classes with class invariants (H.16)

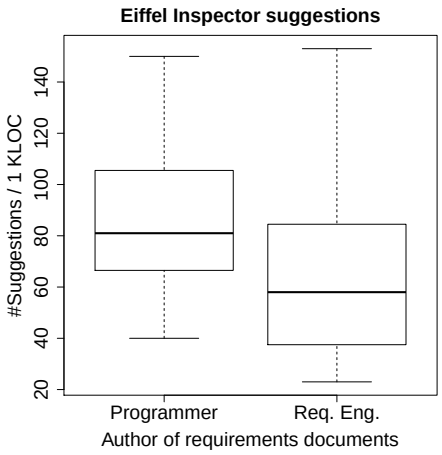


(d) Average percentage of contracts (H.17)

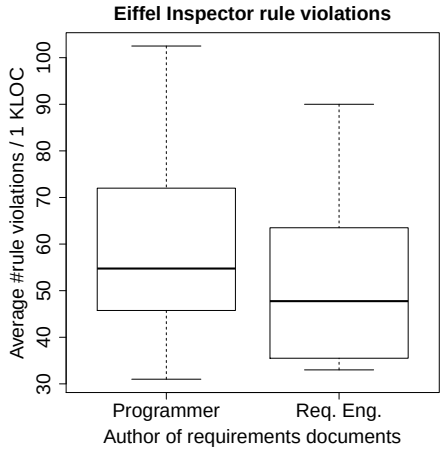
Figure 39: Box plots of metrics related to contracts (H.14 to H.17) for research question RQ.2.



(a) Number of Eiffel Inspector warnings per 1000 LOC (H.18)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.19)



(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.20)

Figure 40: Box plots of metrics related to Eiffel Inspector (H.18 to H.20) for research question RQ.2.

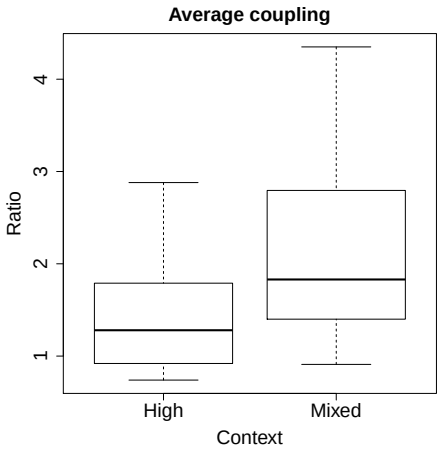
Metric	Programmer			Req. Eng.			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.11)	48	1.839	1.112	8	3.621	1.219	42.5	0*	-3.877	9.05	0.004*
Median coupling† (H.12)		0.959	0.42		3.441	1.914	27	0*	-3.653	7.113	0.008*
Information hiding (H.13)		1.347	2.044		0.876	1.169	258	0.125	0.928	15.347	0.368
Preconditions (H.14)		23.542	11.908		16.139	25.632	300.5	0.011*	0.803	7.511	0.447
Postconditions (H.15)		17.061	11.271		8.549	21.346	325	0.002*	1.103	7.663	0.304
Invariants (H.16)		20.026	17.952		0.348	0.983	357	0*	7.527	48.6	0*
Contracts (H.17)		20.21	10.991		8.345	15.528	329	0.001*	2.076	8.209	0.071
Warnings (H.18)		32.562	12.088		37.125	10.895	131	0.156	-1.079	10.104	0.306
Suggestions (H.19)		88.438	28.516		67	42.105	279.5	0.042*	1.388	8.104	0.202
Rule violations (H.20)		60.5	18.593		52.062	19.854	249.5	0.182	1.123	9.167	0.29

Table 29: Quantitative results for research question RQ.2 comparing projects from 2009 to 2012 with projects from 2014. A metric marked with a dagger (†) follows a normal distribution (Figures 107 to 116 in Appendix B.2.1 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

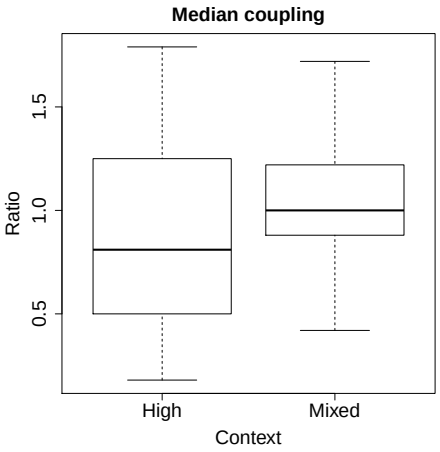
A.3 Research question RQ.3

A.3.1 Main analysis (2009 - 2012)

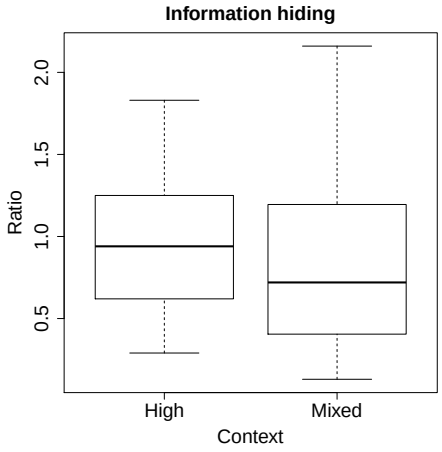
Figure 41 to 43 show the box plots for metrics of the projects done between 2009 and 2012 which the main analysis uses for research question RQ.3. Table 30 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.21)

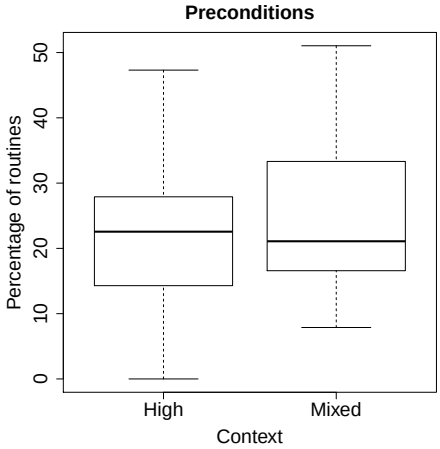


(b) Median coupling ratio (H.22)

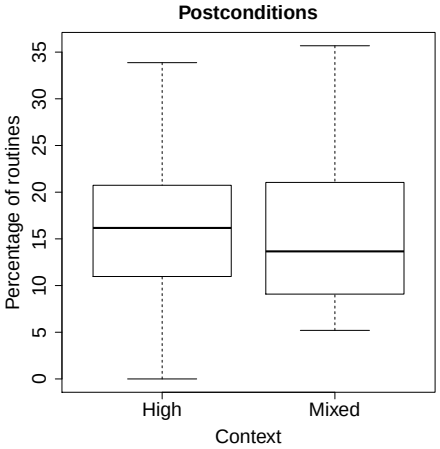


(c) Information hiding ratio (H.23)

Figure 41: Box plots of metrics related to ratios (H.21 to H.23) for research question RQ.3.

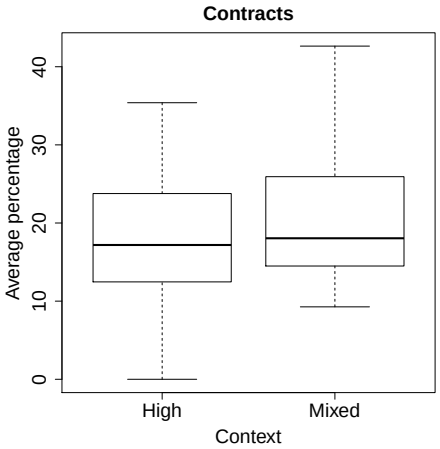


(a) Percentage of routines with preconditions (H.24)



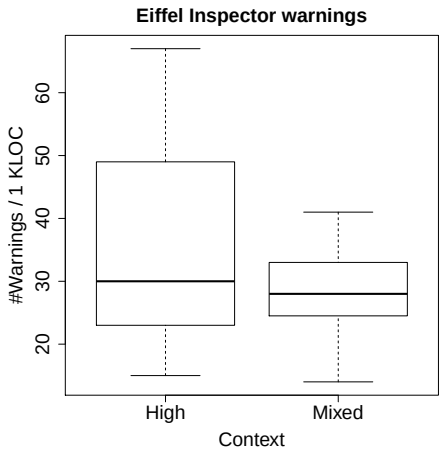
(b) Percentage of routines with postconditions (H.25)

(c) Percentage of classes with class invariants (H.26)

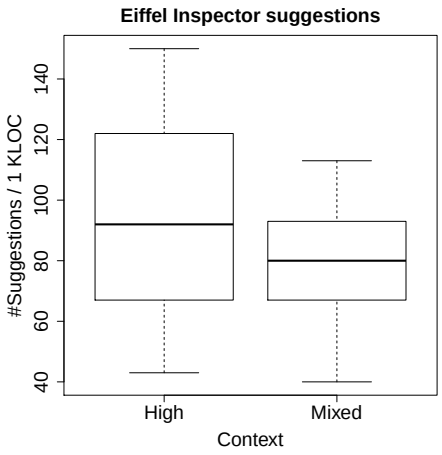


(d) Average percentage of contracts (H.27)

Figure 42: Box plots of metrics related to contracts (H.24 to H.27) for research question RQ.3.



(a) Number of Eiffel Inspector warnings per 1000 LOC (H.28)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.29)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.30)

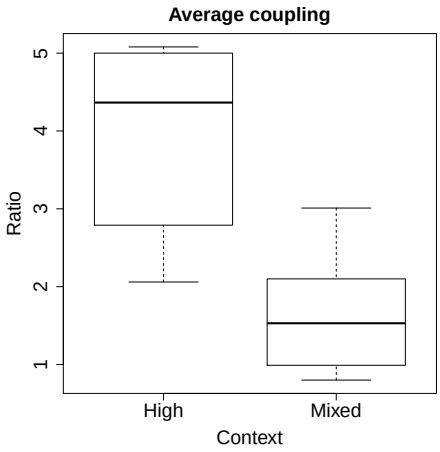
Figure 43: Box plots of metrics related to Eiffel Inspector (H.28 to H.30) for research question RQ.3.

Metric	High			Mixed			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling (H.21)	25	1.449	0.622	23	2.263	1.363	161.5	0.01*	-2.623	30.209	0.014*
Median coupling† (H.22)		0.858	0.414		1.069	0.407	213	0.127	-1.782	45.787	0.081
Information hiding (H.23)		1.289	1.297		1.411	2.661	338	0.302	-0.199	31.296	0.844
Preconditions† (H.24)		22.296	12.06		24.895	11.858	265	0.653	-0.752	45.787	0.456
Postconditions (H.25)		16.749	11.013		17.399	11.784	304.5	0.733	-0.197	44.954	0.845
Invariants (H.26)		15.49	13.763		24.957	20.81	208	0.103	-1.842	37.675	0.073
Contracts (H.27)		18.179	10		22.417	11.799	236	0.295	-1.337	43.336	0.188
Warnings (H.28)		35.64	14.691		29.217	7.367	337	0.311	1.937	35.98	0.061
Suggestions (H.29)		95.96	33.748		80.261	18.986	352.5	0.183	2.006	38.396	0.052
Rule violations (H.30)		65.8	22.713		54.739	10.447	343.5	0.252	2.196	34.326	0.035*

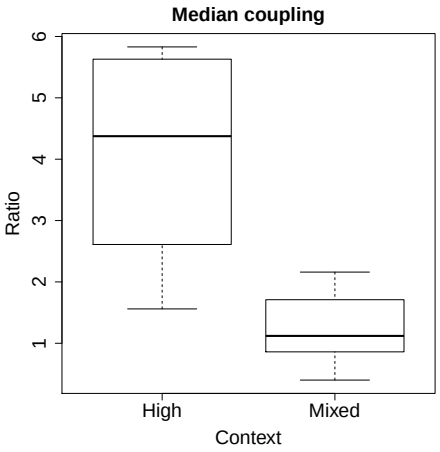
Table 30: Quantitative results for research question RQ.3 for projects from 2009 to 2012. A metric marked with a dagger (†) follows a normal distribution (Figures 147 to 156 in Appendix B.3.2 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

A.3.2 Main analysis (2013 - 2014)

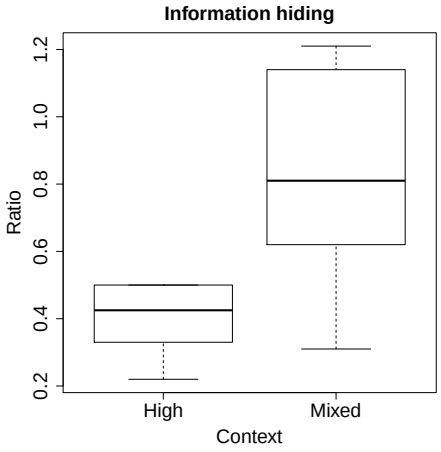
Figure 41 to 43 show the box plots for metrics of the projects done between 2009 and 2012 which the main analysis uses for research question RQ.3. Table 31 shows the results of the corresponding quantitative analysis.



(a) Average coupling ratio (H.21)

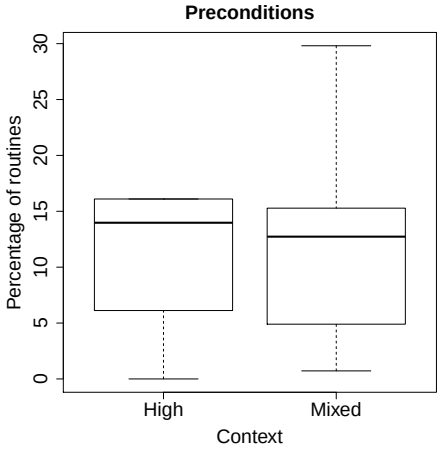


(b) Median coupling ratio (H.22)

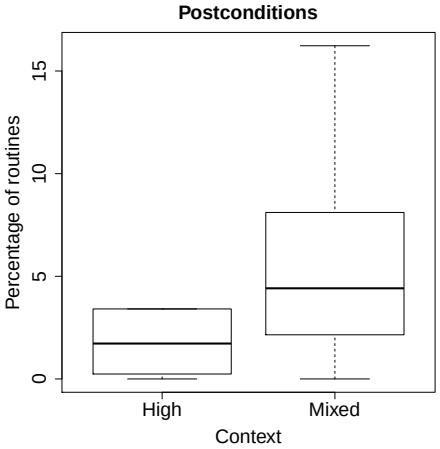


(c) Information hiding ratio (H.23)

Figure 44: Box plots of metrics related to ratios (H.21 to H.23) for research question RQ.3.

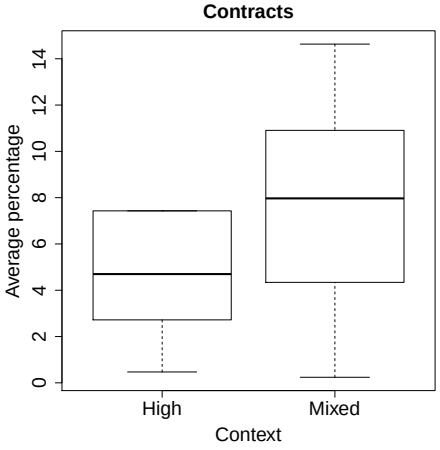


(a) Percentage of routines with preconditions (H.24)



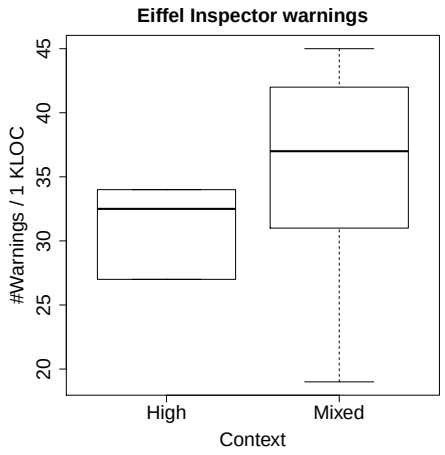
(b) Percentage of routines with postconditions (H.25)

(c) Percentage of classes with class invariants (H.26)

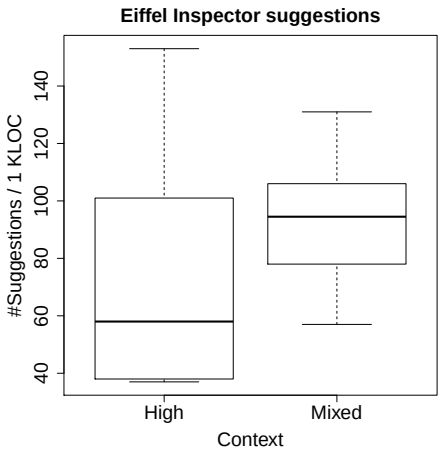


(d) Average percentage of contracts (H.27)

Figure 45: Box plots of metrics related to contracts (H.24 to H.27) for research question RQ.3.



(a) Number of Eiffel Inspector warnings per 1000 LOC (H.28)



(b) Number of Eiffel Inspector suggestions per 1000 LOC (H.29)

(c) Average number of Eiffel Inspector rule violations per 1000 LOC (H.30)

Figure 46: Box plots of metrics related to Eiffel Inspector (H.28 to H.30) for research question RQ.3.

Metric	High			Mixed			U-test		t-test		
	#	Mean	SD	#	Mean	SD	U	p	t	df	p
Average coupling† (H.21)	6	3.943	1.237	14	1.626	0.664	79	0.001*	4.328	6.274	0.004*
Median coupling† (H.22)		4.063	1.771		1.229	0.54	79	0.001*	3.845	5.403	0.01*
Information hiding (H.23)		0.502	0.314		1.054	0.859	18	0.053	-2.099	17.857	0.05
Preconditions (H.24)		21.265	28.172		11.573	8.032	47	0.718	0.828	5.352	0.443
Postconditions (H.25)		11.398	24.474		5.407	4.554	27	0.231	0.595	5.149	0.577
Invariants (H.26)		0.463	1.135		9.252	10.784	17.5	0.034*	-3.011	13.657	0.01*
Contracts (H.27)		11.042	17.397		8.744	6.796	32	0.444	0.313	5.666	0.765
Warnings (H.28)		35.5	12.373		35.643	7.672	32.5	0.457	-0.026	6.713	0.98
Suggestions† (H.29)		74.167	45.088		89.643	28.351	28	0.274	-0.777	6.763	0.463
Rule violations† (H.30)		54.833	21.651		62.643	13.8	31	0.386	-0.815	6.813	0.442

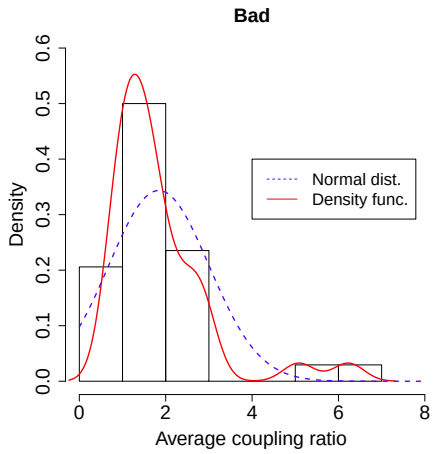
Table 31: Quantitative results for research question RQ.3 for projects done in 2013 and 2014. A metric marked with a dagger (†) follows a normal distribution (Figures 157 to 166 in Appendix B.3.3 for the corresponding normality tests). An asterisk (∗) marks statistically significant results, that is $p < \alpha = 0.05$. # stands for *number of projects* and *SD* stands for *standard deviation*.

B Normality tests

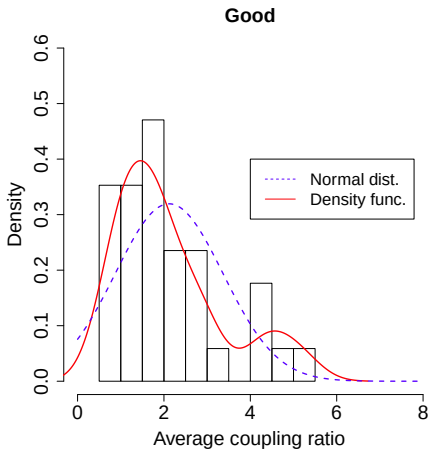
B.1 Research question RQ.1

B.1.1 Main analysis (2009 - 2014)

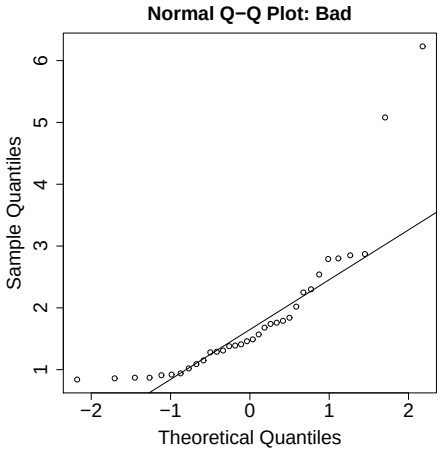
Figure 47 to 56 show the normality tests performed for the main analyis of research question RQ.1 using all projects.



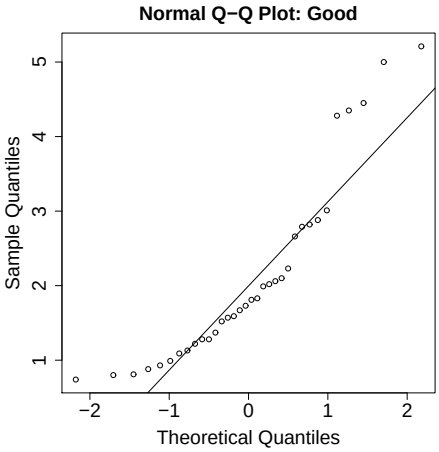
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

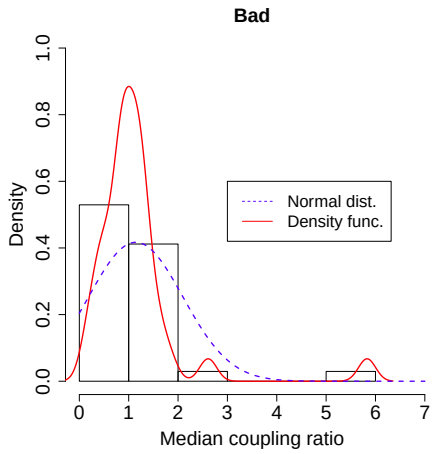
$$W = 0.7433234,$$
$$p = 2.441816\text{e-}06$$

(e) Shapiro-Wilk normality test for bad requirements documents.

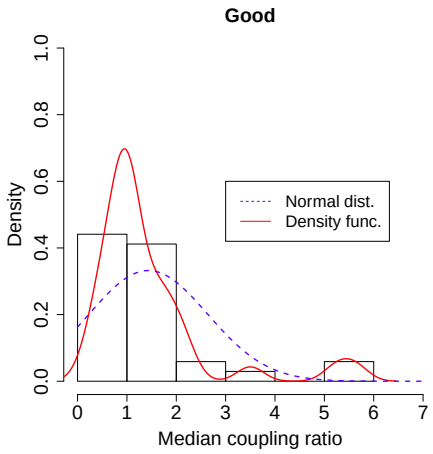
$$W = 0.8603522,$$
$$p = 0.0004810753$$

(f) Shapiro-Wilk normality test for good requirements documents.

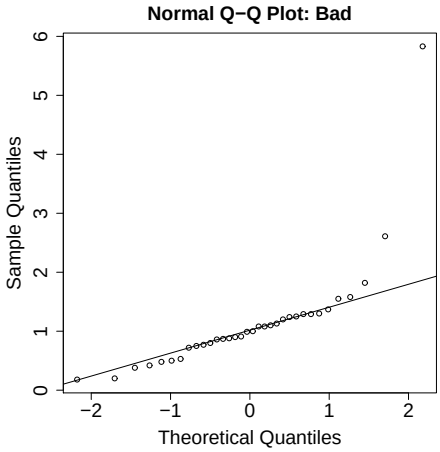
Figure 47: Normality tests for average coupling ratio of projects with bad requirements documents and projects with good requirements documents.



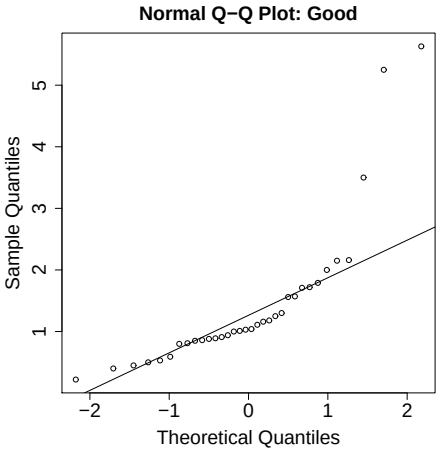
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

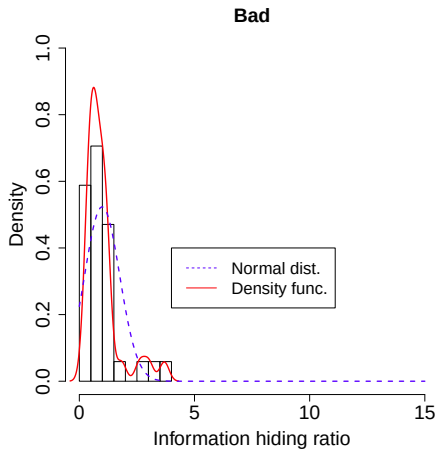
$$W = 0.6169111,$$
$$p = 3.255645\text{e-}08$$

(e) Shapiro-Wilk normality test for bad requirements documents.

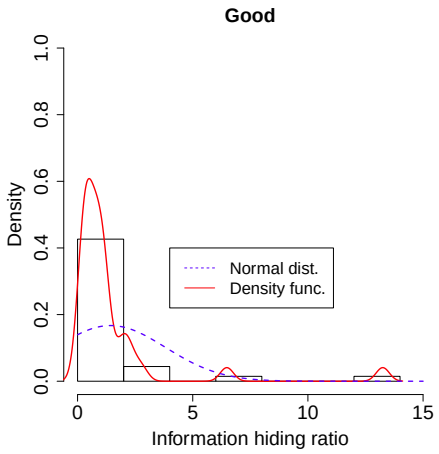
$$W = 0.7073995,$$
$$p = 6.405065\text{e-}07$$

(f) Shapiro-Wilk normality test for good requirements documents.

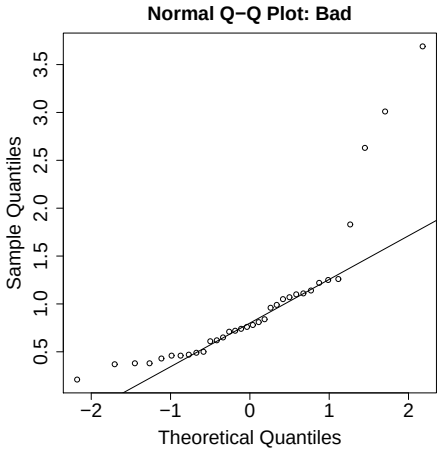
Figure 48: Normality tests for median coupling ratio of projects with bad requirements documents and projects with good requirements documents.



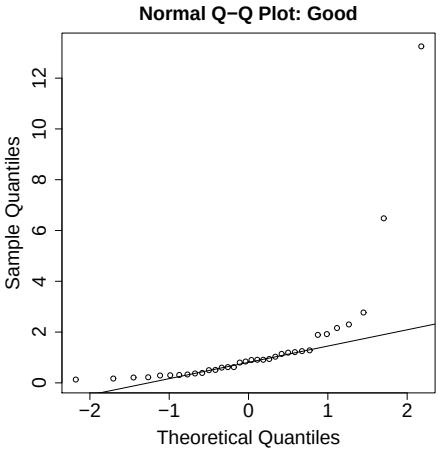
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

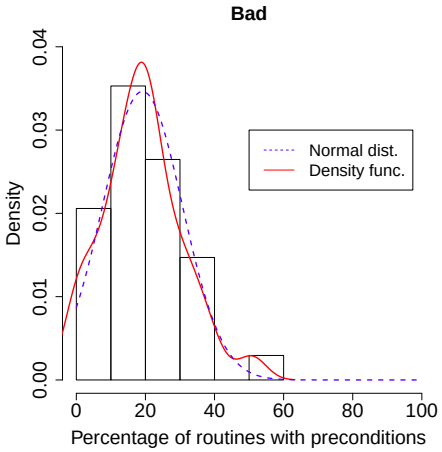
$$W = 0.7393039,$$
$$p = 2.091222\text{e-}06$$

(e) Shapiro-Wilk normality test for bad requirements documents.

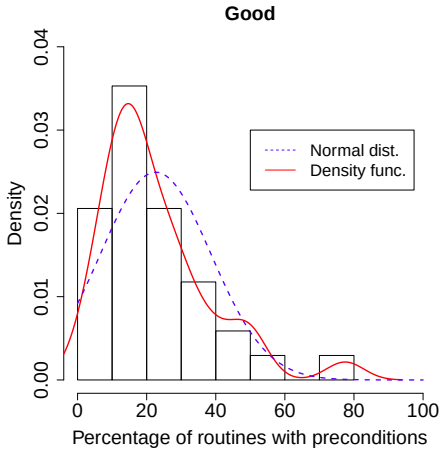
$$W = 0.479877,$$
$$p = 7.640863\text{e-}10$$

(f) Shapiro-Wilk normality test for good requirements documents.

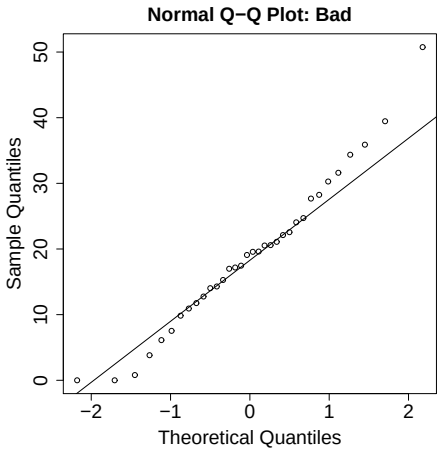
Figure 49: Normality tests for information hiding ratio of projects with bad requirements documents and projects with good requirements documents.



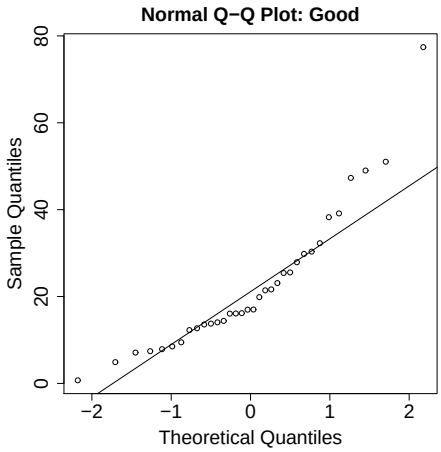
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

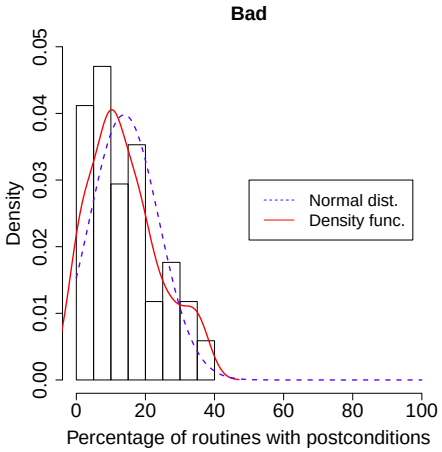
$$W = 0.9744244,$$
$$p = 0.5931857$$

(e) Shapiro-Wilk normality test for bad requirements documents.

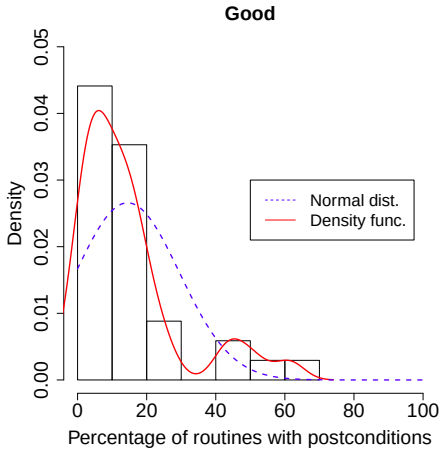
$$W = 0.8779952,$$
$$p = 0.001264165$$

(f) Shapiro-Wilk normality test for good requirements documents.

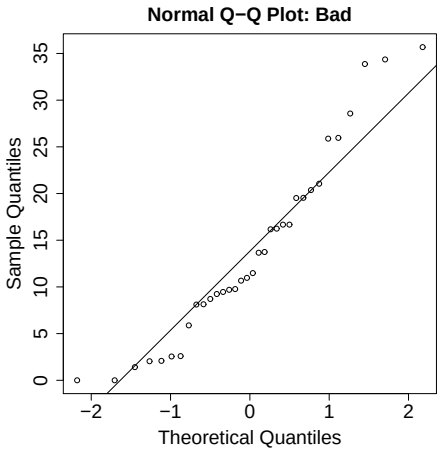
Figure 50: Normality tests for percentage of routines with preconditions of projects with bad requirements documents and projects with good requirements documents.



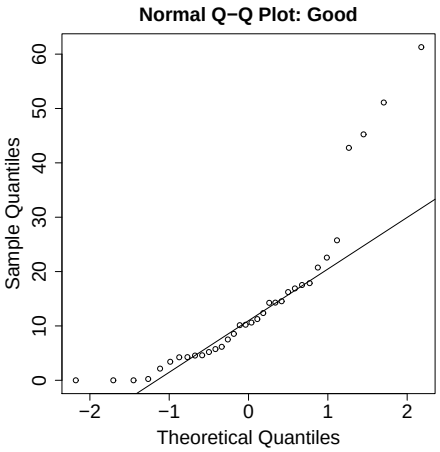
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

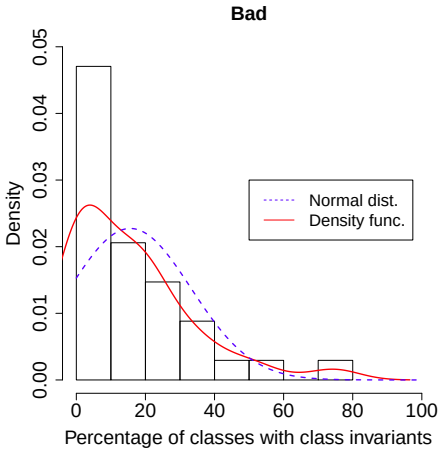
$W = 0.9370825$,
$p = 0.05050214$

(e) Shapiro-Wilk normality test for bad requirements documents.

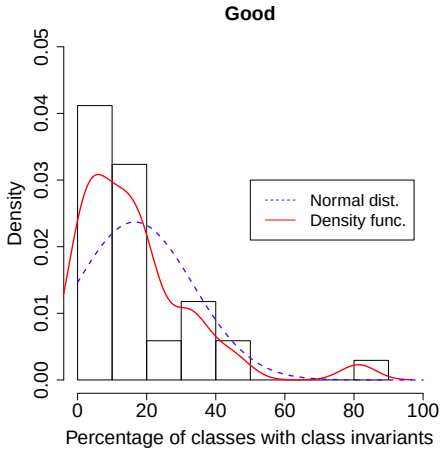
$W = 0.7976854$,
$p = 2.312512\text{e-}05$

(f) Shapiro-Wilk normality test for good requirements documents.

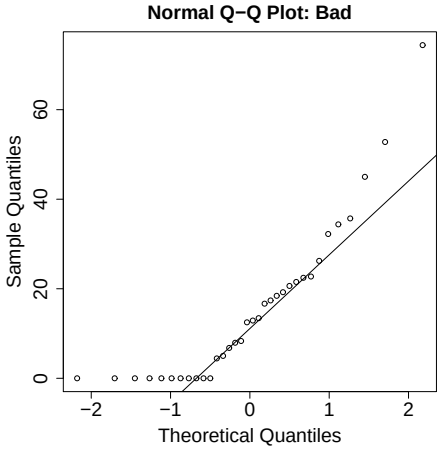
Figure 51: Normality tests for percentage of routines with postconditions of projects with bad requirements documents and projects with good requirements documents.



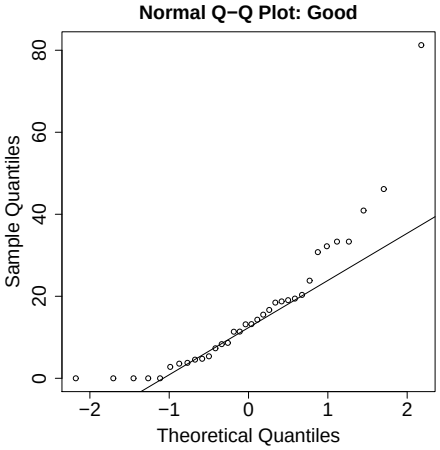
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

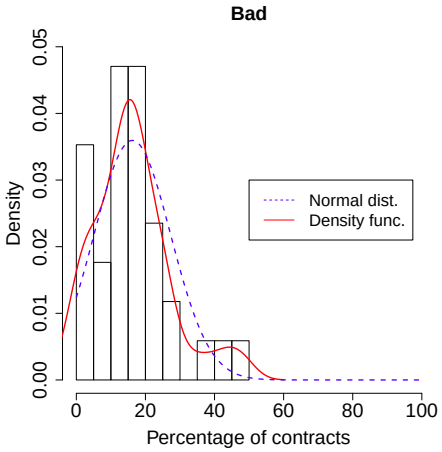
$$W = 0.8350313,$$
$$p = 0.0001318442$$

(e) Shapiro-Wilk normality test for bad requirements documents.

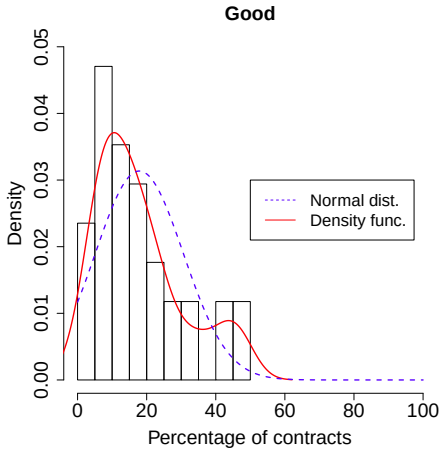
$$W = 0.8226423,$$
$$p = 7.250162\text{e-}05$$

(f) Shapiro-Wilk normality test for good requirements documents.

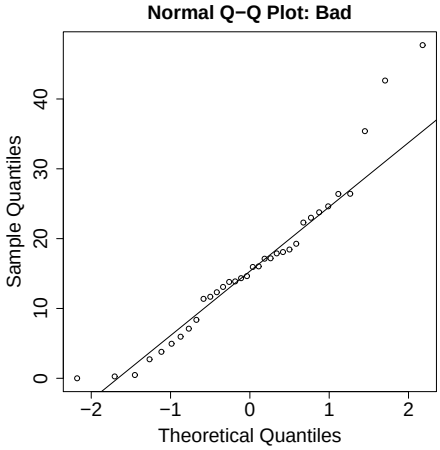
Figure 52: Normality tests for percentage of classes with class invariants of projects with bad requirements documents and projects with good requirements documents.



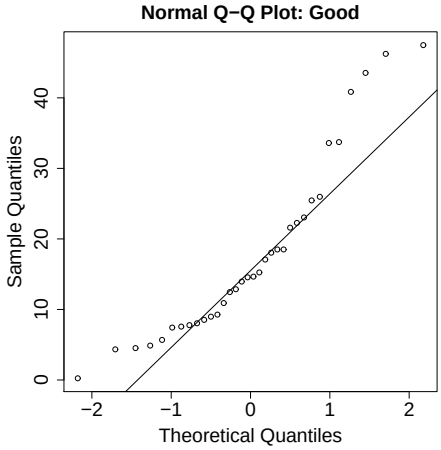
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

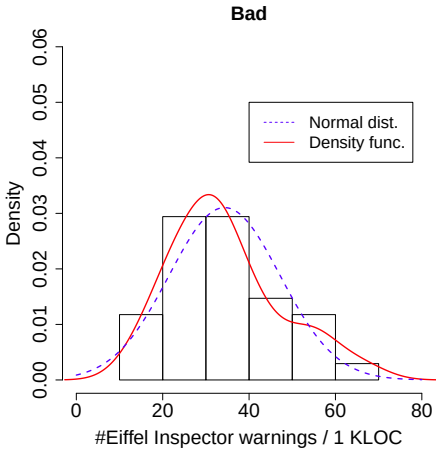
$$W = 0.9323069,$$
$$p = 0.03653057$$

(e) Shapiro-Wilk normality test for bad requirements documents.

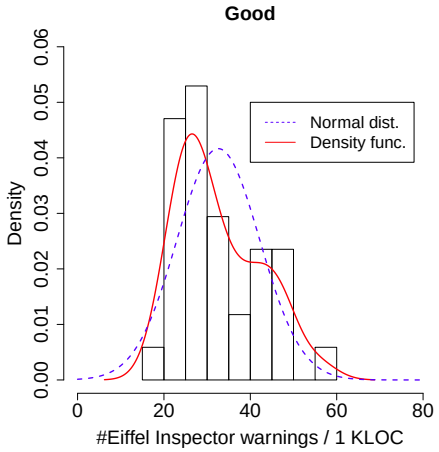
$$W = 0.8932751,$$
$$p = 0.003061569$$

(f) Shapiro-Wilk normality test for good requirements documents.

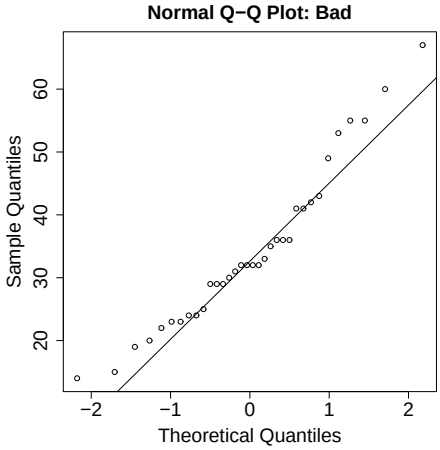
Figure 53: Normality tests for average percentage of contracts of projects with bad requirements documents and projects with good requirements documents.



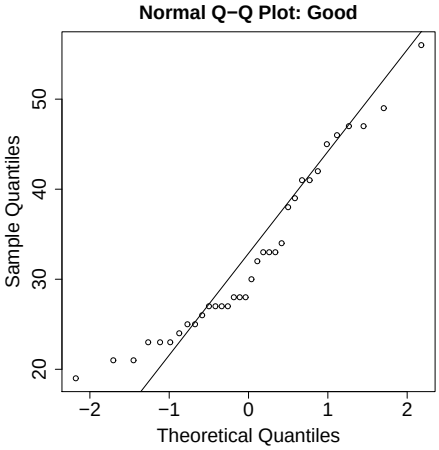
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

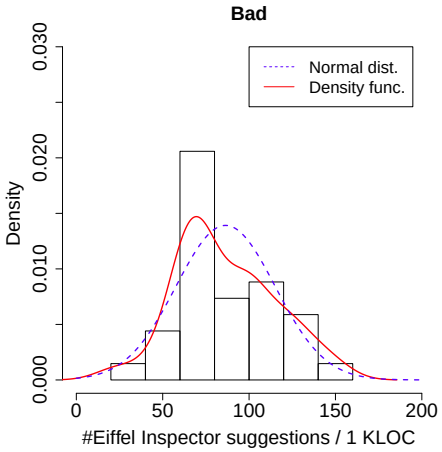
$$W = 0.9458538,$$
$$p = 0.09222797$$

(e) Shapiro-Wilk normality test for bad requirements documents.

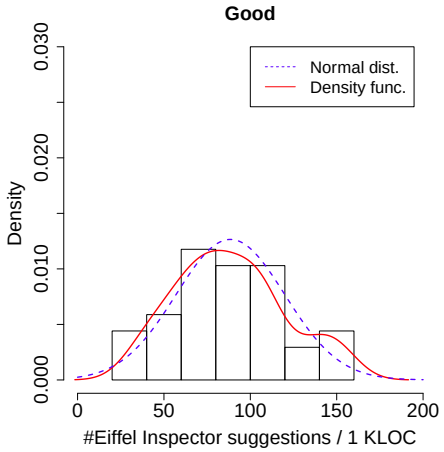
$$W = 0.925829,$$
$$p = 0.02369048$$

(f) Shapiro-Wilk normality test for good requirements documents.

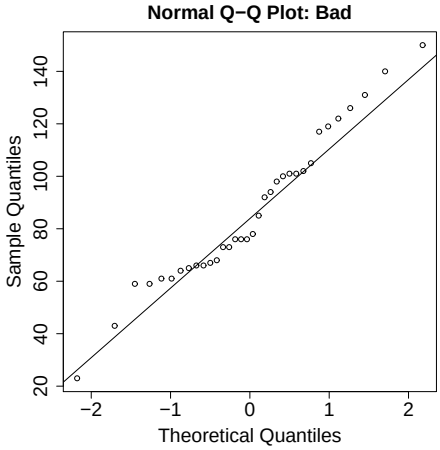
Figure 54: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



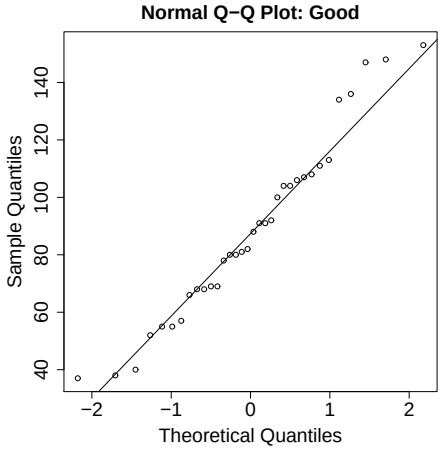
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

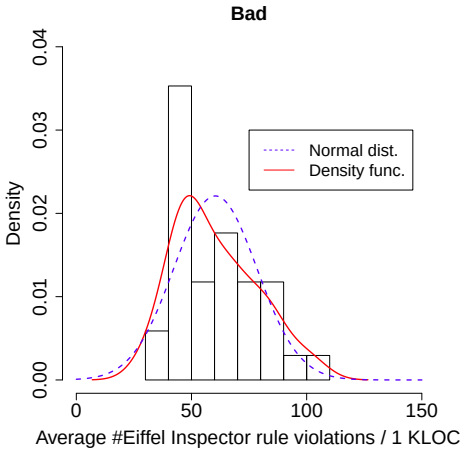
$$W = 0.9667239,$$
$$p = 0.3772567$$

(e) Shapiro-Wilk normality test for bad requirements documents.

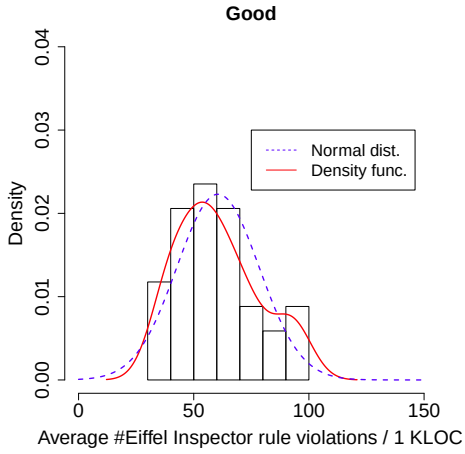
$$W = 0.9629235,$$
$$p = 0.2952765$$

(f) Shapiro-Wilk normality test for good requirements documents.

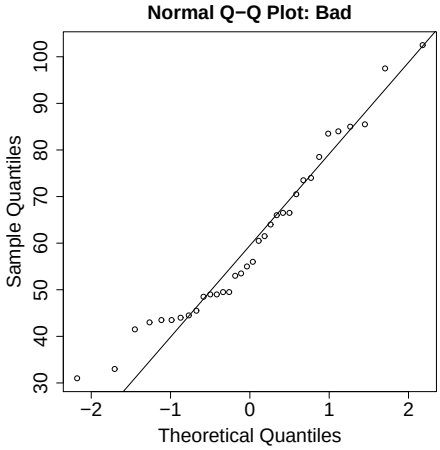
Figure 55: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



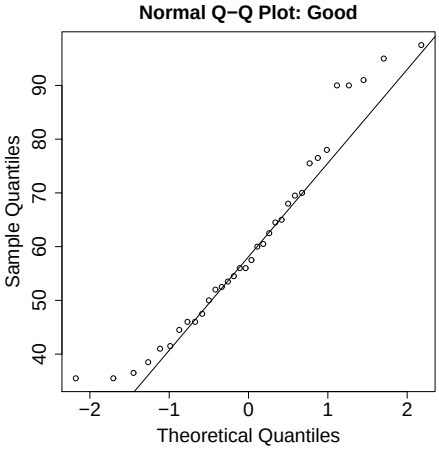
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

$$W = 0.9497274,$$
$$p = 0.1205007$$

(e) Shapiro-Wilk normality test for bad requirements documents.

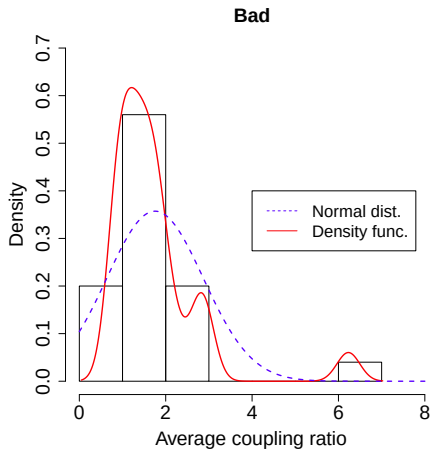
$$W = 0.941903,$$
$$p = 0.07025326$$

(f) Shapiro-Wilk normality test for good requirements documents.

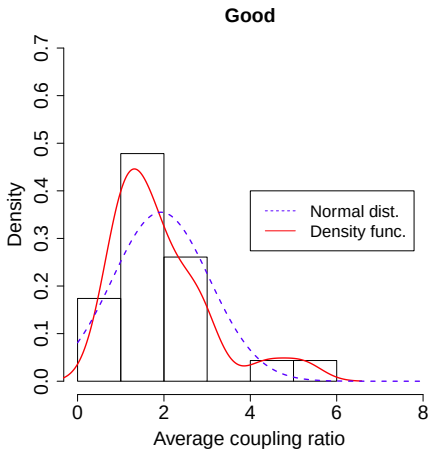
Figure 56: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.

B.1.2 Main analysis (2009 - 2012)

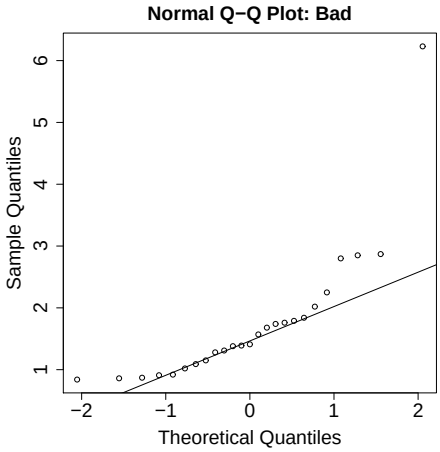
Figure 57 to 66 show the normality tests performed for the main analyis of research question RQ.1 using the projects from 2009 to 2012.



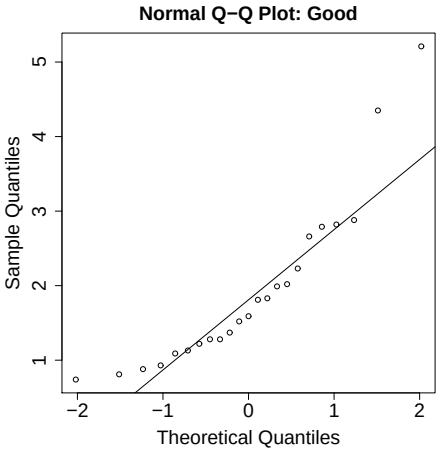
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

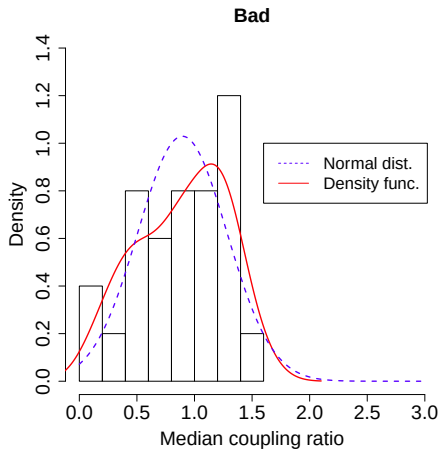
$$W = 0.6944883,$$
$$p = 6.149601\text{e-}06$$

(e) Shapiro-Wilk normality test for bad requirements documents.

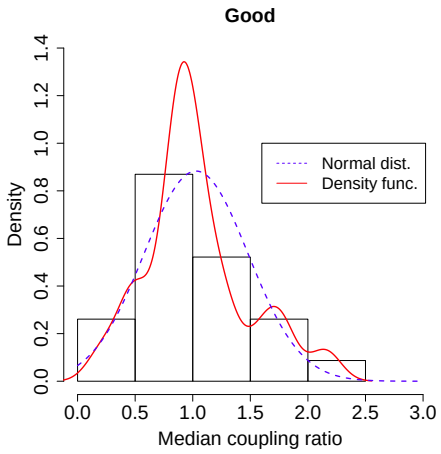
$$W = 0.8482277,$$
$$p = 0.00252692$$

(f) Shapiro-Wilk normality test for good requirements documents.

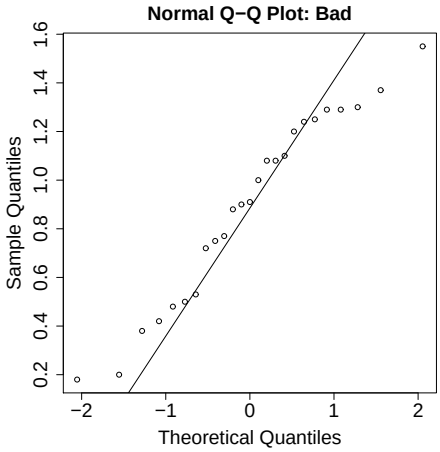
Figure 57: Normality tests for average coupling ratio of projects with bad requirements documents and projects with good requirements documents.



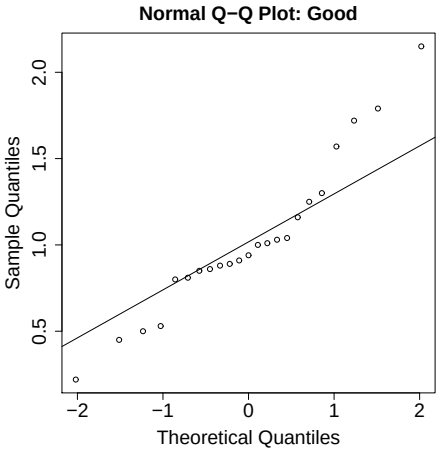
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

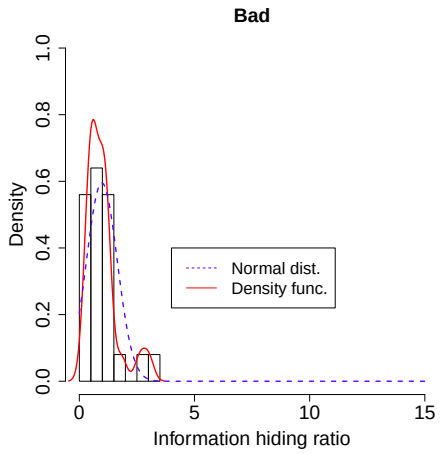
$$W = 0.9527092,$$
$$p = 0.2883095$$

(e) Shapiro-Wilk normality test for bad requirements documents.

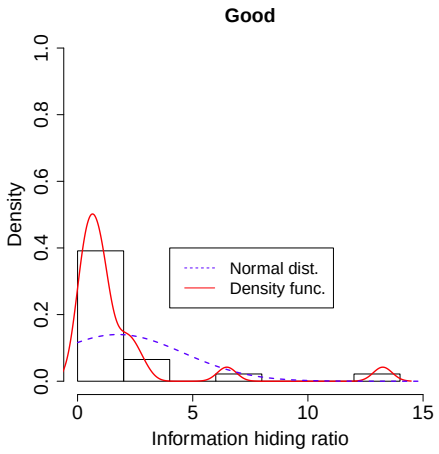
$$W = 0.9426501,$$
$$p = 0.2048001$$

(f) Shapiro-Wilk normality test for good requirements documents.

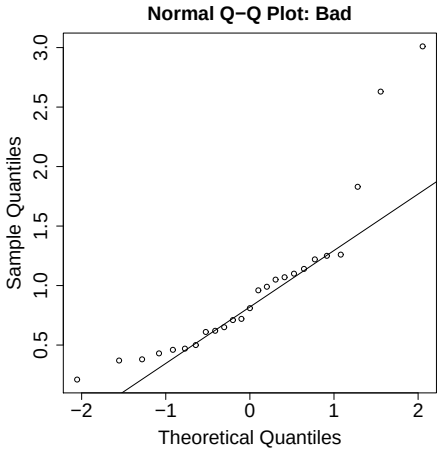
Figure 58: Normality tests for median coupling ratio of projects with bad requirements documents and projects with good requirements documents.



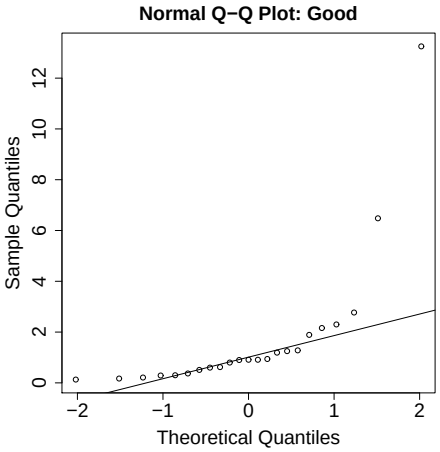
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

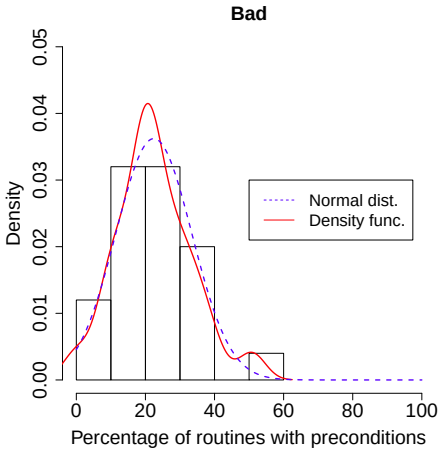
$$W = 0.8131783,$$
$$p = 0.0003802438$$

(e) Shapiro-Wilk normality test for bad requirements documents.

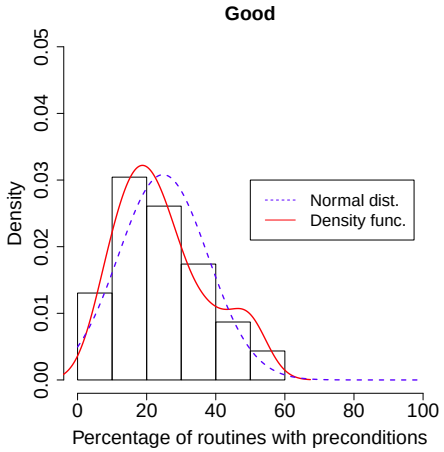
$$W = 0.5358539,$$
$$p = 1.921032\text{e-}07$$

(f) Shapiro-Wilk normality test for good requirements documents.

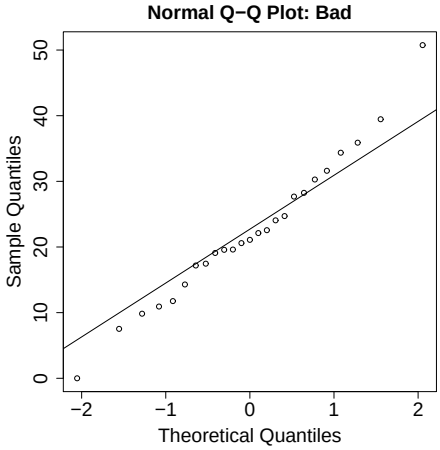
Figure 59: Normality tests for information hiding ratio of projects with bad requirements documents and projects with good requirements documents.



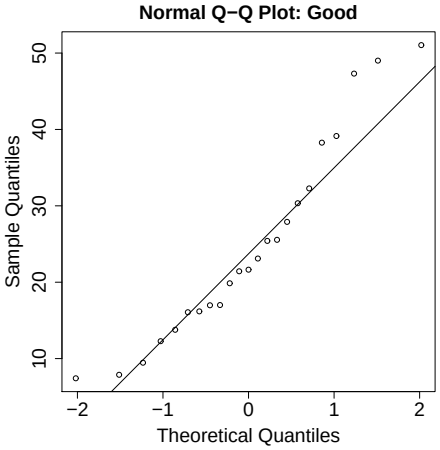
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

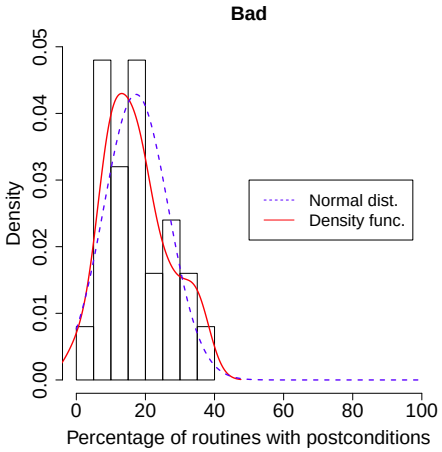
$$W = 0.9825417,$$
$$p = 0.930398$$

(e) Shapiro-Wilk normality test for bad requirements documents.

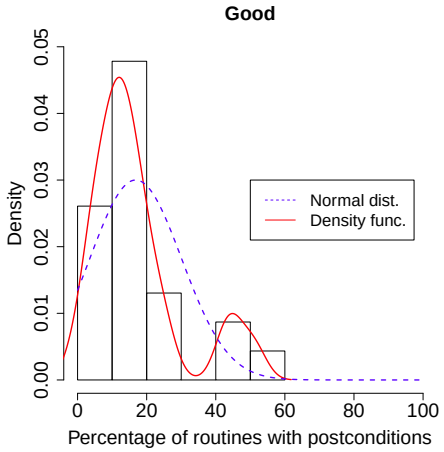
$$W = 0.9282496,$$
$$p = 0.1002483$$

(f) Shapiro-Wilk normality test for good requirements documents.

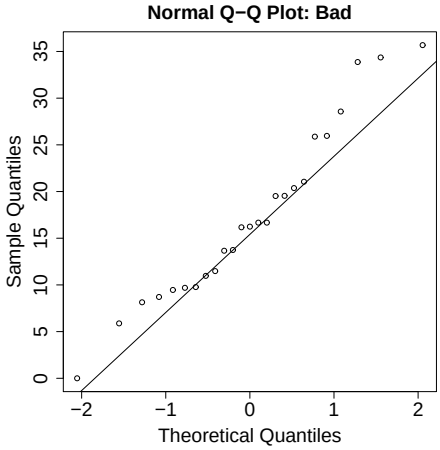
Figure 60: Normality tests for percentage of routines with preconditions of projects with bad requirements documents and projects with good requirements documents.



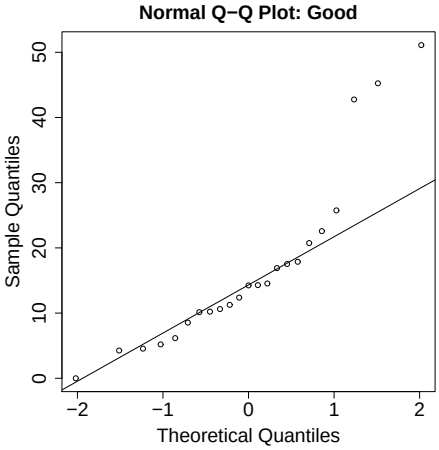
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

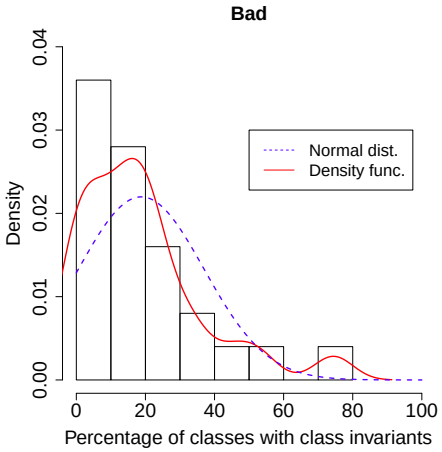
$$W = 0.9537694,$$
$$p = 0.3043331$$

(e) Shapiro-Wilk normality test for bad requirements documents.

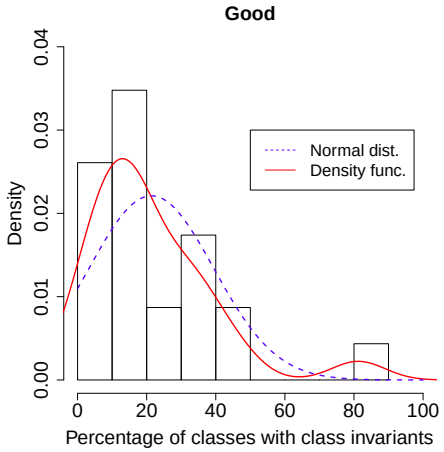
$$W = 0.8413859,$$
$$p = 0.001906438$$

(f) Shapiro-Wilk normality test for good requirements documents.

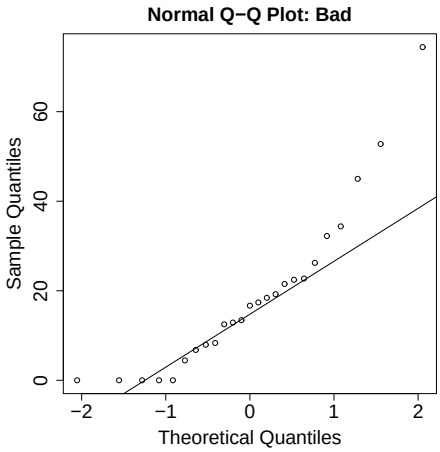
Figure 61: Normality tests for percentage of routines with postconditions of projects with bad requirements documents and projects with good requirements documents.



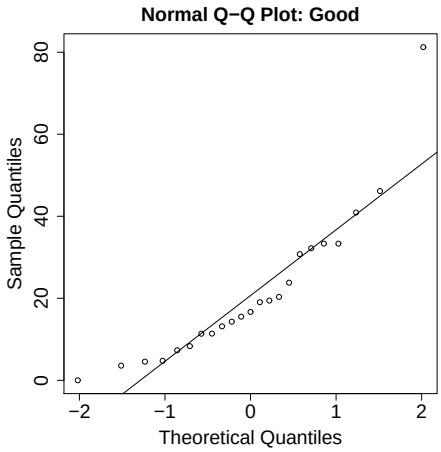
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

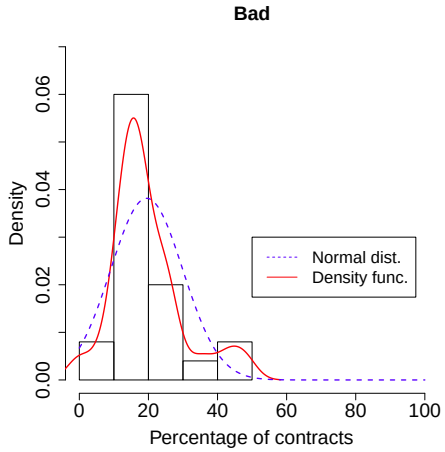
$$W = 0.8661847,$$
$$p = 0.003630437$$

(e) Shapiro-Wilk normality test for bad requirements documents.

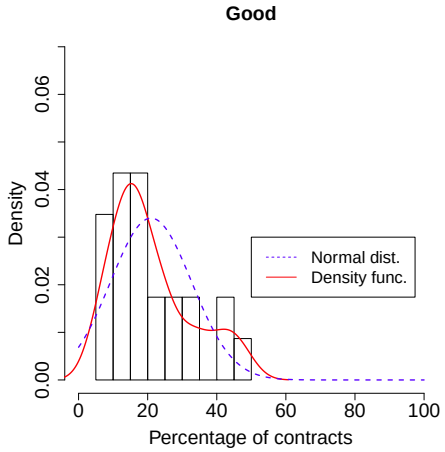
$$W = 0.8520365,$$
$$p = 0.002962534$$

(f) Shapiro-Wilk normality test for good requirements documents.

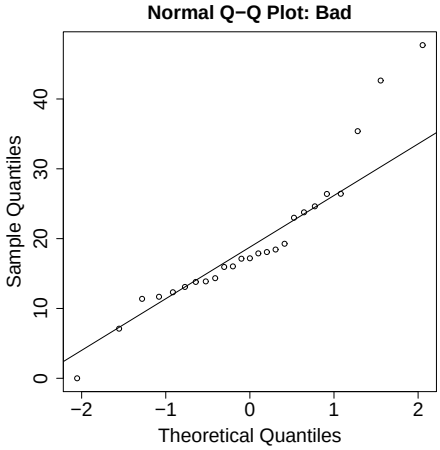
Figure 62: Normality tests for percentage of classes with class invariants of projects with bad requirements documents and projects with good requirements documents.



(a) Histogram for bad requirements documents.



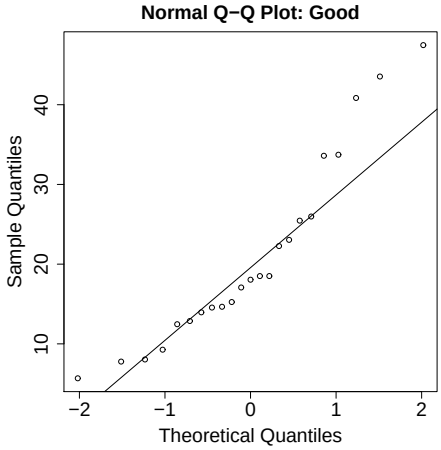
(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.

$$W = 0.902657,$$
$$p = 0.02096977$$

(e) Shapiro-Wilk normality test for bad requirements documents.

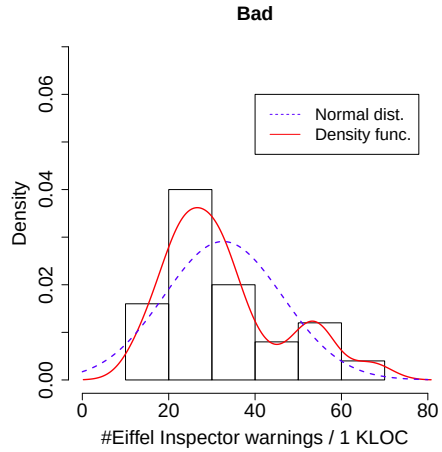


(d) QQ-Plot for good requirements documents.

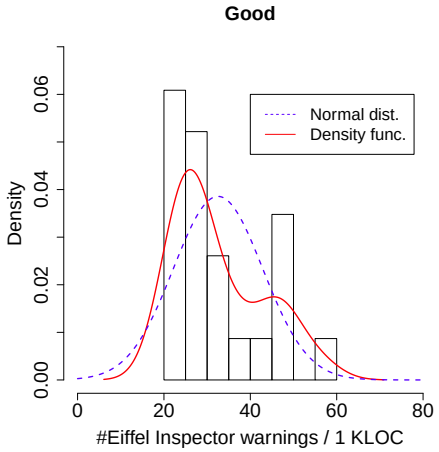
$$W = 0.9069548,$$
$$p = 0.03525516$$

(f) Shapiro-Wilk normality test for good requirements documents.

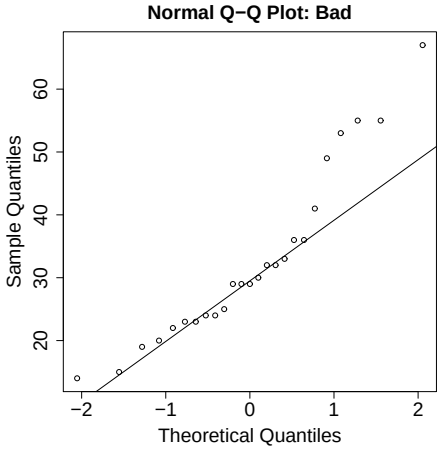
Figure 63: Normality tests for average percentage of contracts of projects with bad requirements documents and projects with good requirements documents.



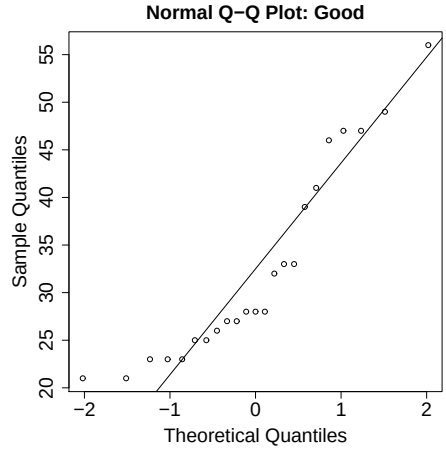
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

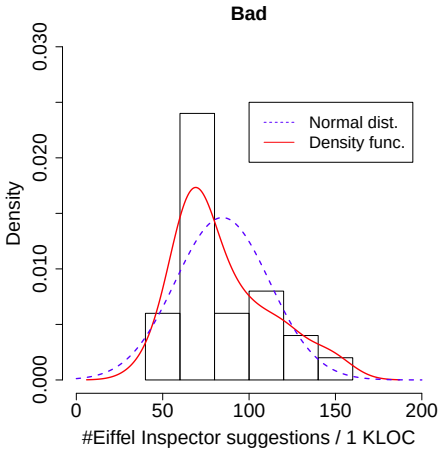
$$W = 0.9088737,$$
$$p = 0.02876841$$

(e) Shapiro-Wilk normality test for bad requirements documents.

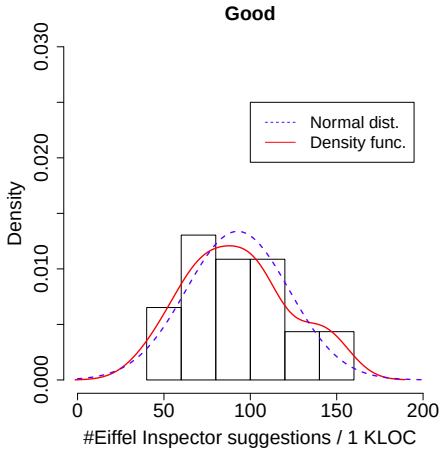
$$W = 0.8754338,$$
$$p = 0.008155464$$

(f) Shapiro-Wilk normality test for good requirements documents.

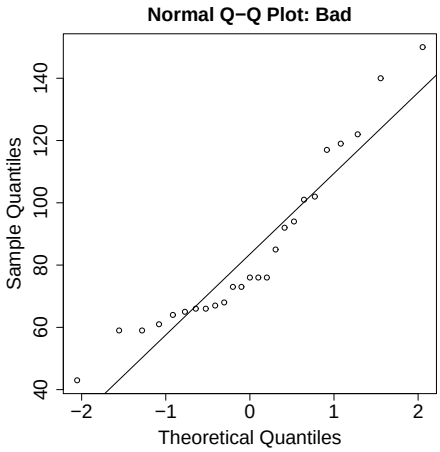
Figure 64: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



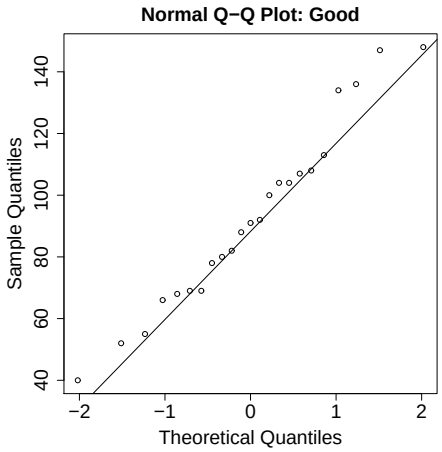
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

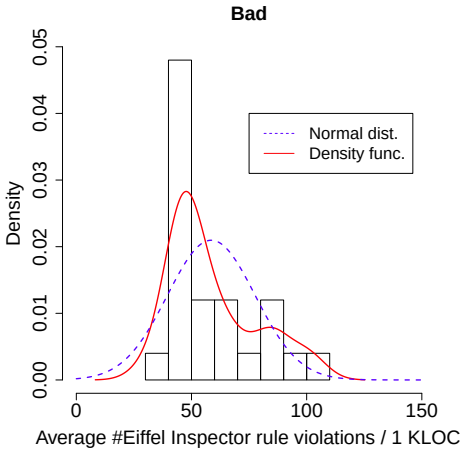
$$W = 0.903417,$$
$$p = 0.02179064$$

(e) Shapiro-Wilk normality test for bad requirements documents.

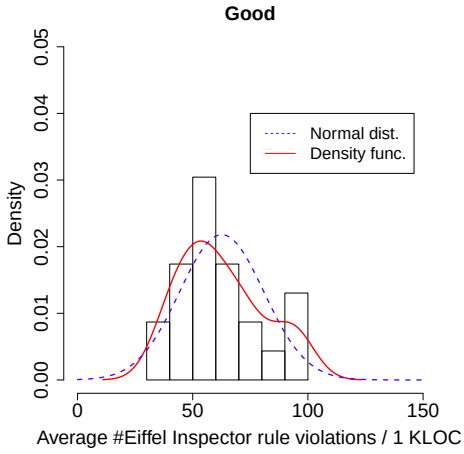
$$W = 0.9660516,$$
$$p = 0.5954575$$

(f) Shapiro-Wilk normality test for good requirements documents.

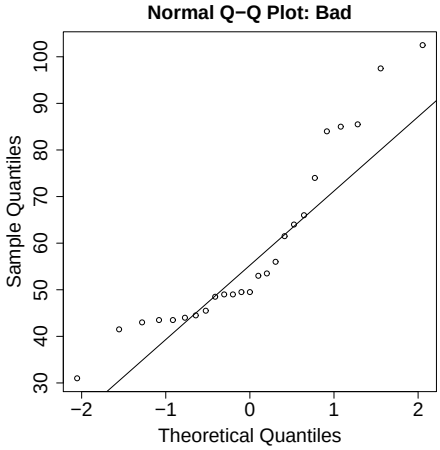
Figure 65: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



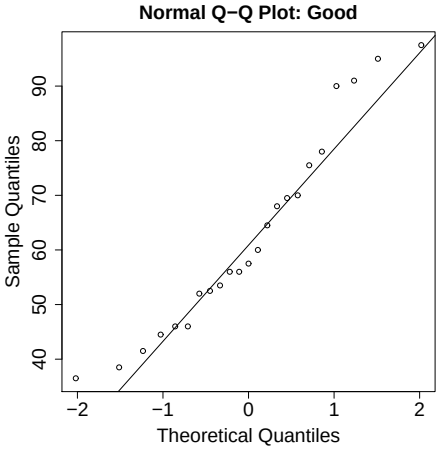
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

$$W = 0.8753278,$$
$$p = 0.005543235$$

(e) Shapiro-Wilk normality test for bad requirements documents.

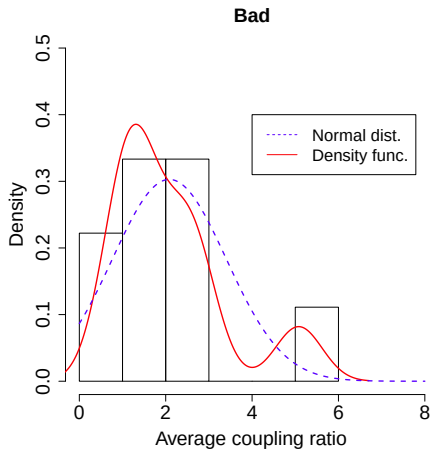
$$W = 0.9379924,$$
$$p = 0.1627104$$

(f) Shapiro-Wilk normality test for good requirements documents.

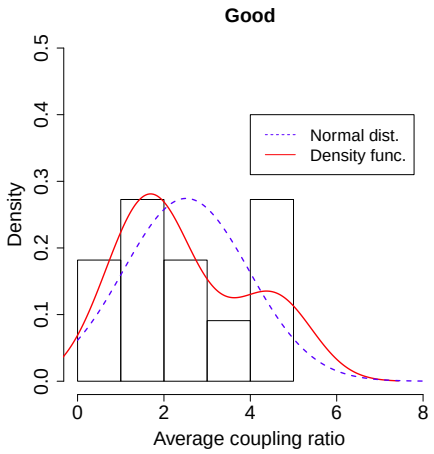
Figure 66: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.

B.1.3 Main analysis (2013 - 2014)

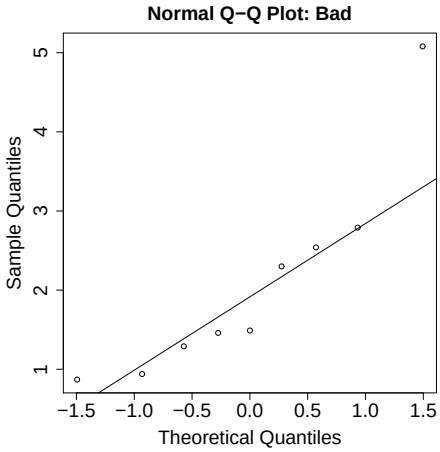
Figure 67 to 76 show the normality tests performed for the main analyis of research question RQ.1 using the projects from 2013 and 2014.



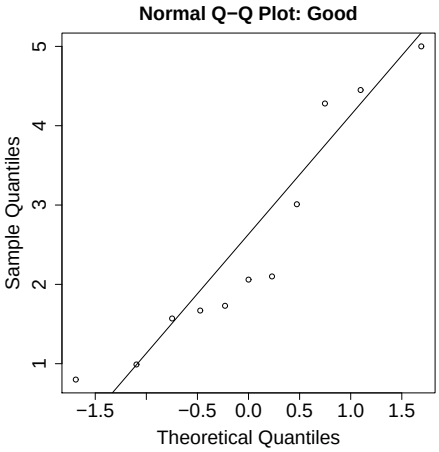
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

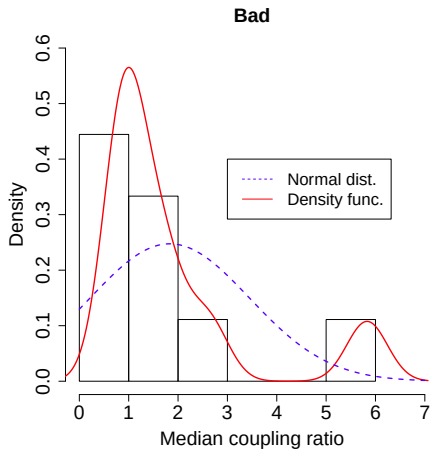
$$W = 0.8354418,$$
$$p = 0.05137814$$

(e) Shapiro-Wilk normality test for bad requirements documents.

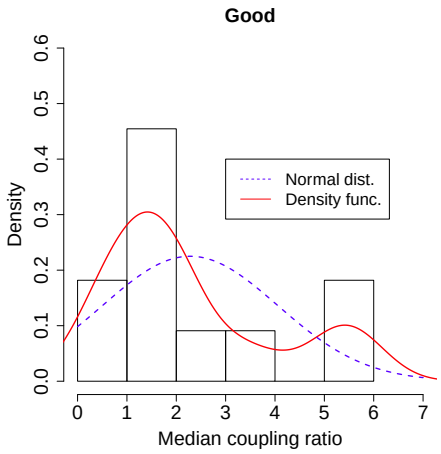
$$W = 0.8862678,$$
$$p = 0.1248668$$

(f) Shapiro-Wilk normality test for good requirements documents.

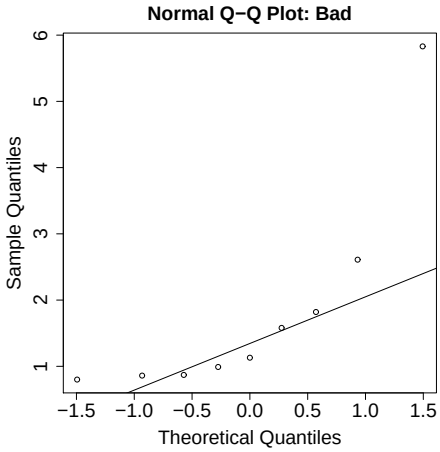
Figure 67: Normality tests for average coupling ratio of projects with bad requirements documents and projects with good requirements documents.



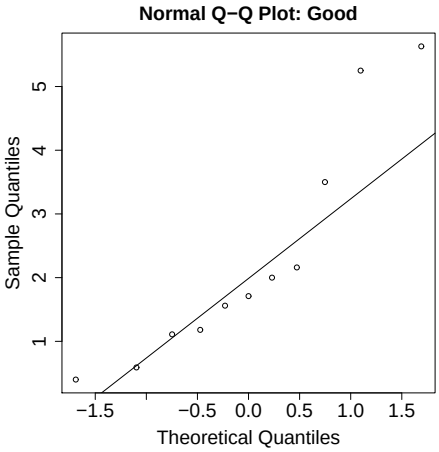
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

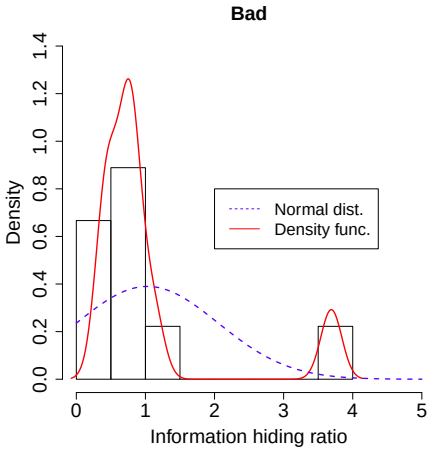
$$W = 0.6846764,$$
$$p = 0.0009397463$$

(e) Shapiro-Wilk normality test for bad requirements documents.

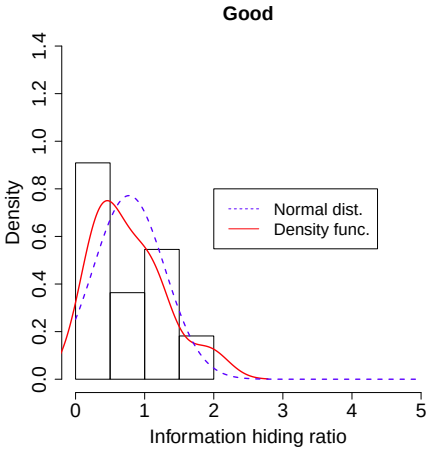
$$W = 0.8551713,$$
$$p = 0.04983947$$

(f) Shapiro-Wilk normality test for good requirements documents.

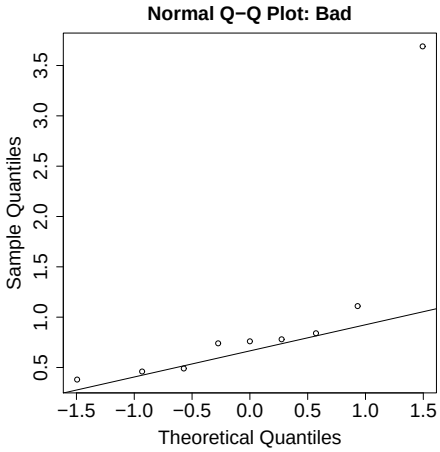
Figure 68: Normality tests for median coupling ratio of projects with bad requirements documents and projects with good requirements documents.



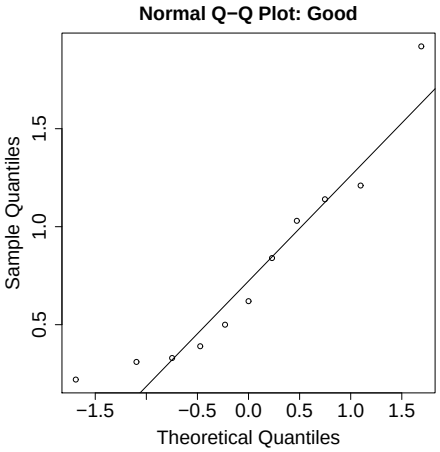
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

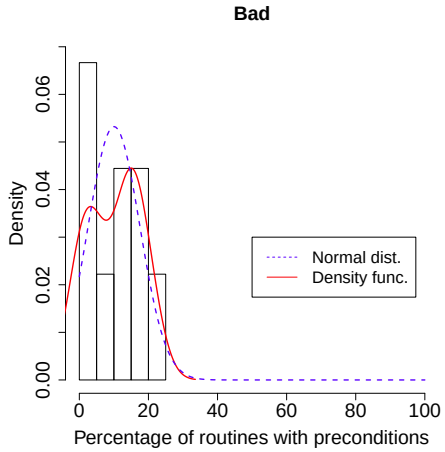
$$W = 0.5949602,$$
$$p = 8.349409\text{e-}05$$

(e) Shapiro-Wilk normality test for bad requirements documents.

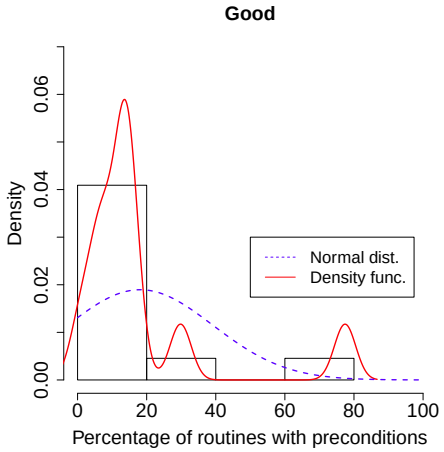
$$W = 0.8975609,$$
$$p = 0.172559$$

(f) Shapiro-Wilk normality test for good requirements documents.

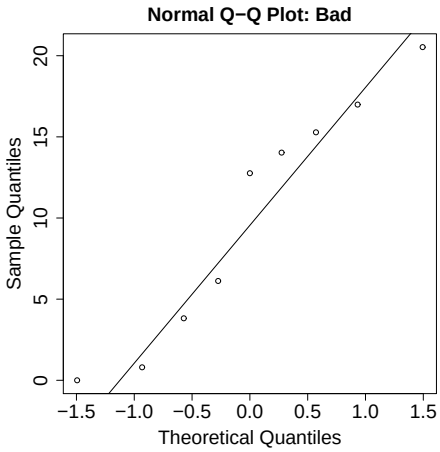
Figure 69: Normality tests for information hiding ratio of projects with bad requirements documents and projects with good requirements documents.



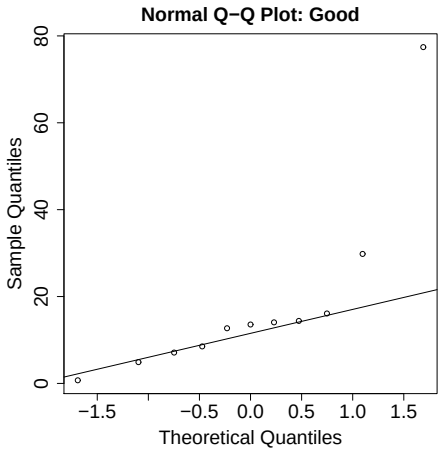
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

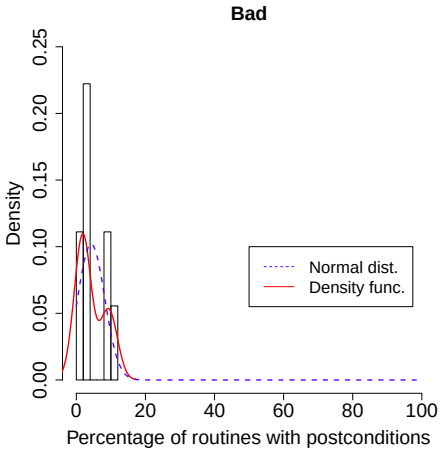
$$W = 0.9223862,$$
$$p = 0.4123615$$

(e) Shapiro-Wilk normality test for bad requirements documents.

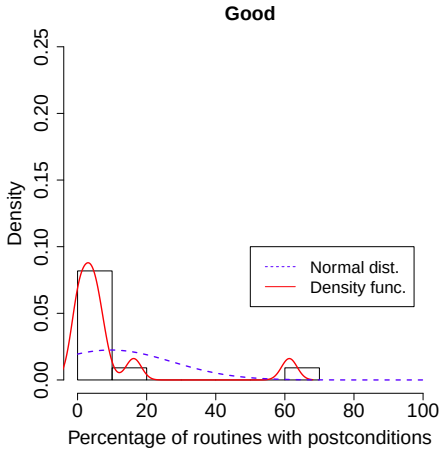
$$W = 0.6628371,$$
$$p = 0.0001567055$$

(f) Shapiro-Wilk normality test for good requirements documents.

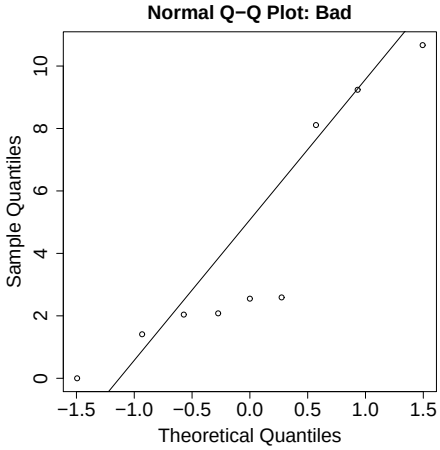
Figure 70: Normality tests for percentage of routines with preconditions of projects with bad requirements documents and projects with good requirements documents.



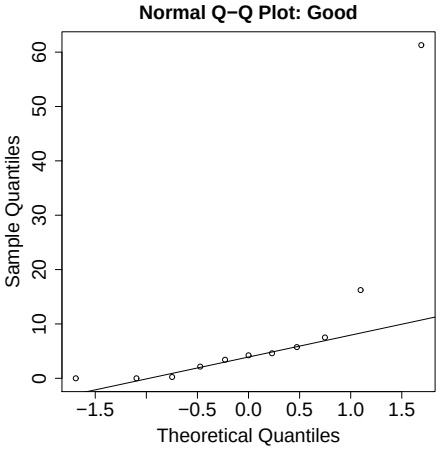
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

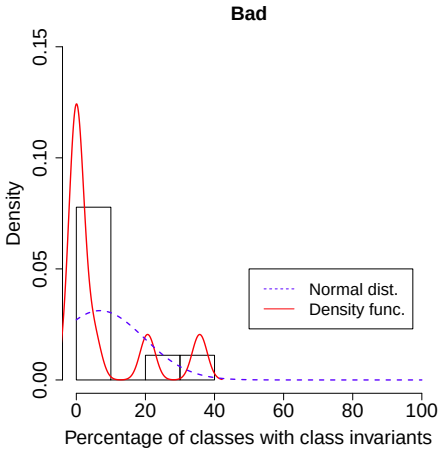
$$W = 0.828984,$$
$$p = 0.04349772$$

(e) Shapiro-Wilk normality test for bad requirements documents.

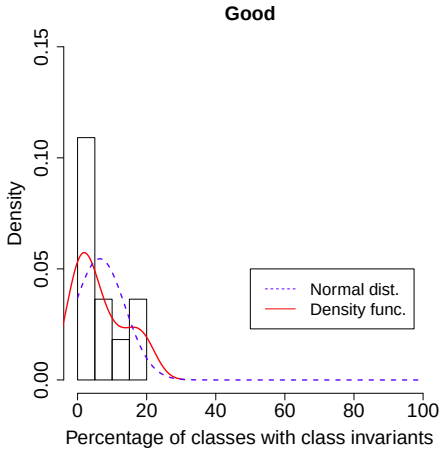
$$W = 0.5577011,$$
$$p = 7.631674\text{e-}06$$

(f) Shapiro-Wilk normality test for good requirements documents.

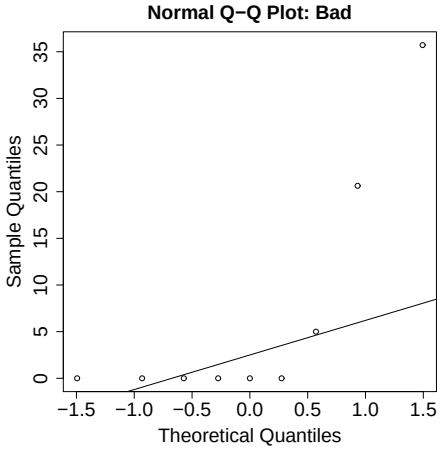
Figure 71: Normality tests for percentage of routines with postconditions of projects with bad requirements documents and projects with good requirements documents.



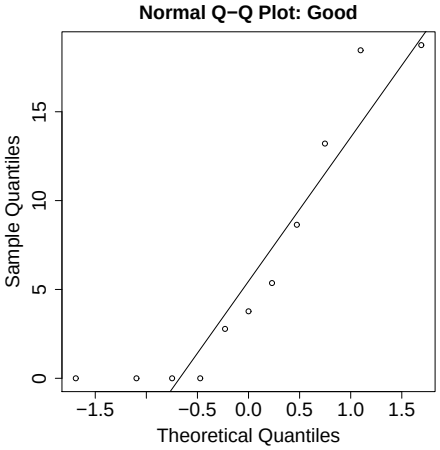
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

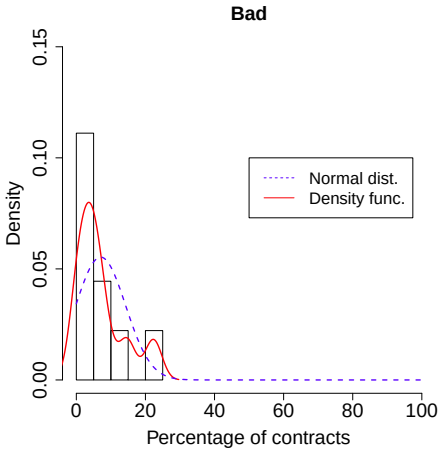
$$W = 0.6299493,$$
$$p = 0.0002148504$$

(e) Shapiro-Wilk normality test for bad requirements documents.

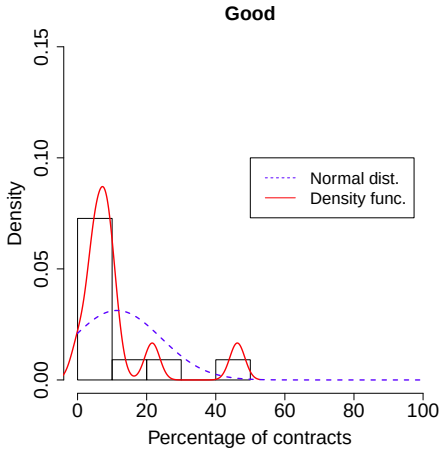
$$W = 0.8272744,$$
$$p = 0.02149736$$

(f) Shapiro-Wilk normality test for good requirements documents.

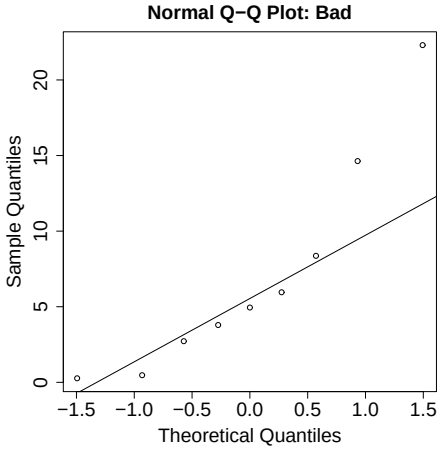
Figure 72: Normality tests for percentage of classes with class invariants of projects with bad requirements documents and projects with good requirements documents.



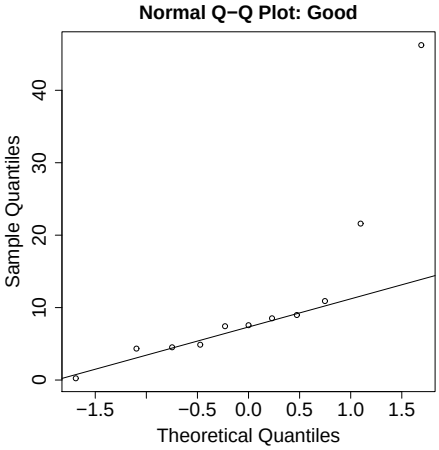
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

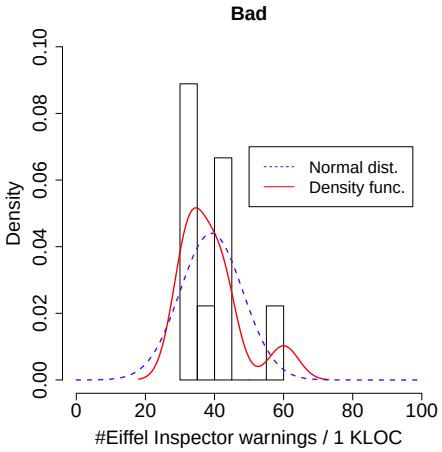
$$W = 0.8573523,$$
$$p = 0.08972123$$

(e) Shapiro-Wilk normality test for bad requirements documents.

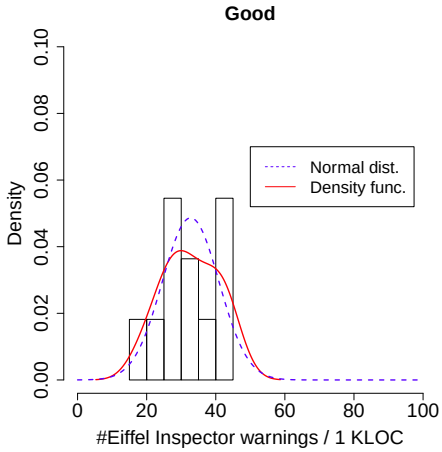
$$W = 0.6935019,$$
$$p = 0.0003859748$$

(f) Shapiro-Wilk normality test for good requirements documents.

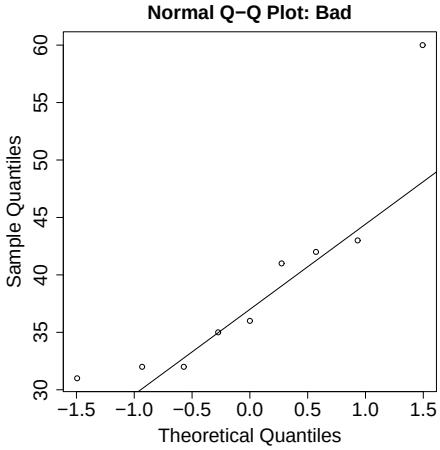
Figure 73: Normality tests for average percentage of contracts of projects with bad requirements documents and projects with good requirements documents.



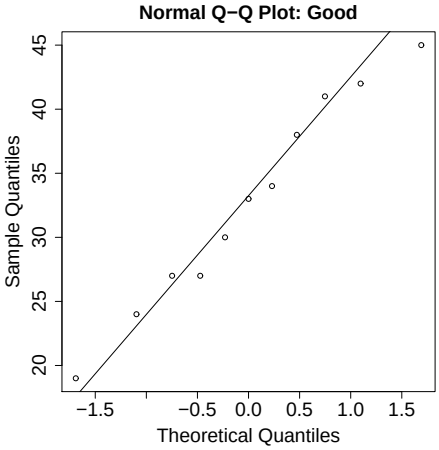
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

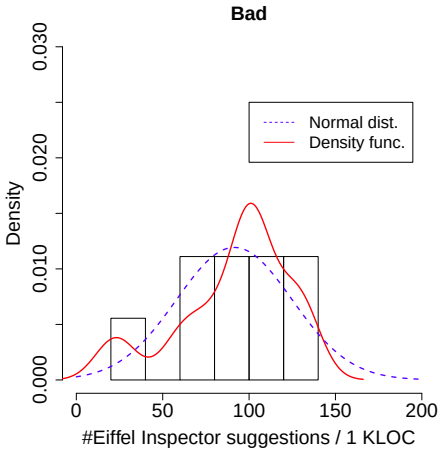
$$W = 0.8187177,$$
$$p = 0.03332726$$

(e) Shapiro-Wilk normality test for bad requirements documents.

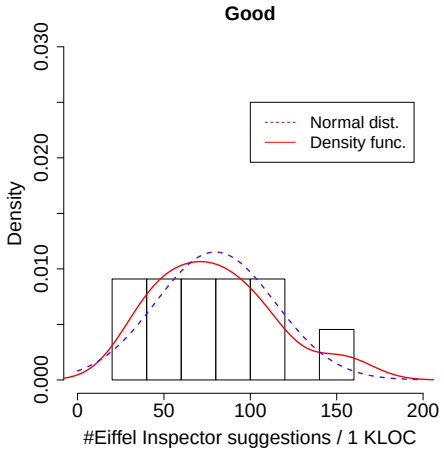
$$W = 0.9709449,$$
$$p = 0.8959037$$

(f) Shapiro-Wilk normality test for good requirements documents.

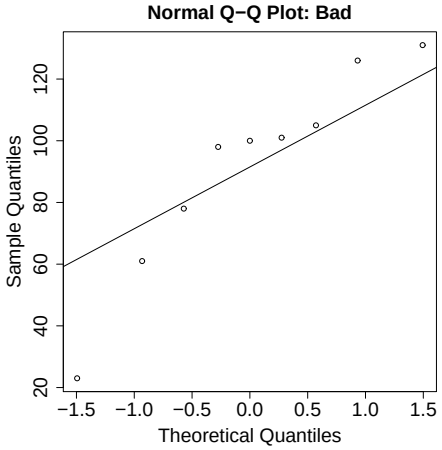
Figure 74: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



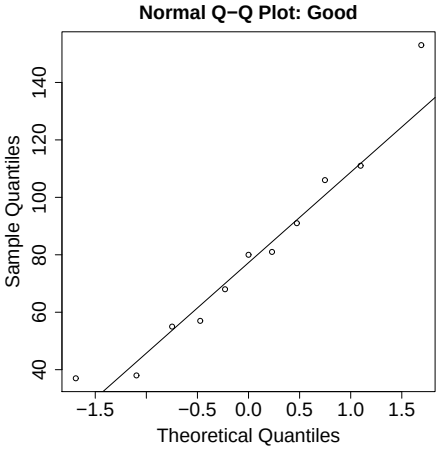
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

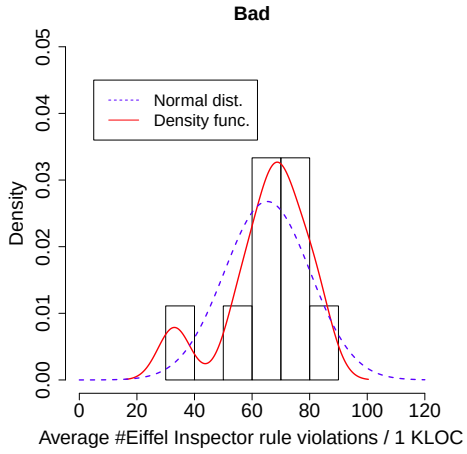
$$W = 0.9116609,$$
$$p = 0.327673$$

(e) Shapiro-Wilk normality test for bad requirements documents.

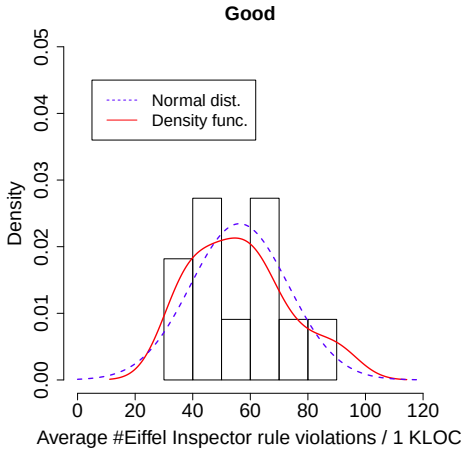
$$W = 0.944758,$$
$$p = 0.5779335$$

(f) Shapiro-Wilk normality test for good requirements documents.

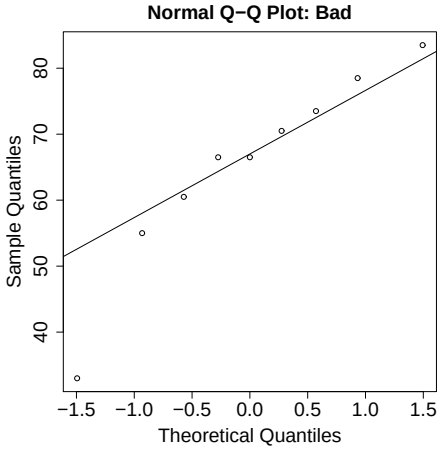
Figure 75: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



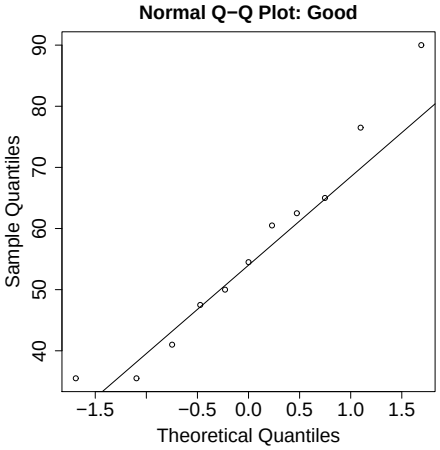
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

$$W = 0.9151683,$$
$$p = 0.3537408$$

(e) Shapiro-Wilk normality test for bad requirements documents.

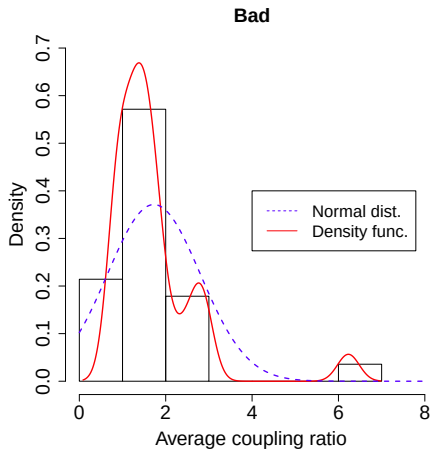
$$W = 0.9490765,$$
$$p = 0.6321742$$

(f) Shapiro-Wilk normality test for good requirements documents.

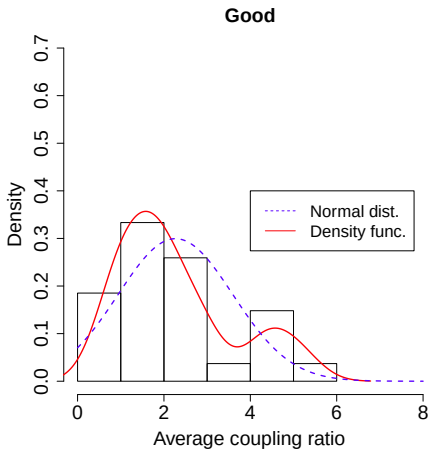
Figure 76: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.

B.1.4 40-20-40 analysis (2009 - 2014)

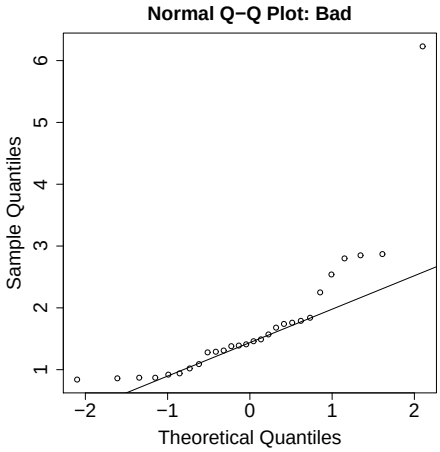
Figure 77 to 86 show the normality tests performed for the 40-20-40 analyis of research question RQ.1 using all projects.



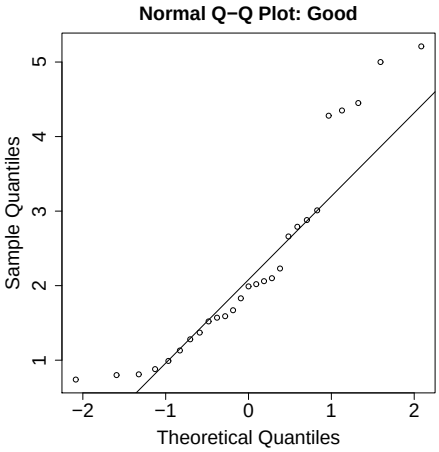
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

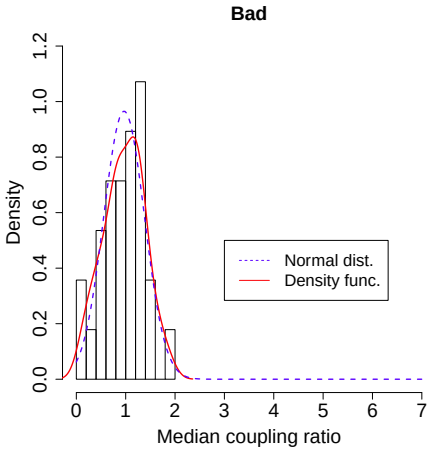
$$W = 0.6932295,$$
$$p = 2.289206\text{e-}06$$

(e) Shapiro-Wilk normality test for bad requirements documents.

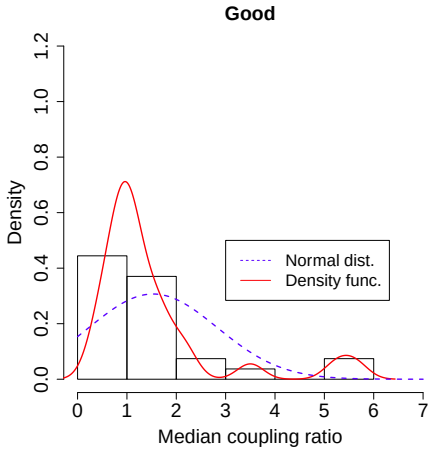
$$W = 0.8800431,$$
$$p = 0.004814094$$

(f) Shapiro-Wilk normality test for good requirements documents.

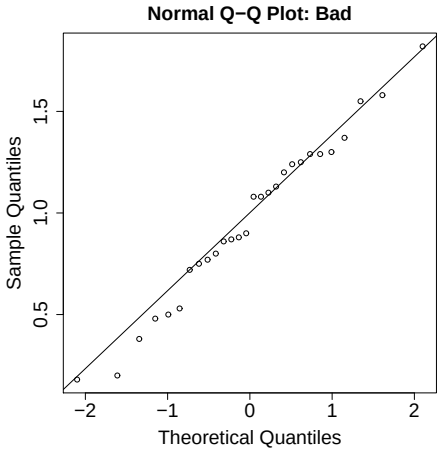
Figure 77: Normality tests for average coupling ratio of projects with bad requirements documents and projects with good requirements documents.



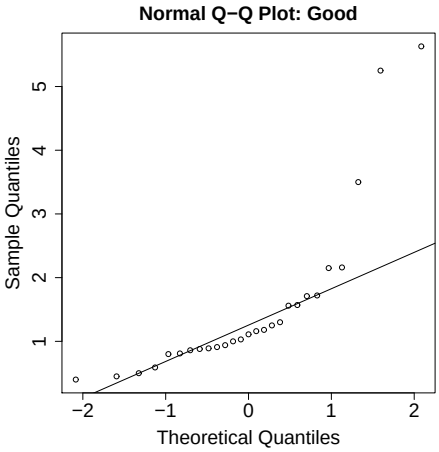
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

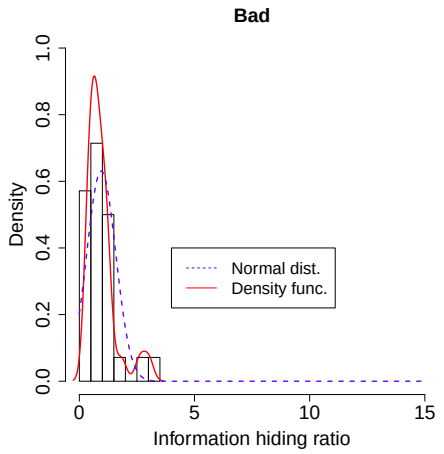
$$W = 0.979809,$$
$$p = 0.8458063$$

(e) Shapiro-Wilk normality test for bad requirements documents.

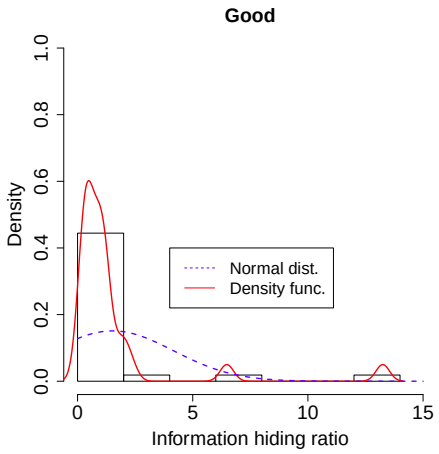
$$W = 0.6959638,$$
$$p = 3.408718\text{e-}06$$

(f) Shapiro-Wilk normality test for good requirements documents.

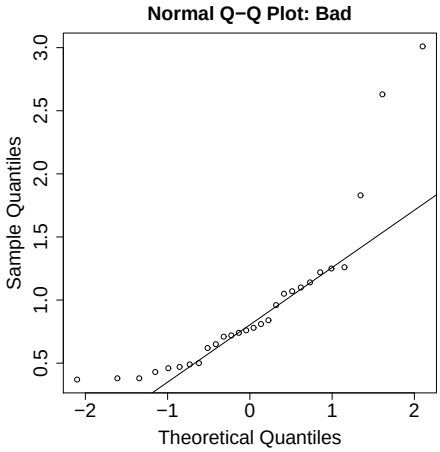
Figure 78: Normality tests for median coupling ratio of projects with bad requirements documents and projects with good requirements documents.



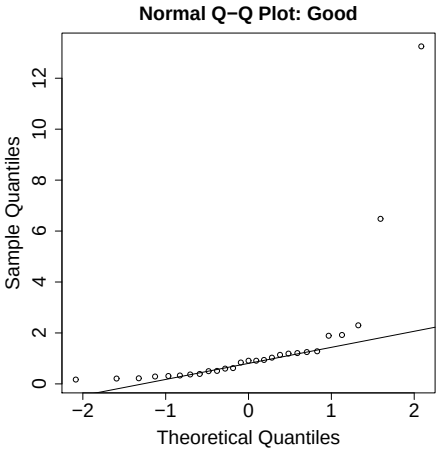
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

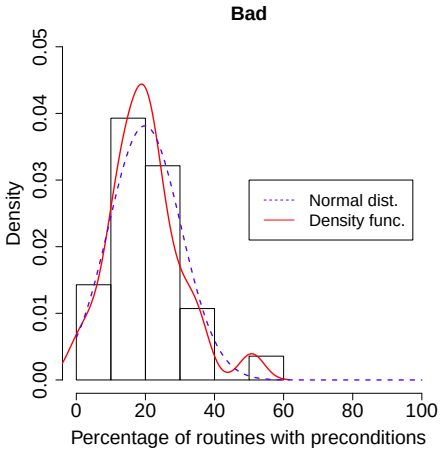
$$W = 0.7702342,$$
$$p = 3.329601\text{e-}05$$

(e) Shapiro-Wilk normality test for bad requirements documents.

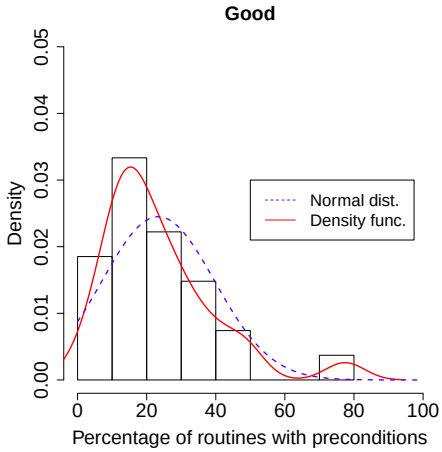
$$W = 0.4711402,$$
$$p = 8.679596\text{e-}09$$

(f) Shapiro-Wilk normality test for good requirements documents.

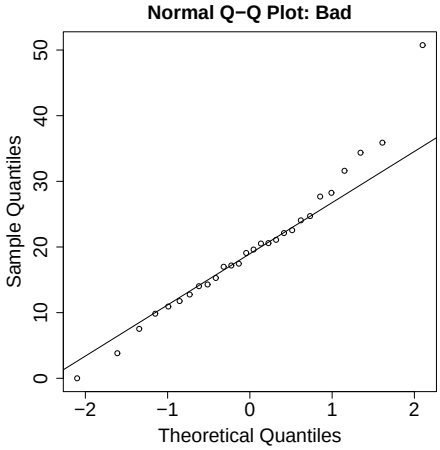
Figure 79: Normality tests for information hiding ratio of projects with bad requirements documents and projects with good requirements documents.



(a) Histogram for bad requirements documents.



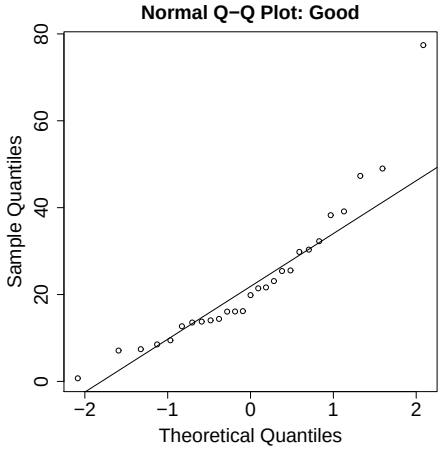
(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.

$W = 0.9615724,$
$p = 0.3796737$

(e) Shapiro-Wilk normality test for bad requirements documents.

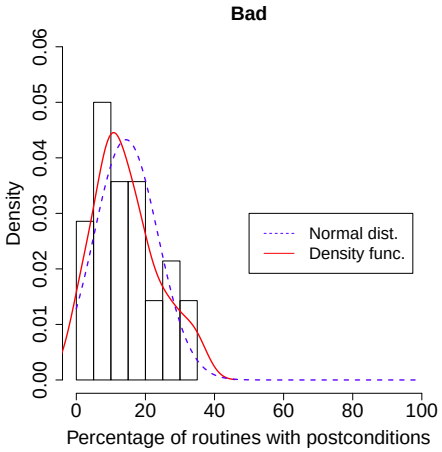


(d) QQ-Plot for good requirements documents.

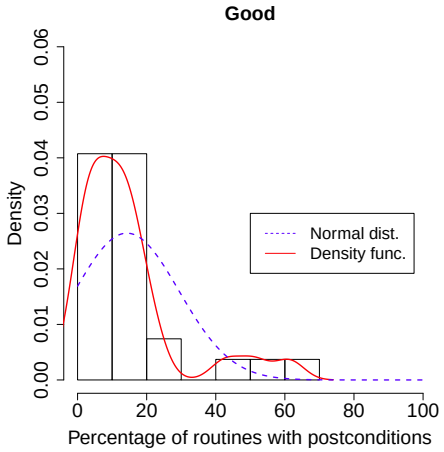
$W = 0.8762933,$
$p = 0.00399925$

(f) Shapiro-Wilk normality test for good requirements documents.

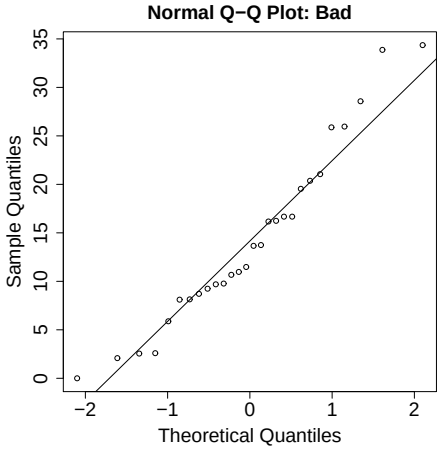
Figure 80: Normality tests for percentage of routines with preconditions of projects with bad requirements documents and projects with good requirements documents.



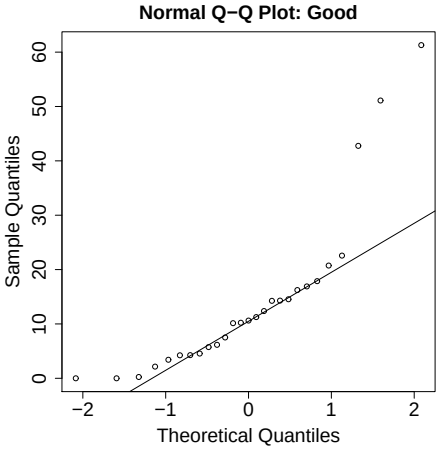
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

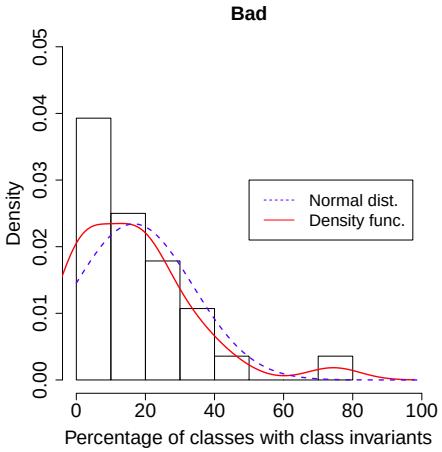
$$W = 0.9501077,$$
$$p = 0.1993673$$

(e) Shapiro-Wilk normality test for bad requirements documents.

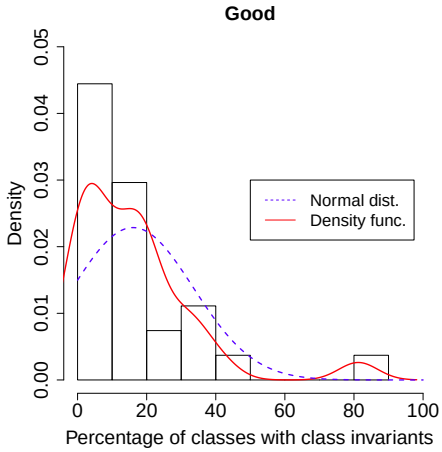
$$W = 0.7707309,$$
$$p = 4.436265\text{e-}05$$

(f) Shapiro-Wilk normality test for good requirements documents.

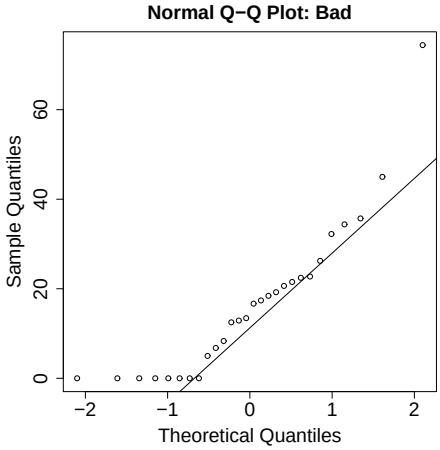
Figure 81: Normality tests for percentage of routines with postconditions of projects with bad requirements documents and projects with good requirements documents.



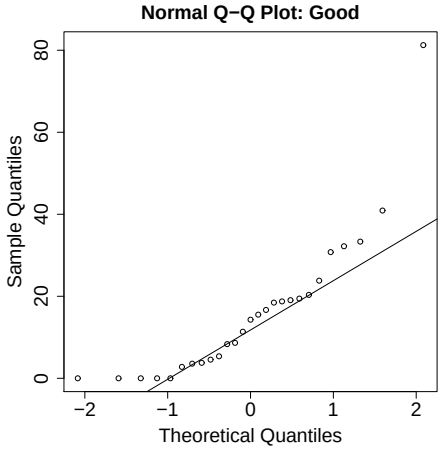
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

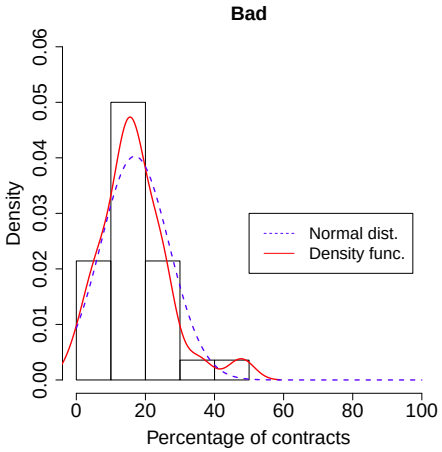
$$W = 0.8499029,$$
$$p = 0.000930599$$

(e) Shapiro-Wilk normality test for bad requirements documents.

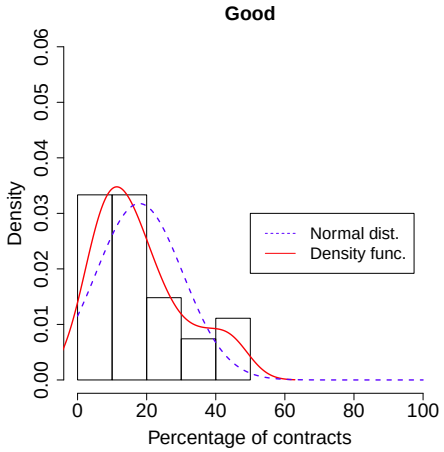
$$W = 0.7945508,$$
$$p = 0.0001102975$$

(f) Shapiro-Wilk normality test for good requirements documents.

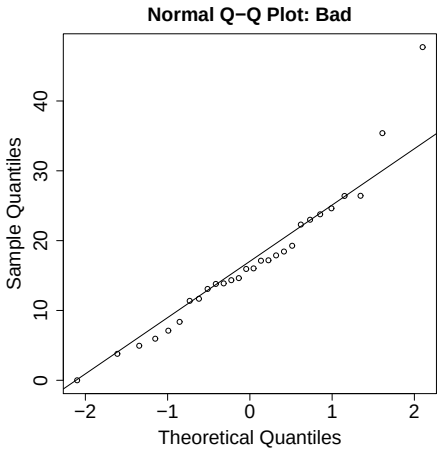
Figure 82: Normality tests for percentage of classes with class invariants of projects with bad requirements documents and projects with good requirements documents.



(a) Histogram for bad requirements documents.



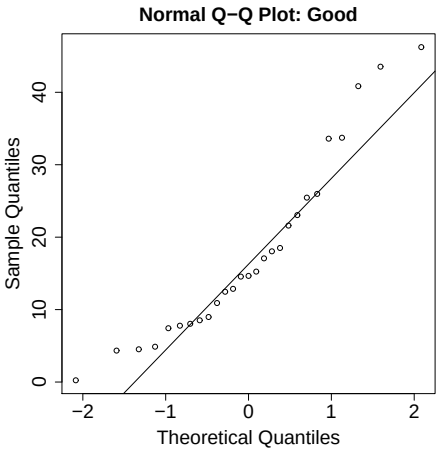
(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.

$$W = 0.9359815,$$
$$p = 0.0873644$$

(e) Shapiro-Wilk normality test for bad requirements documents.

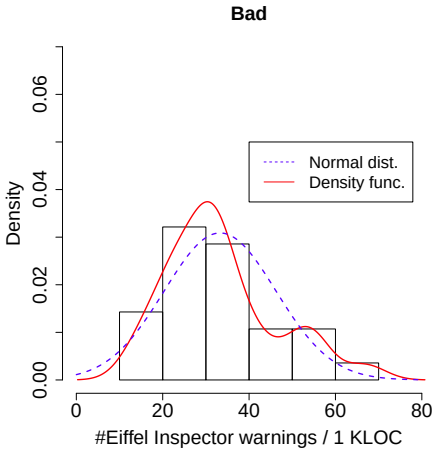


(d) QQ-Plot for good requirements documents.

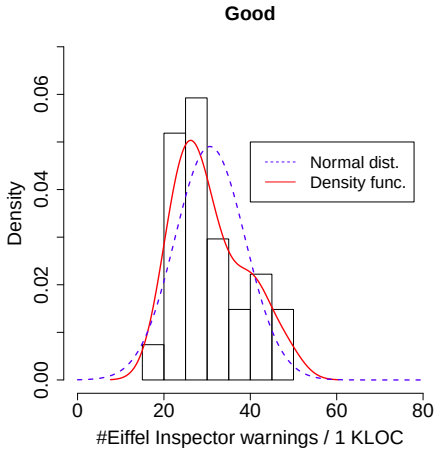
$$W = 0.9139196,$$
$$p = 0.02824557$$

(f) Shapiro-Wilk normality test for good requirements documents.

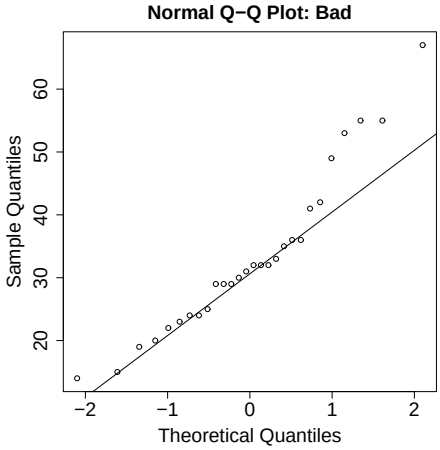
Figure 83: Normality tests for average percentage of contracts of projects with bad requirements documents and projects with good requirements documents.



(a) Histogram for bad requirements documents.



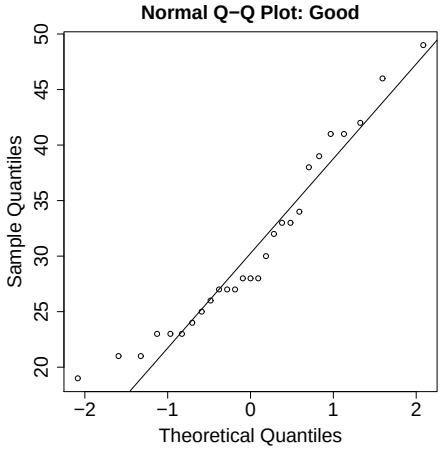
(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.

$$W = 0.9320248,$$
$$p = 0.06935181$$

(e) Shapiro-Wilk normality test for bad requirements documents.

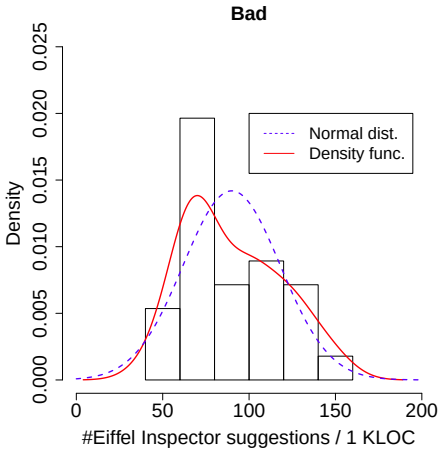


(d) QQ-Plot for good requirements documents.

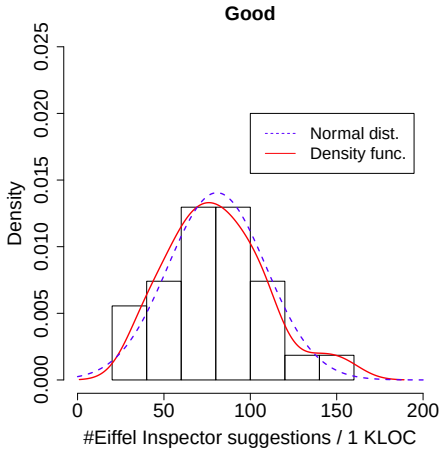
$$W = 0.9334631,$$
$$p = 0.08406678$$

(f) Shapiro-Wilk normality test for good requirements documents.

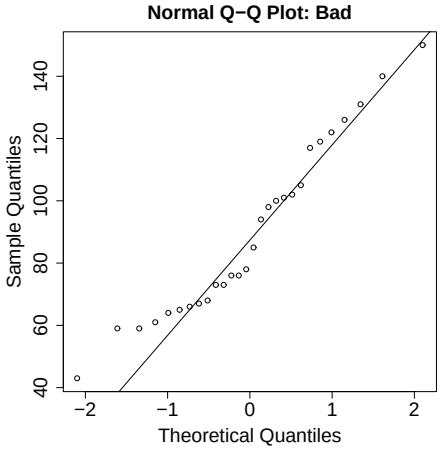
Figure 84: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



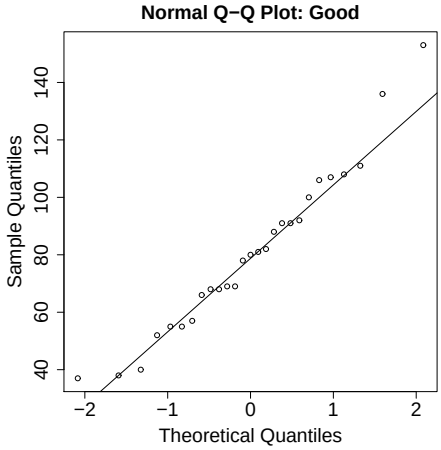
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

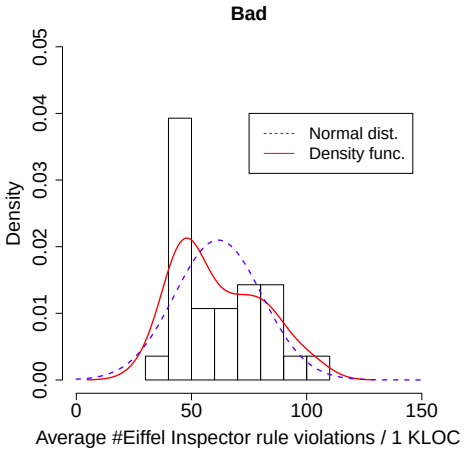
$$W = 0.9457791,$$
$$p = 0.1550107$$

(e) Shapiro-Wilk normality test for bad requirements documents.

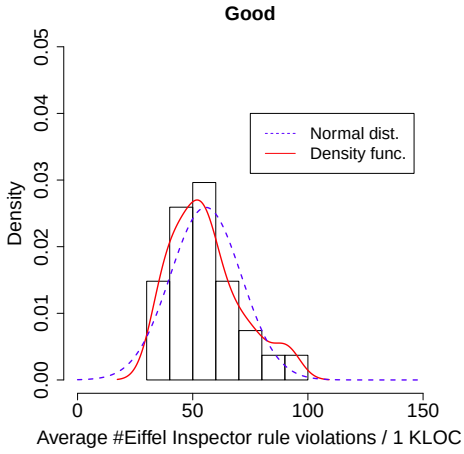
$$W = 0.9620912,$$
$$p = 0.4119213$$

(f) Shapiro-Wilk normality test for good requirements documents.

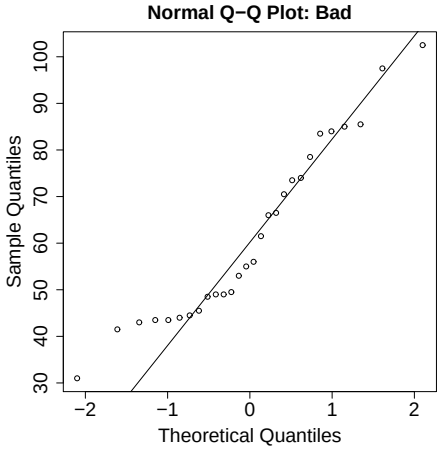
Figure 85: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



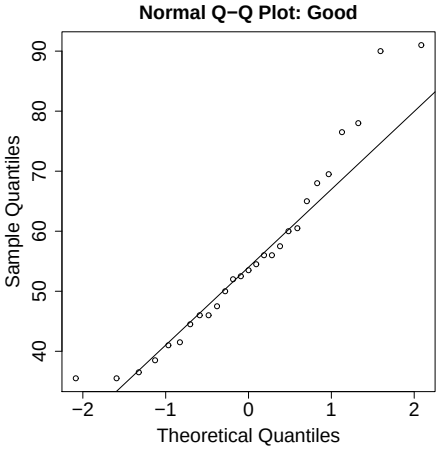
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

$$W = 0.9305303,$$
$$p = 0.06357813$$

(e) Shapiro-Wilk normality test for bad requirements documents.

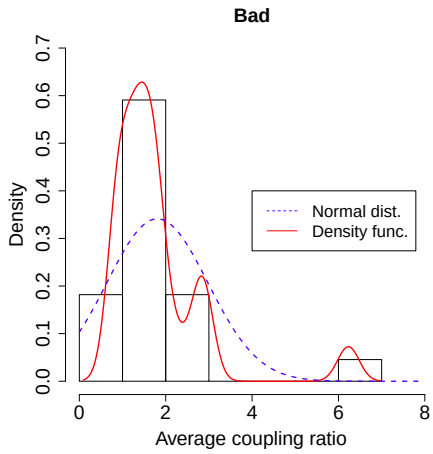
$$W = 0.9317421,$$
$$p = 0.07626031$$

(f) Shapiro-Wilk normality test for good requirements documents.

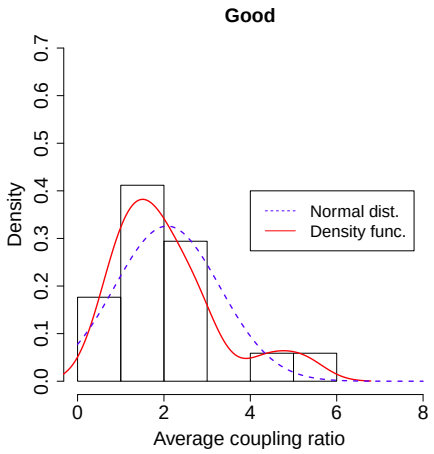
Figure 86: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.

B.1.5 40-20-40 analysis (2009 - 2012)

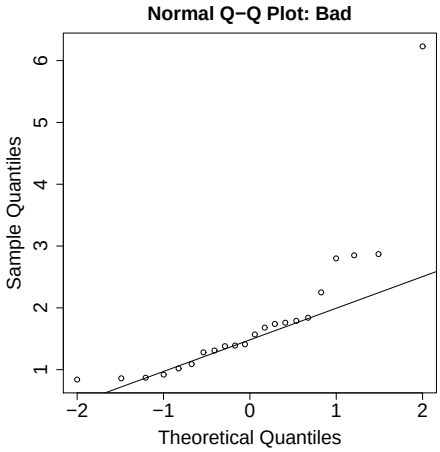
Figure 87 to 96 show the normality tests performed for the 40-20-40 analyis of research question RQ.1 using the projects from 2009 to 2012.



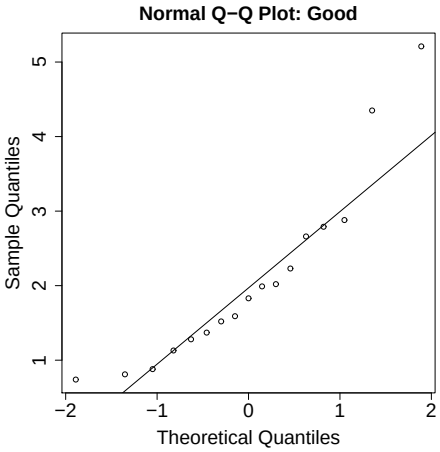
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

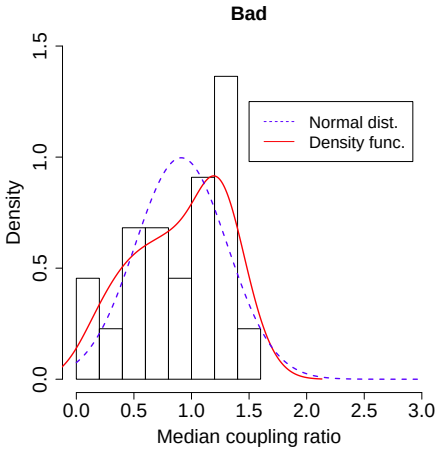
$$W = 0.6981933,$$
$$p = 1.856305\text{e-}05$$

(e) Shapiro-Wilk normality test for bad requirements documents.

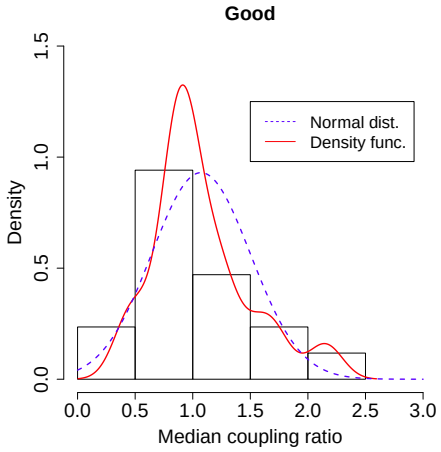
$$W = 0.872579,$$
$$p = 0.02415269$$

(f) Shapiro-Wilk normality test for good requirements documents.

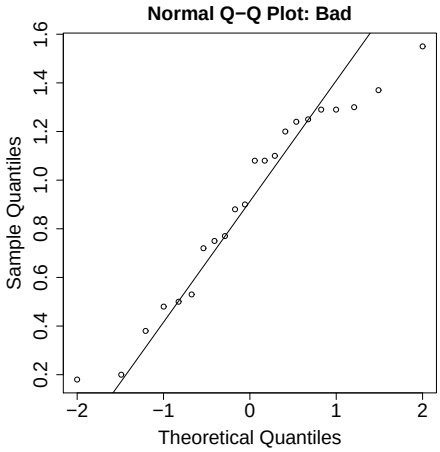
Figure 87: Normality tests for average coupling ratio of projects with bad requirements documents and projects with good requirements documents.



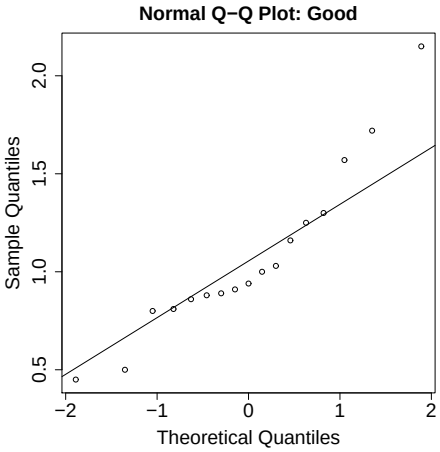
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

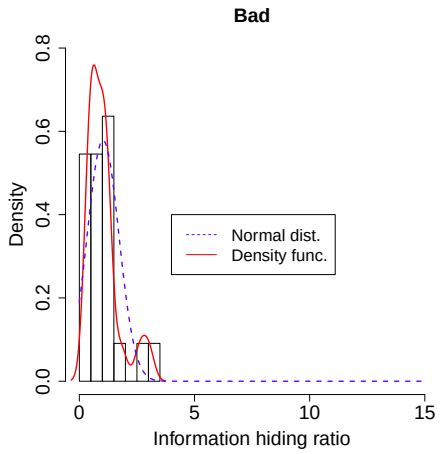
$$W = 0.9414519,$$
$$p = 0.2120383$$

(e) Shapiro-Wilk normality test for bad requirements documents.

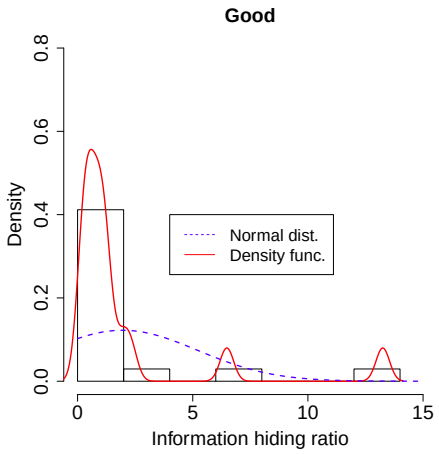
$$W = 0.9145178,$$
$$p = 0.1192721$$

(f) Shapiro-Wilk normality test for good requirements documents.

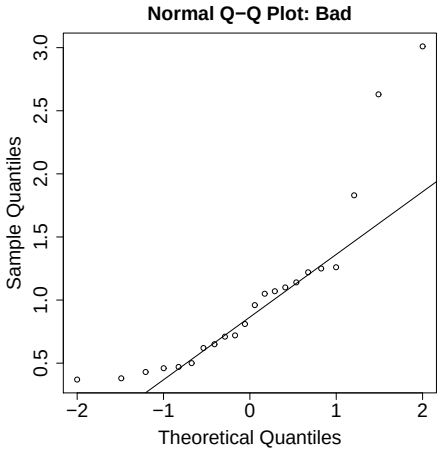
Figure 88: Normality tests for median coupling ratio of projects with bad requirements documents and projects with good requirements documents.



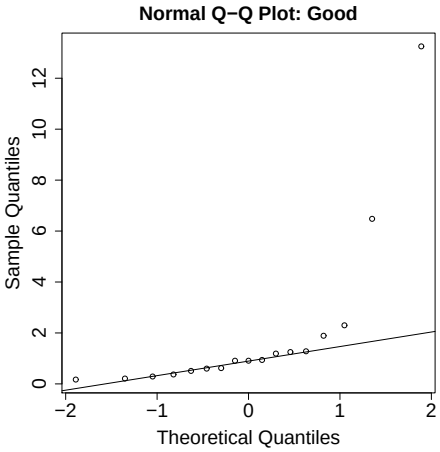
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

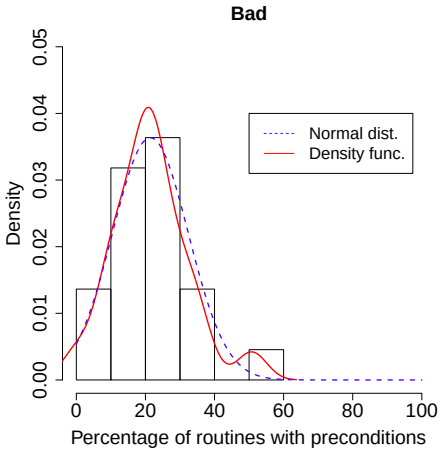
$$W = 0.8031691,$$
$$p = 0.0005611021$$

(e) Shapiro-Wilk normality test for bad requirements documents.

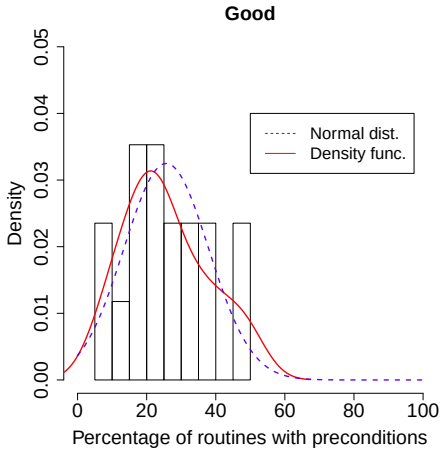
$$W = 0.5412031,$$
$$p = 2.957667\text{e-}06$$

(f) Shapiro-Wilk normality test for good requirements documents.

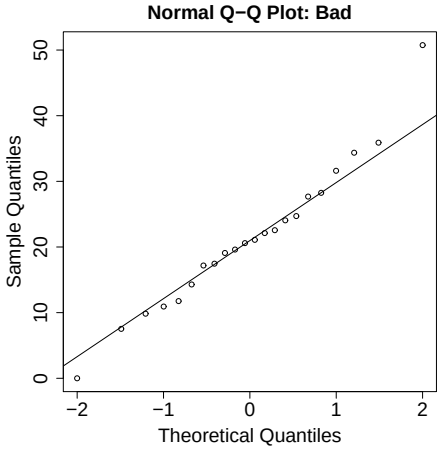
Figure 89: Normality tests for information hiding ratio of projects with bad requirements documents and projects with good requirements documents.



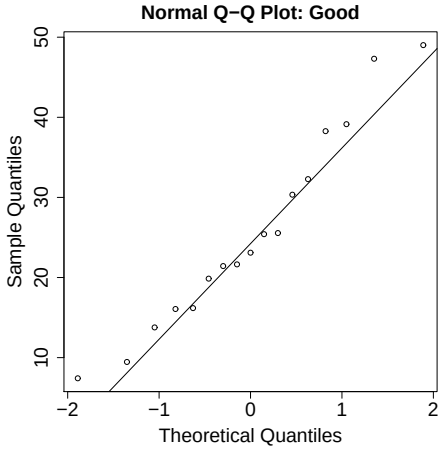
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

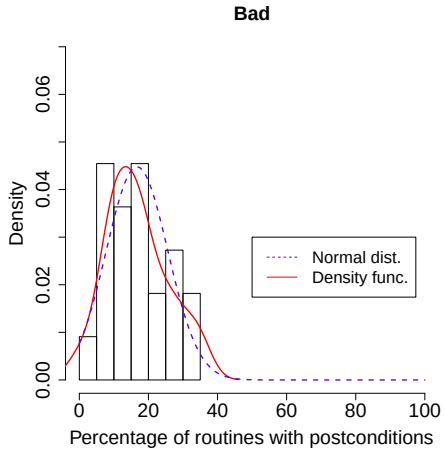
$$W = 0.9710249,$$
$$p = 0.7346519$$

(e) Shapiro-Wilk normality test for bad requirements documents.

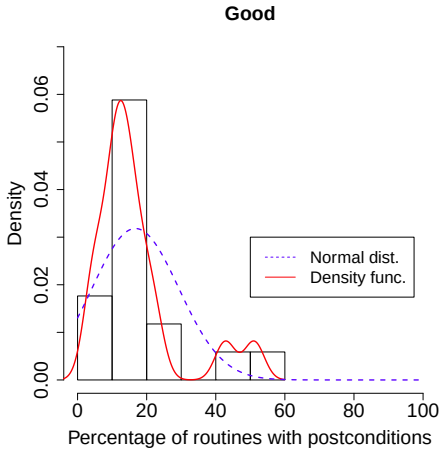
$$W = 0.9530849,$$
$$p = 0.5070968$$

(f) Shapiro-Wilk normality test for good requirements documents.

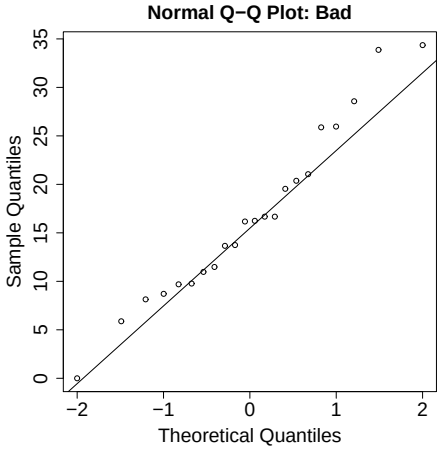
Figure 90: Normality tests for percentage of routines with preconditions of projects with bad requirements documents and projects with good requirements documents.



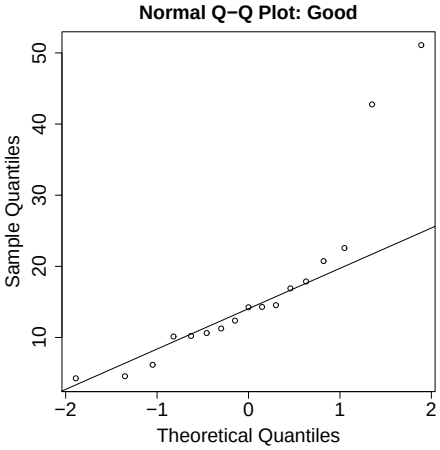
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

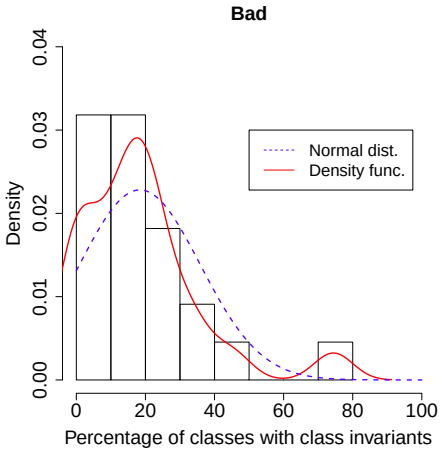
$$W = 0.9647749,$$
$$p = 0.5912975$$

(e) Shapiro-Wilk normality test for bad requirements documents.

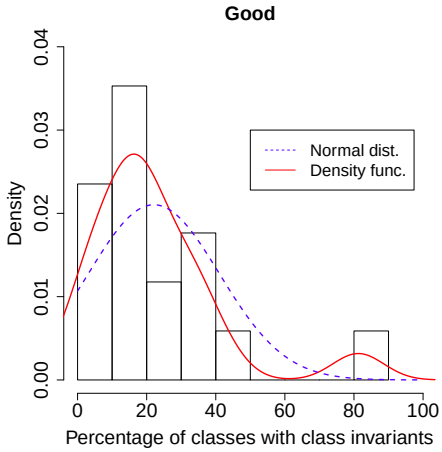
$$W = 0.7777443,$$
$$p = 0.001020918$$

(f) Shapiro-Wilk normality test for good requirements documents.

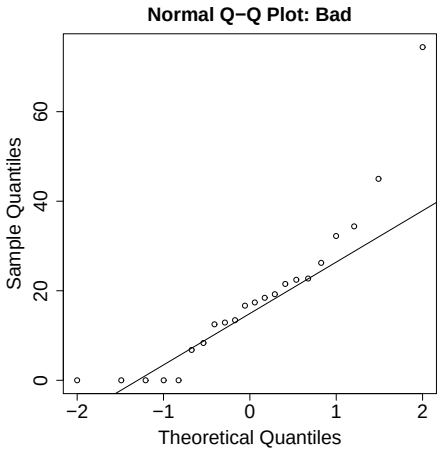
Figure 91: Normality tests for percentage of routines with postconditions of projects with bad requirements documents and projects with good requirements documents.



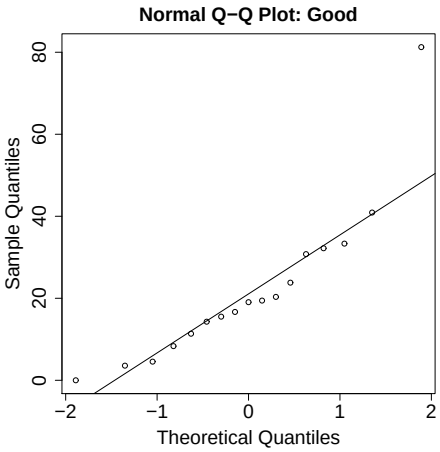
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

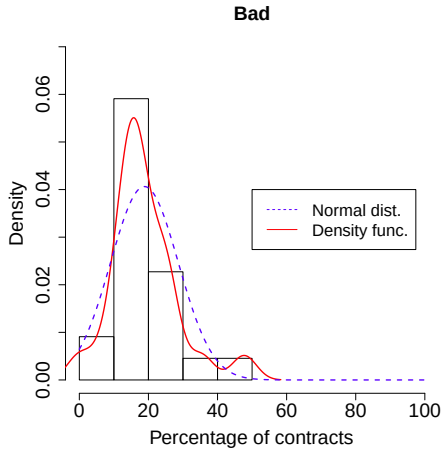
$$W = 0.8528686,$$
$$p = 0.003816916$$

(e) Shapiro-Wilk normality test for bad requirements documents.

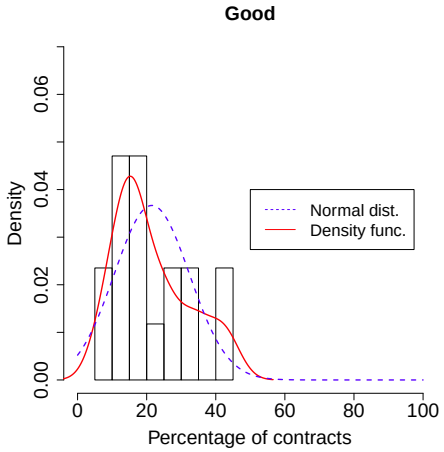
$$W = 0.8294444,$$
$$p = 0.005292713$$

(f) Shapiro-Wilk normality test for good requirements documents.

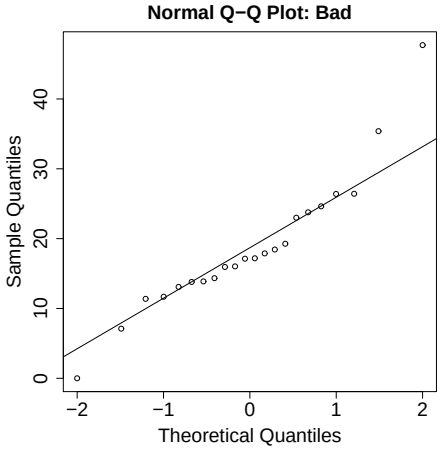
Figure 92: Normality tests for percentage of classes with class invariants of projects with bad requirements documents and projects with good requirements documents.



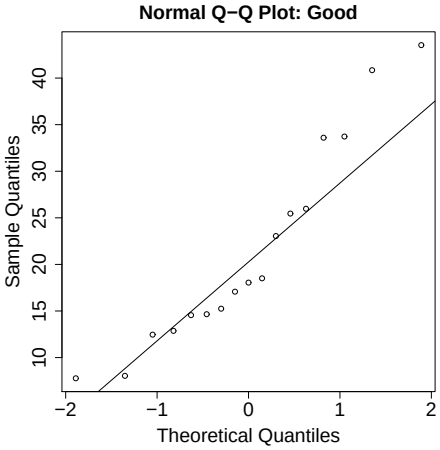
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

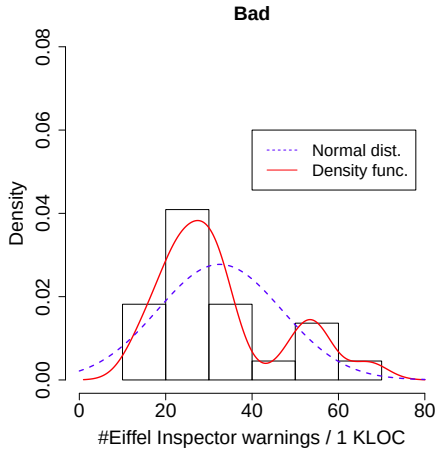
$$W = 0.9154783,$$
$$p = 0.06131113$$

(e) Shapiro-Wilk normality test for bad requirements documents.

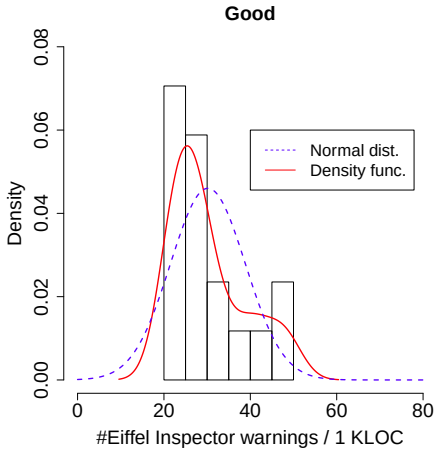
$$W = 0.9127893,$$
$$p = 0.1115068$$

(f) Shapiro-Wilk normality test for good requirements documents.

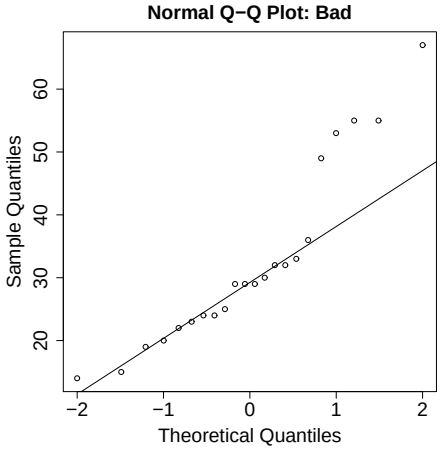
Figure 93: Normality tests for average percentage of contracts of projects with bad requirements documents and projects with good requirements documents.



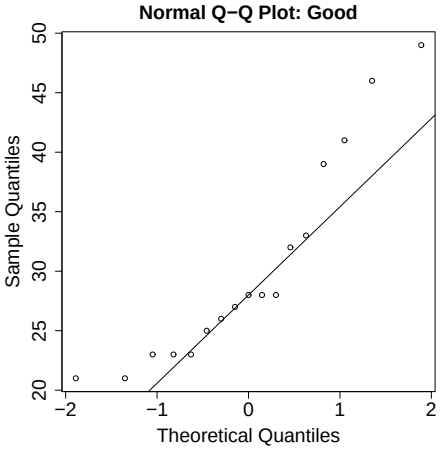
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

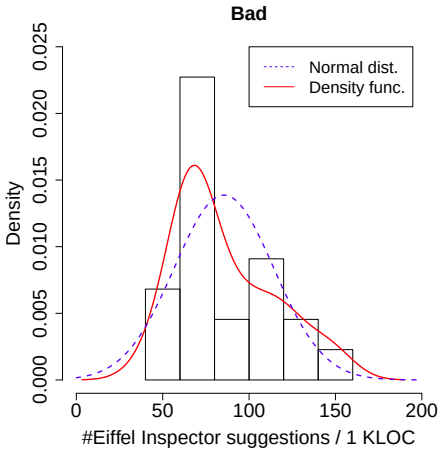
$W = 0.8879138$,
$p = 0.01716403$

(e) Shapiro-Wilk normality test for bad requirements documents.

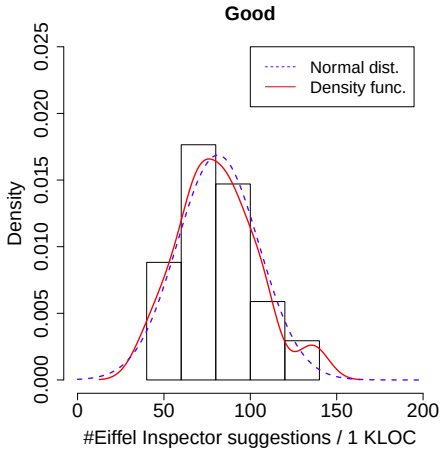
$W = 0.8689709$,
$p = 0.02116025$

(f) Shapiro-Wilk normality test for good requirements documents.

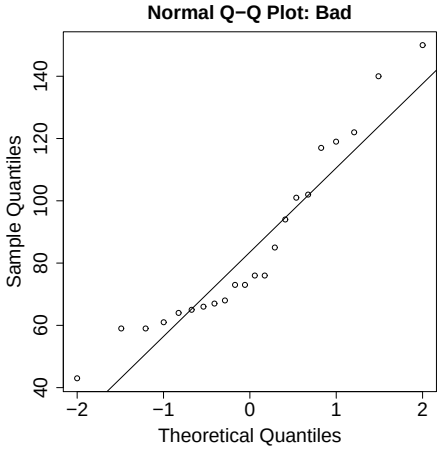
Figure 94: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



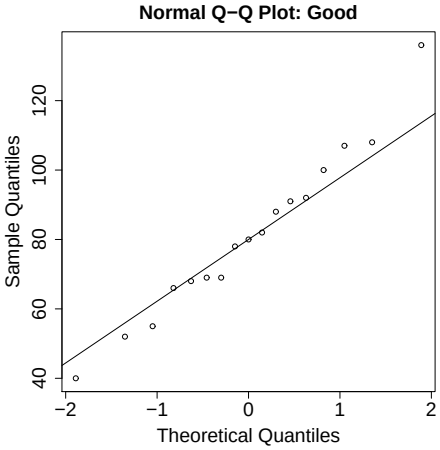
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

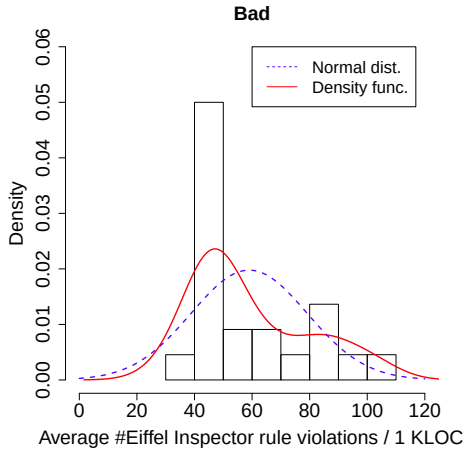
$W = 0.9063254,$
$p = 0.03985724$

(e) Shapiro-Wilk normality test for bad requirements documents.

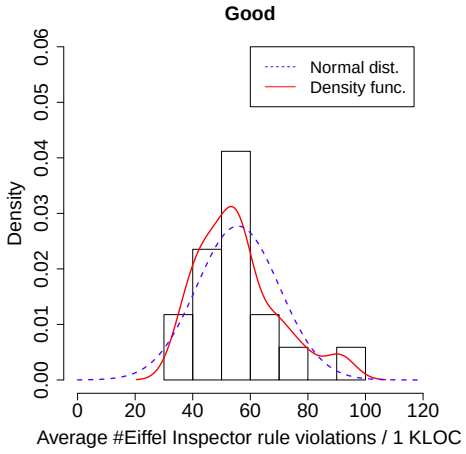
$W = 0.978732,$
$p = 0.9446916$

(f) Shapiro-Wilk normality test for good requirements documents.

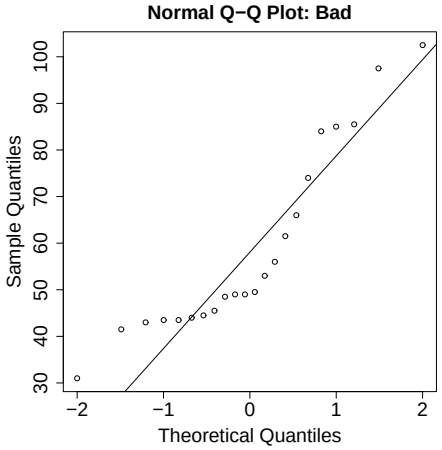
Figure 95: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



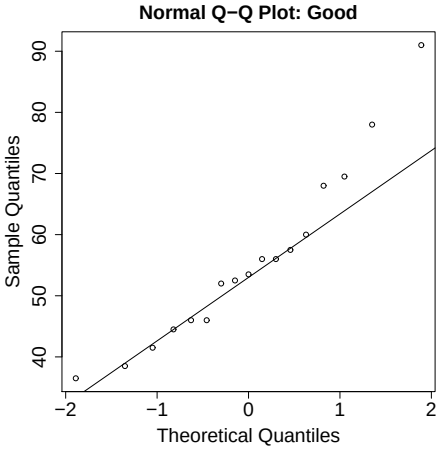
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

$$W = 0.8695372,$$
$$p = 0.007674386$$

(e) Shapiro-Wilk normality test for bad requirements documents.

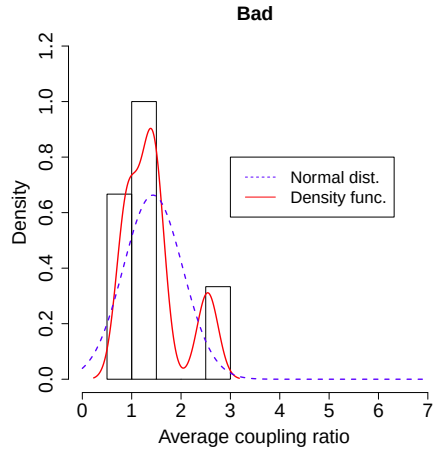
$$W = 0.9327271,$$
$$p = 0.2416721$$

(f) Shapiro-Wilk normality test for good requirements documents.

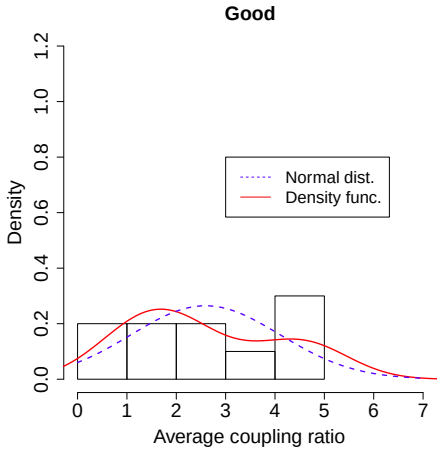
Figure 96: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.

B.1.6 40-20-40 analysis (2013 - 2014)

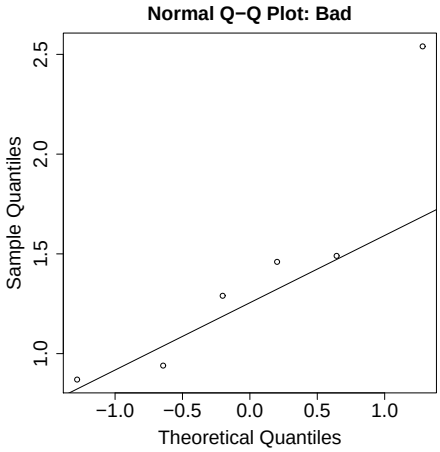
Figure 97 to 106 show the normality tests performed for the 40-20-40 analyis of research question RQ.1 using the projects from 2013 and 2014.



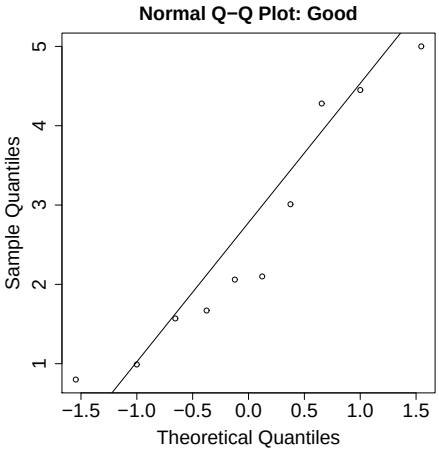
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

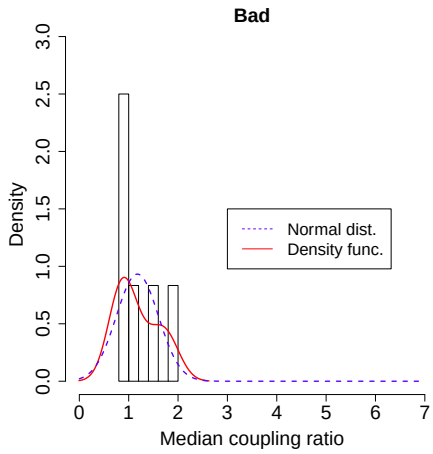
$$W = 0.8538885,$$
$$p = 0.1691641$$

(e) Shapiro-Wilk normality test for bad requirements documents.

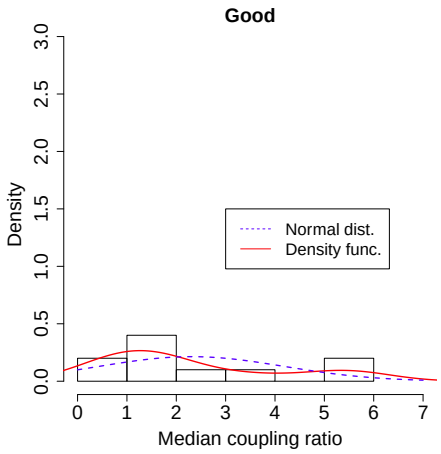
$$W = 0.9010294,$$
$$p = 0.2248642$$

(f) Shapiro-Wilk normality test for good requirements documents.

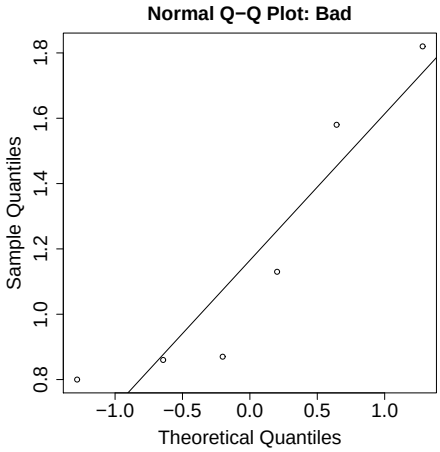
Figure 97: Normality tests for average coupling ratio of projects with bad requirements documents and projects with good requirements documents.



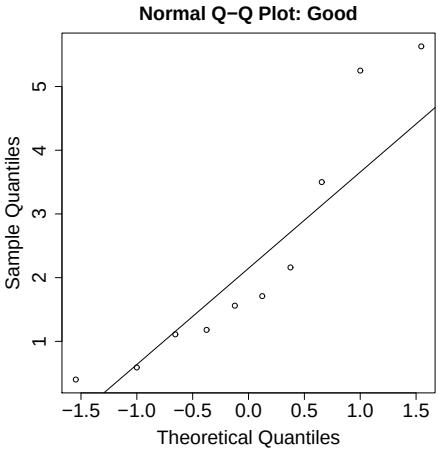
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

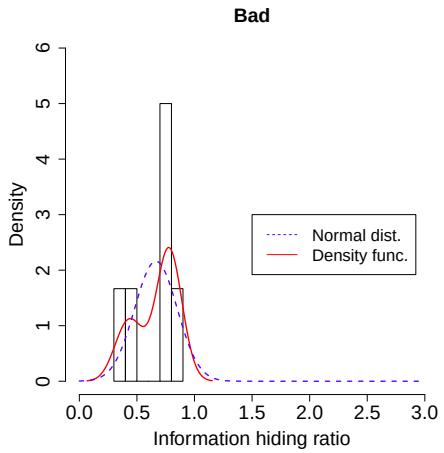
$$W = 0.848112,$$
$$p = 0.1519561$$

(e) Shapiro-Wilk normality test for bad requirements documents.

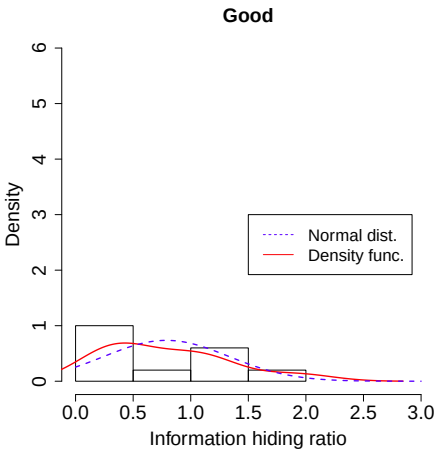
$$W = 0.8532994,$$
$$p = 0.06357717$$

(f) Shapiro-Wilk normality test for good requirements documents.

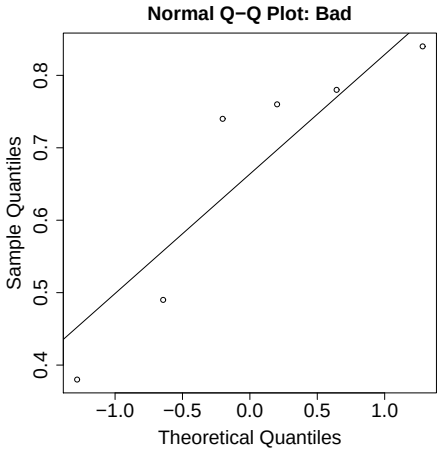
Figure 98: Normality tests for median coupling ratio of projects with bad requirements documents and projects with good requirements documents.



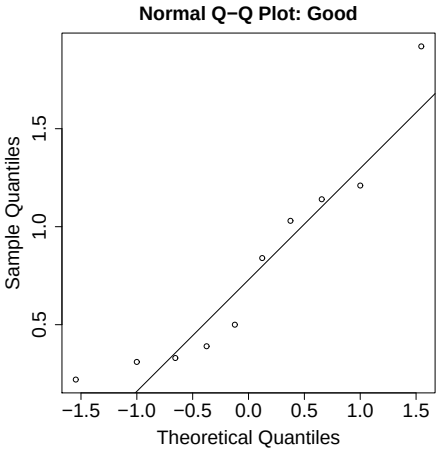
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

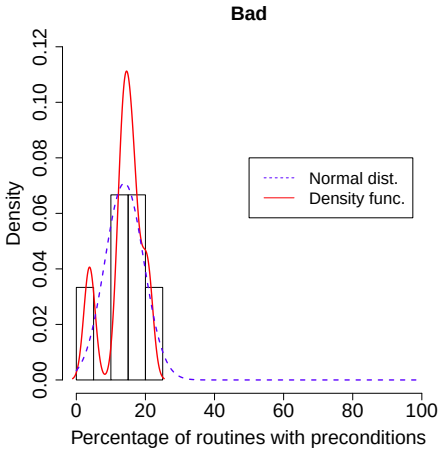
$$W = 0.8429372,$$
$$p = 0.1378506$$

(e) Shapiro-Wilk normality test for bad requirements documents.

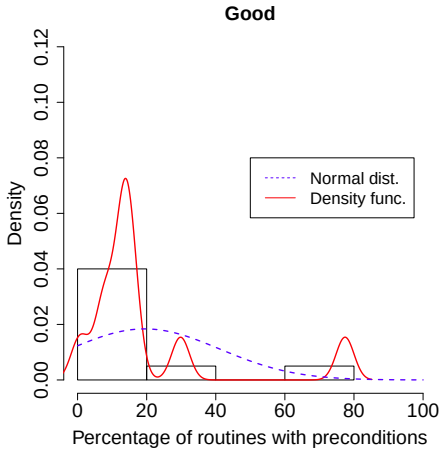
$$W = 0.892771,$$
$$p = 0.1821696$$

(f) Shapiro-Wilk normality test for good requirements documents.

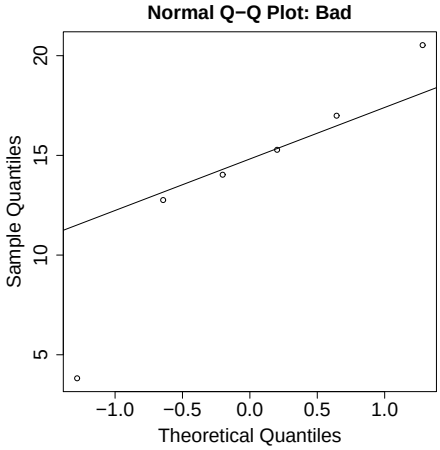
Figure 99: Normality tests for information hiding ratio of projects with bad requirements documents and projects with good requirements documents.



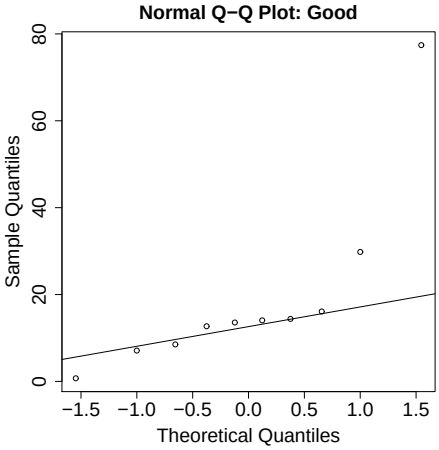
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

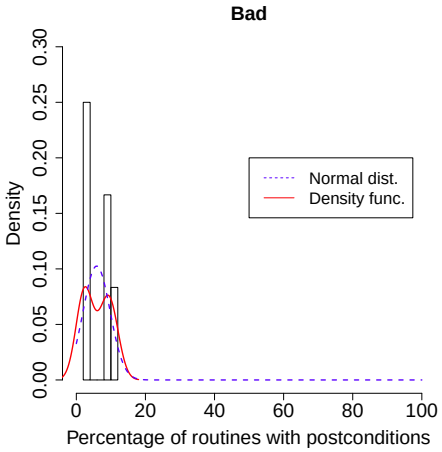
$$W = 0.9158237,$$
$$p = 0.4758019$$

(e) Shapiro-Wilk normality test for bad requirements documents.

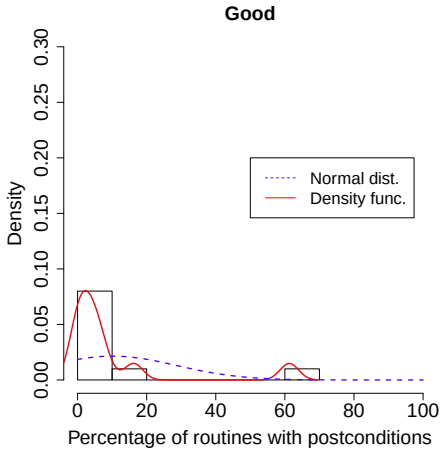
$$W = 0.6712135,$$
$$p = 0.0003952468$$

(f) Shapiro-Wilk normality test for good requirements documents.

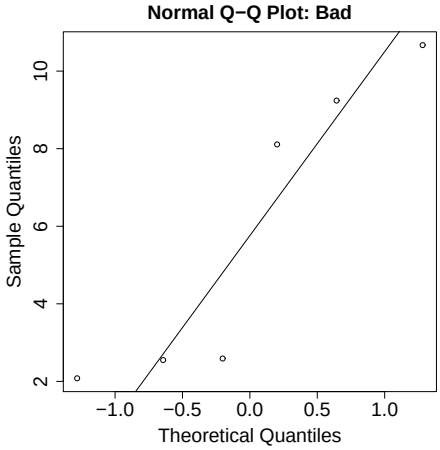
Figure 100: Normality tests for percentage of routines with preconditions of projects with bad requirements documents and projects with good requirements documents.



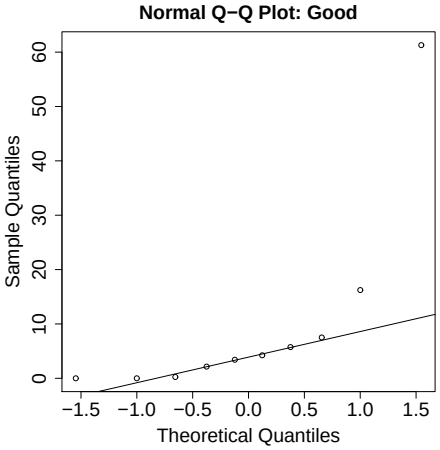
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

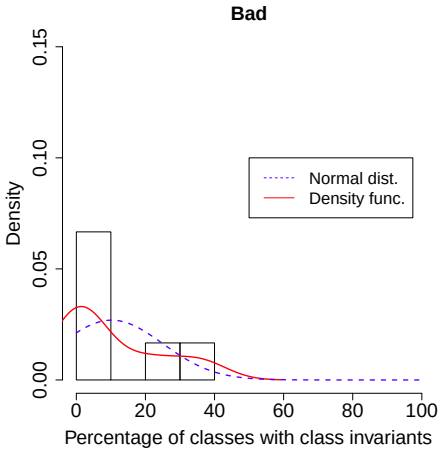
$$W = 0.8235169,$$
$$p = 0.09463502$$

(e) Shapiro-Wilk normality test for bad requirements documents.

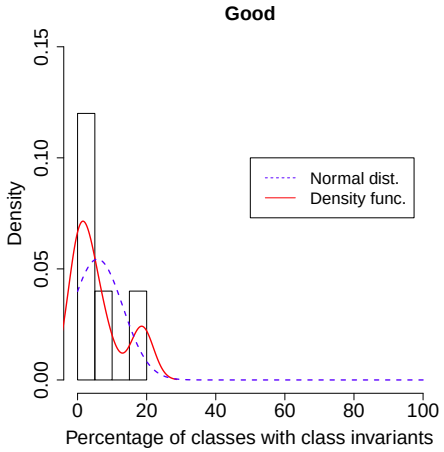
$$W = 0.5784927,$$
$$p = 3.067987\text{e-}05$$

(f) Shapiro-Wilk normality test for good requirements documents.

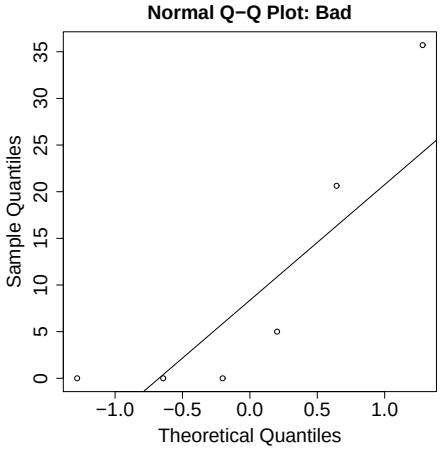
Figure 101: Normality tests for percentage of routines with postconditions of projects with bad requirements documents and projects with good requirements documents.



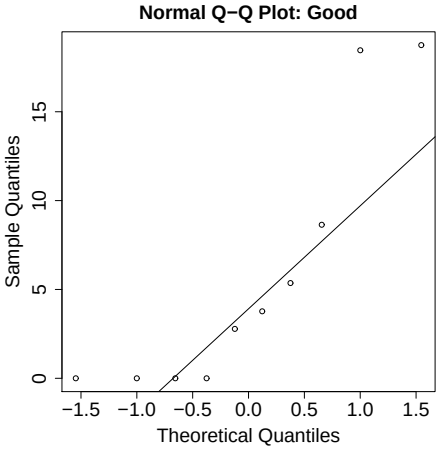
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

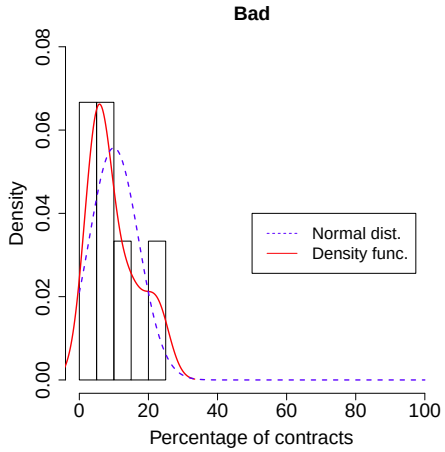
$$W = 0.7756639,$$
$$p = 0.03510825$$

(e) Shapiro-Wilk normality test for bad requirements documents.

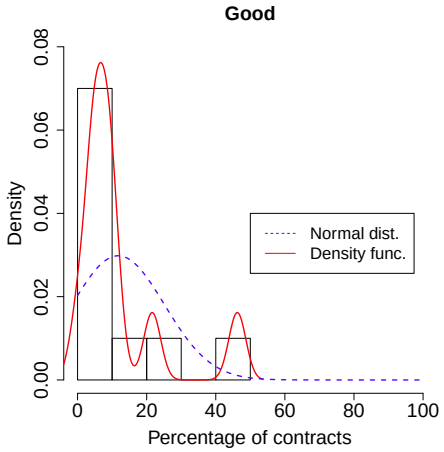
$$W = 0.775929,$$
$$p = 0.007385413$$

(f) Shapiro-Wilk normality test for good requirements documents.

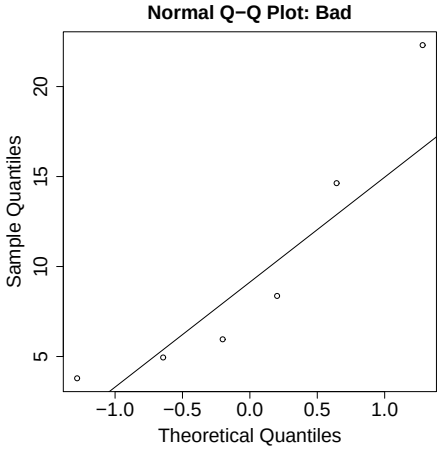
Figure 102: Normality tests for percentage of classes with class invariants of projects with bad requirements documents and projects with good requirements documents.



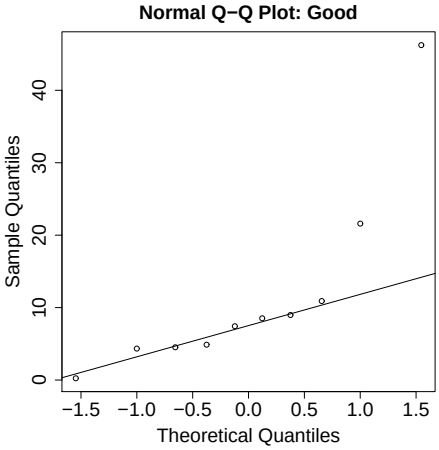
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

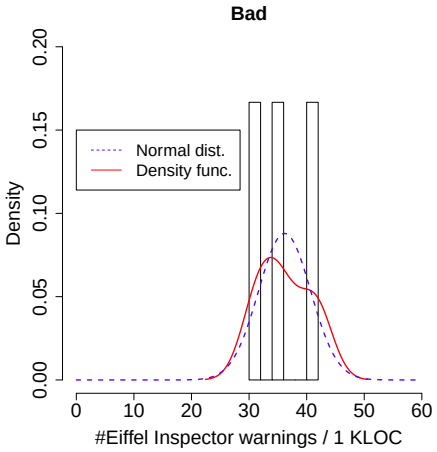
$$W = 0.8598693,$$
$$p = 0.1887182$$

(e) Shapiro-Wilk normality test for bad requirements documents.

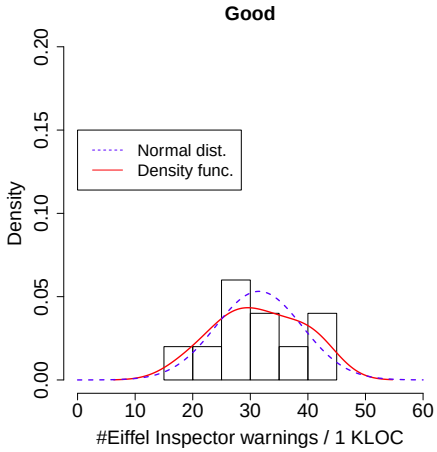
$$W = 0.7169622,$$
$$p = 0.00141477$$

(f) Shapiro-Wilk normality test for good requirements documents.

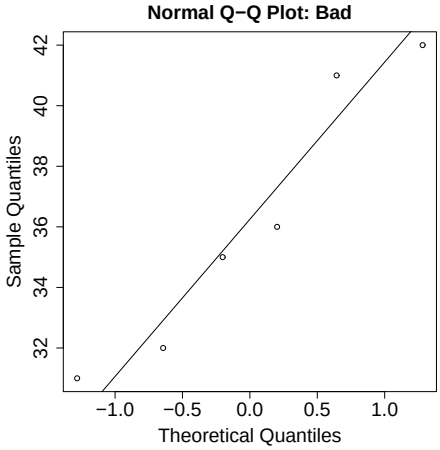
Figure 103: Normality tests for average percentage of contracts of projects with bad requirements documents and projects with good requirements documents.



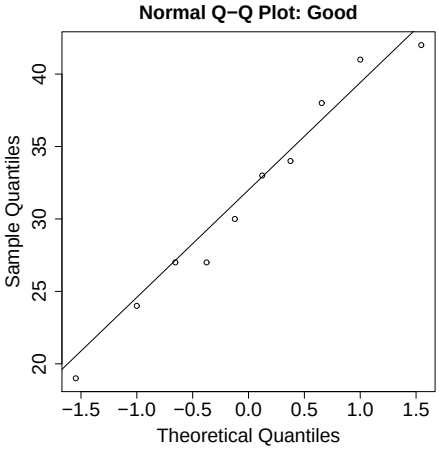
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

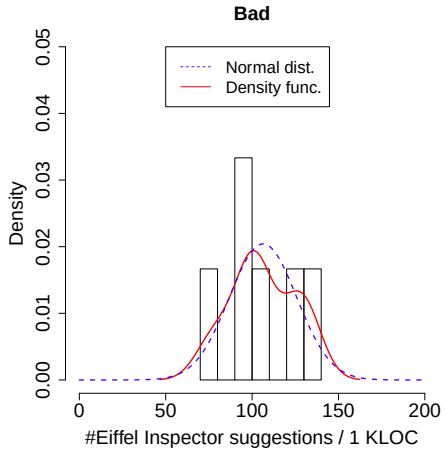
$$W = 0.9125905,$$
$$p = 0.4536573$$

(e) Shapiro-Wilk normality test for bad requirements documents.

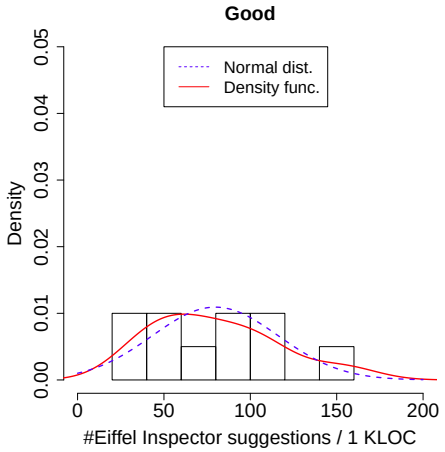
$$W = 0.9665168,$$
$$p = 0.8568068$$

(f) Shapiro-Wilk normality test for good requirements documents.

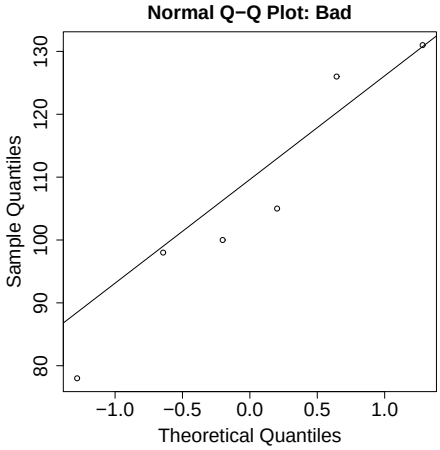
Figure 104: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



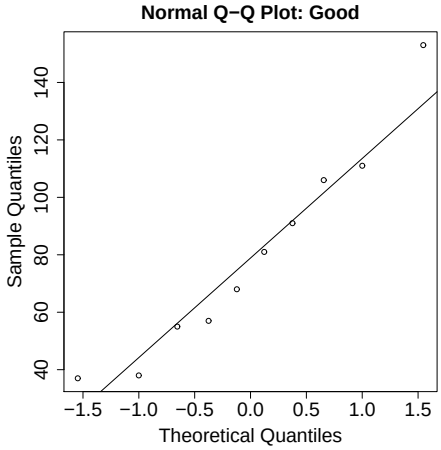
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

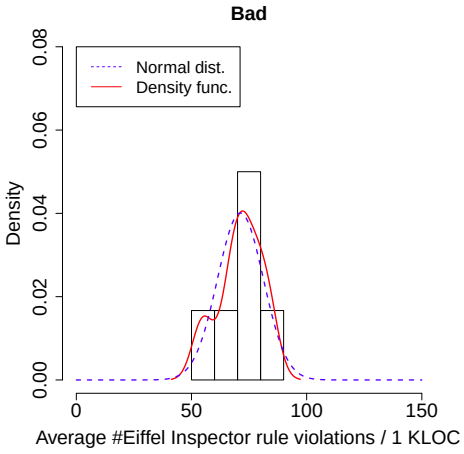
$$W = 0.9406178,$$
$$p = 0.6641809$$

(e) Shapiro-Wilk normality test for bad requirements documents.

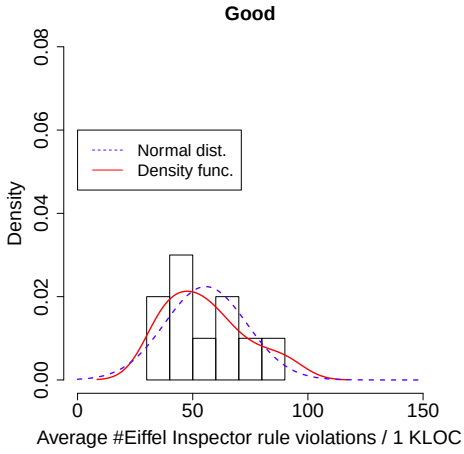
$$W = 0.9399047,$$
$$p = 0.5519533$$

(f) Shapiro-Wilk normality test for good requirements documents.

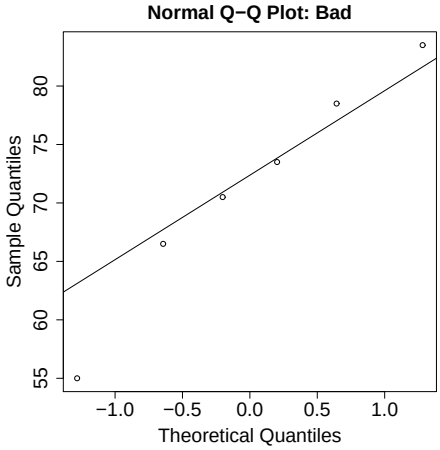
Figure 105: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.



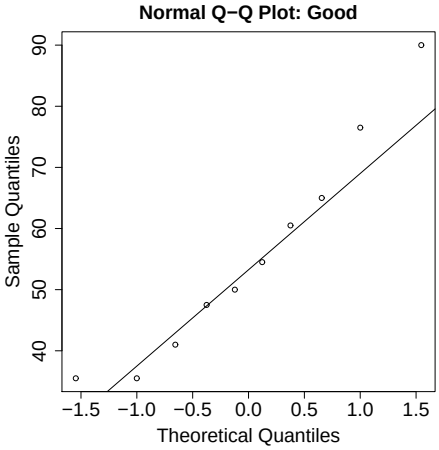
(a) Histogram for bad requirements documents.



(b) Histogram for good requirements documents.



(c) QQ-Plot for bad requirements documents.



(d) QQ-Plot for good requirements documents.

$$W = 0.974293,$$
$$p = 0.9199305$$

(e) Shapiro-Wilk normality test for bad requirements documents.

$$W = 0.9363715,$$
$$p = 0.5133885$$

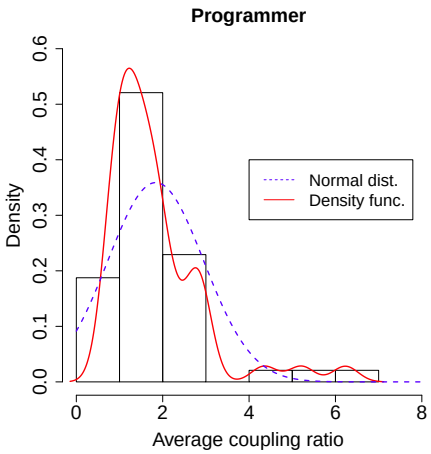
(f) Shapiro-Wilk normality test for good requirements documents.

Figure 106: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with bad requirements documents and projects with good requirements documents.

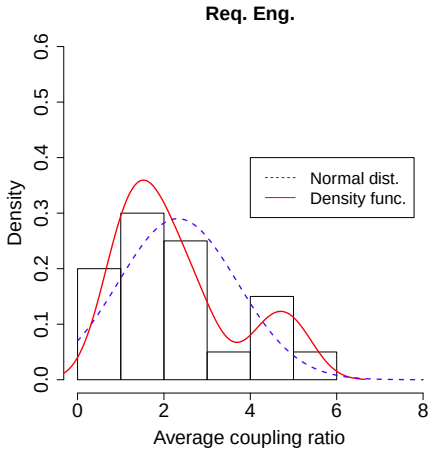
B.2 Research question RQ.2

B.2.1 Main analysis (2009 - 2012 vs. 2013 - 2014)

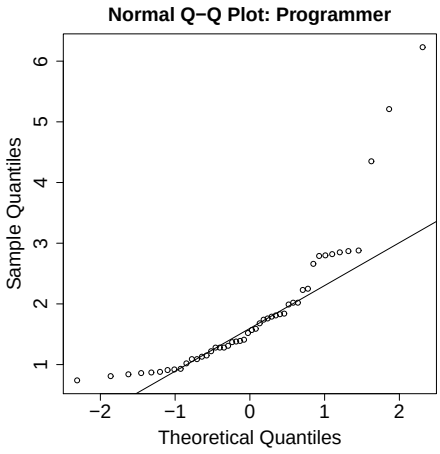
Figure 107 to 116 show the normality tests performed for the main analyis of research question RQ.2 comparing projects from 2009 to 2012 with projects from 2013 and 2014.



(a) Histogram for programmer-written requirements documents.



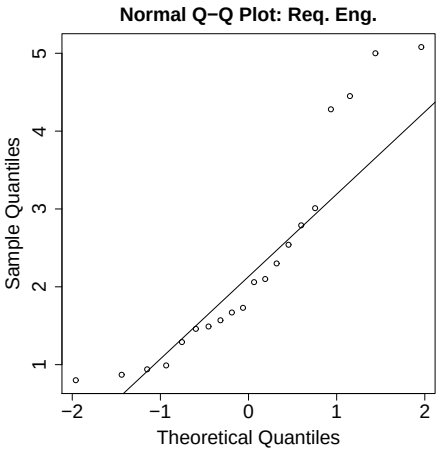
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.7787101,$$
$$p = 4.430485\text{e-}07$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

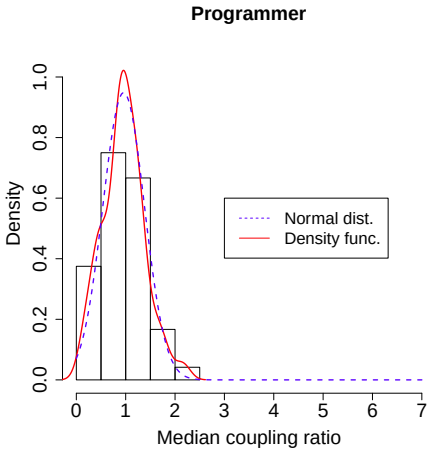


(d) QQ-Plot for requirements engineer-written requirements documents.

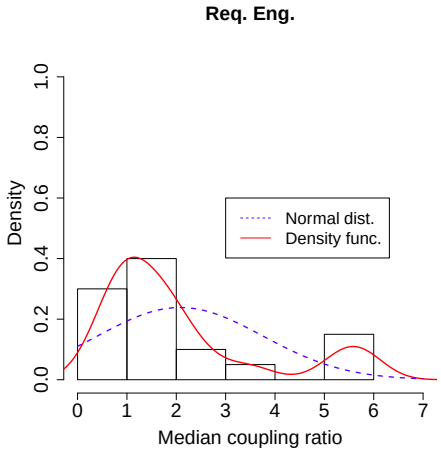
$$W = 0.8681896,$$
$$p = 0.01092081$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

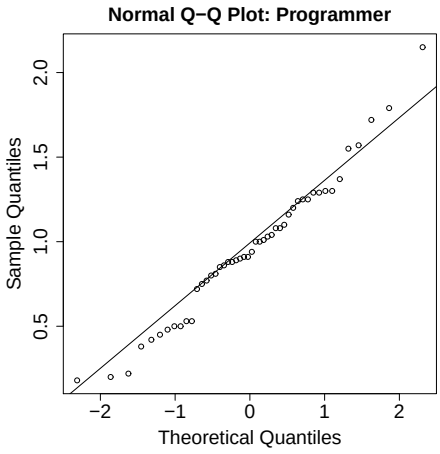
Figure 107: Normality tests for average coupling ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



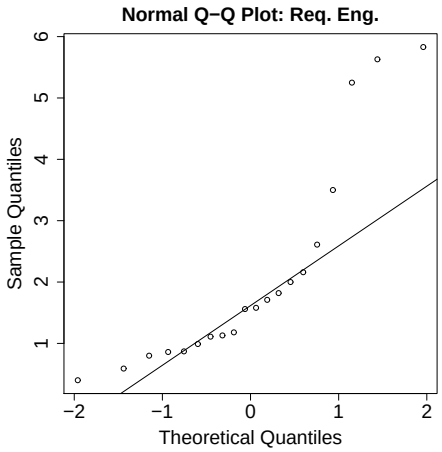
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9764807,$$
$$p = 0.4418487$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

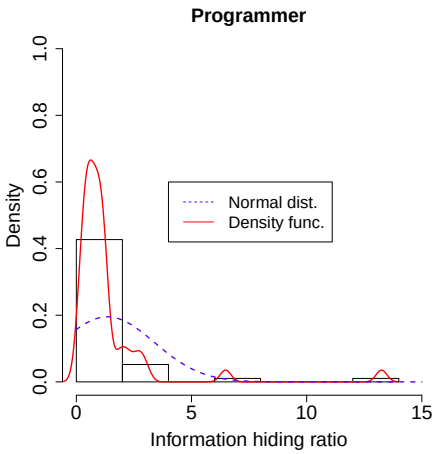


(d) QQ-Plot for requirements engineer-written requirements documents.

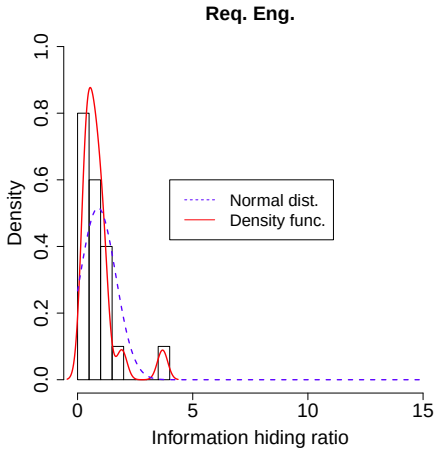
$$W = 0.7917488,$$
$$p = 0.0006524934$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

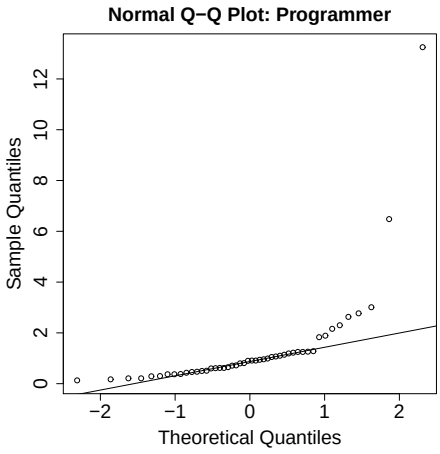
Figure 108: Normality tests for median coupling ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



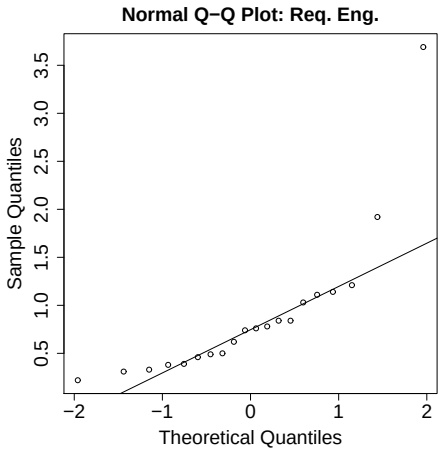
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.4700603,$$
$$p = 6.293918\text{e-}12$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

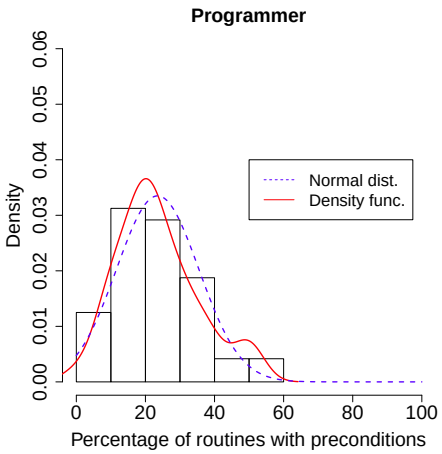


(d) QQ-Plot for requirements engineer-written requirements documents.

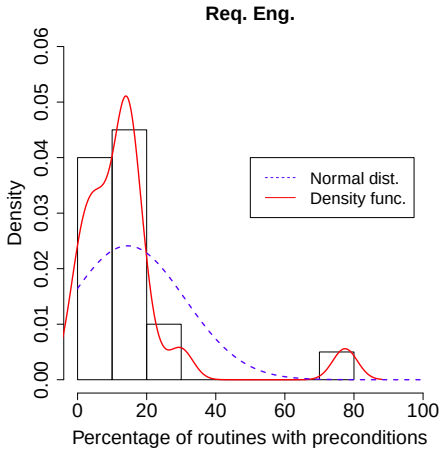
$$W = 0.6936615,$$
$$p = 3.26643\text{e-}05$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

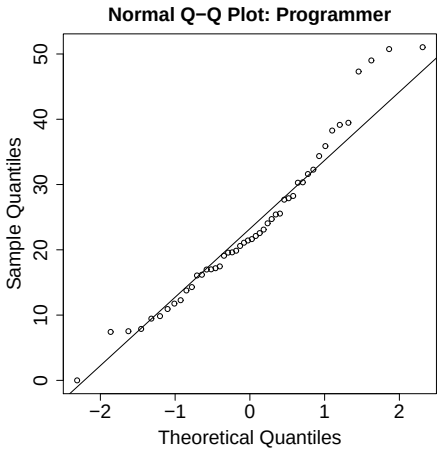
Figure 109: Normality tests for information hiding ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



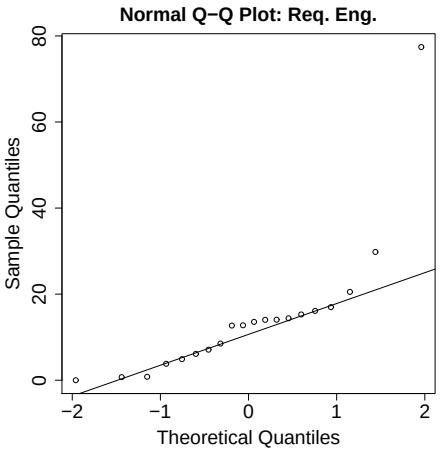
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.958913,$$
$$p = 0.09124195$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

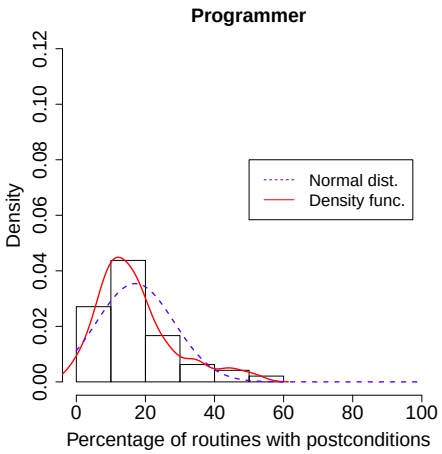


(d) QQ-Plot for requirements engineer-written requirements documents.

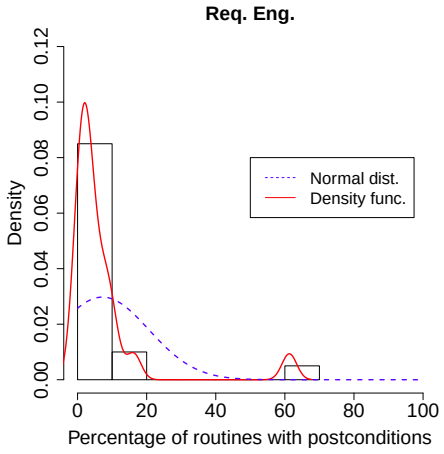
$$W = 0.6492008,$$
$$p = 9.958875\text{e-}06$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

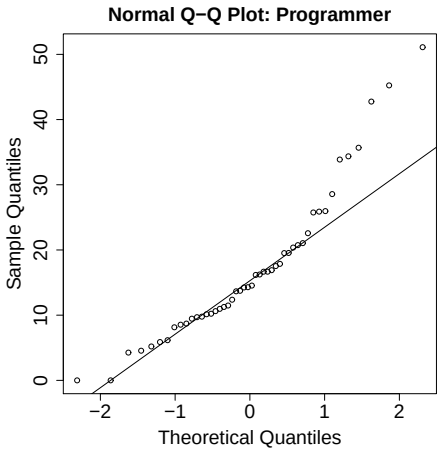
Figure 110: Normality tests for percentage of routines with preconditions of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



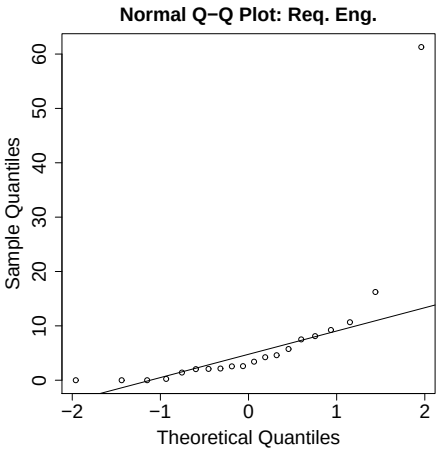
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$W = 0.9105844$,
$p = 0.00140638$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

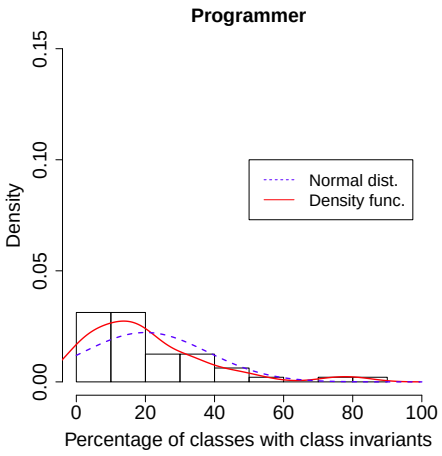


(d) QQ-Plot for requirements engineer-written requirements documents.

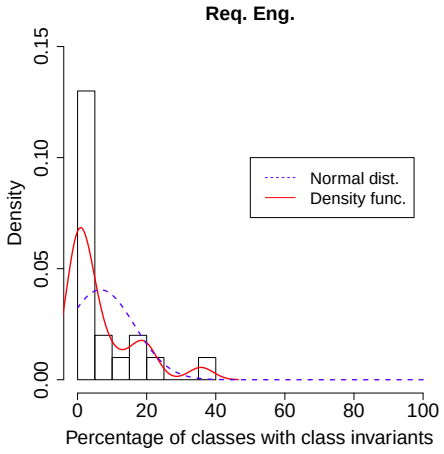
$W = 0.5000885$,
$p = 3.184572\text{e-}07$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

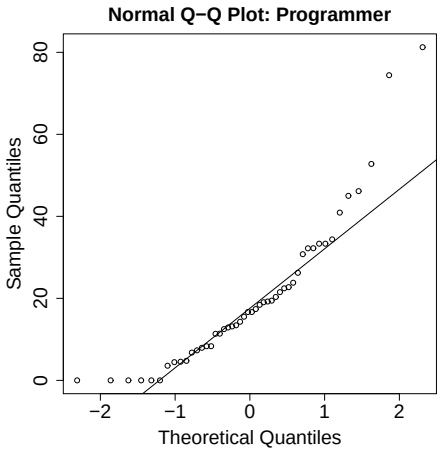
Figure 111: Normality tests for percentage of routines with postconditions of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



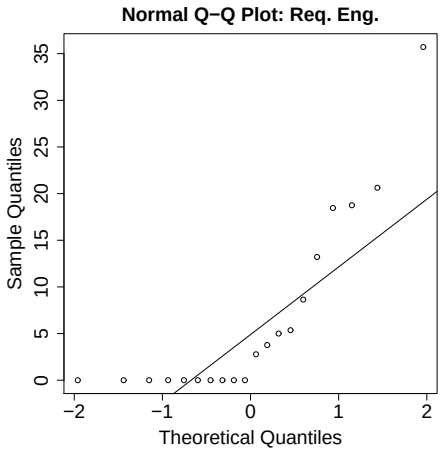
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.8651632,$$
$$p = 5.636879\text{e-}05$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

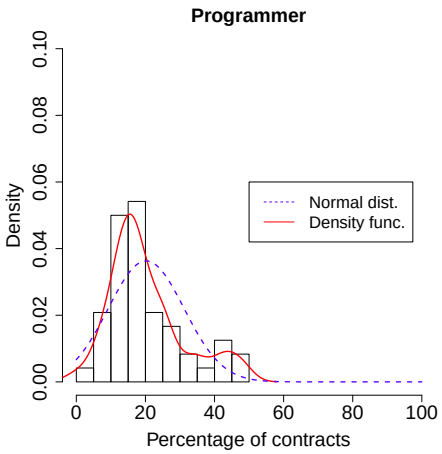


(d) QQ-Plot for requirements engineer-written requirements documents.

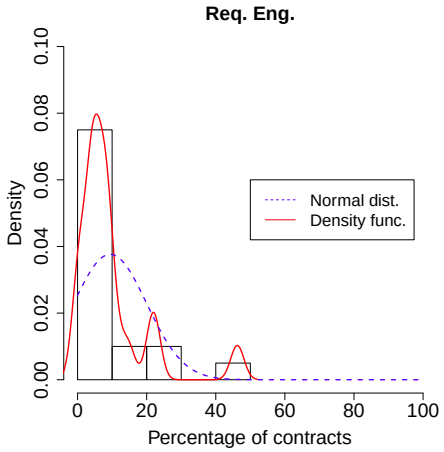
$$W = 0.7294054,$$
$$p = 9.104553\text{e-}05$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

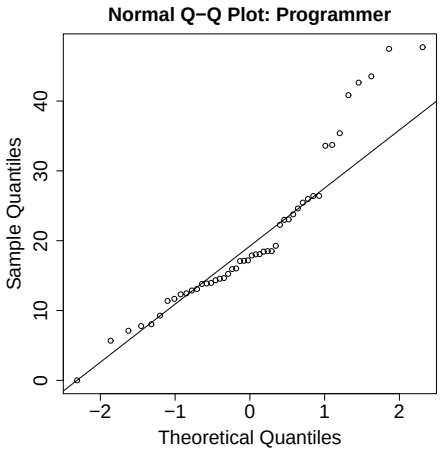
Figure 112: Normality tests for percentage of classes with class invariants of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



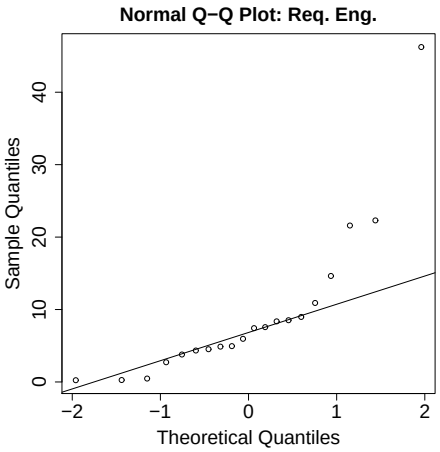
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9088658,$$
$$p = 0.001230903$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

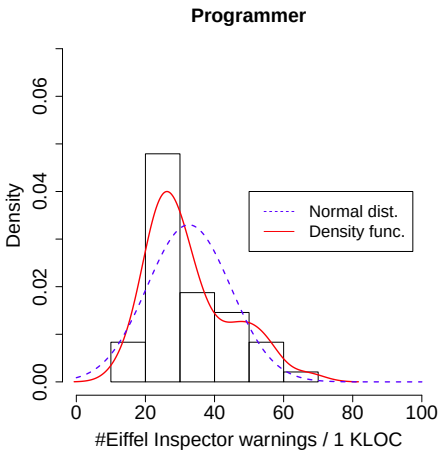


(d) QQ-Plot for requirements engineer-written requirements documents.

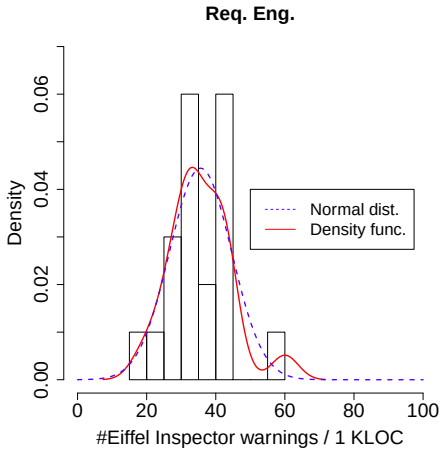
$$W = 0.7250816,$$
$$p = 8.013356\text{e-}05$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

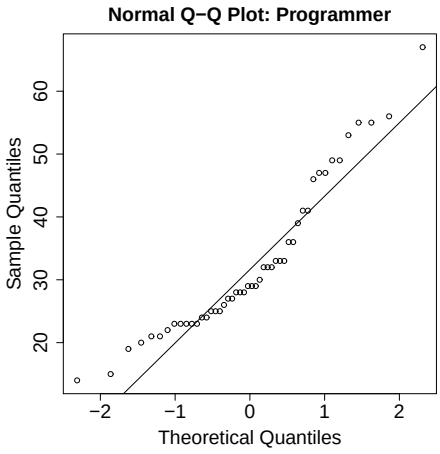
Figure 113: Normality tests for average percentage of contracts of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



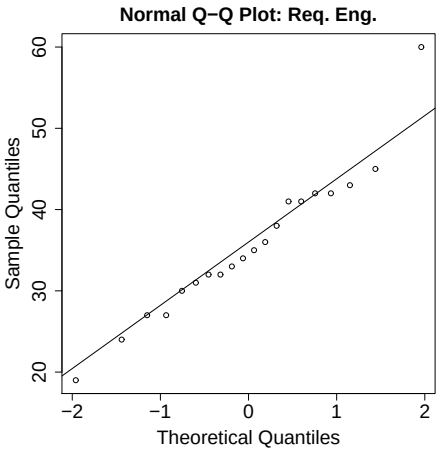
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9135019,$$
$$p = 0.001767548$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

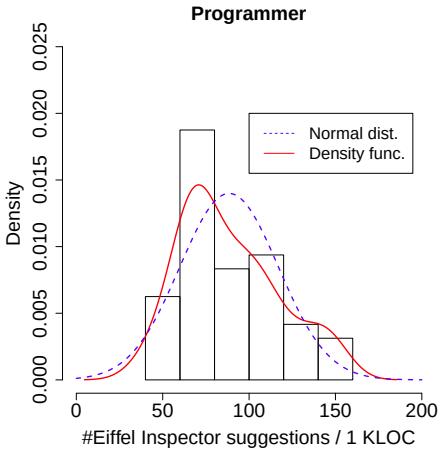


(d) QQ-Plot for requirements engineer-written requirements documents.

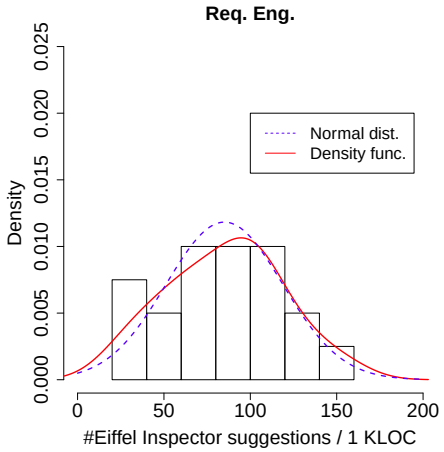
$$W = 0.9543196,$$
$$p = 0.437434$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

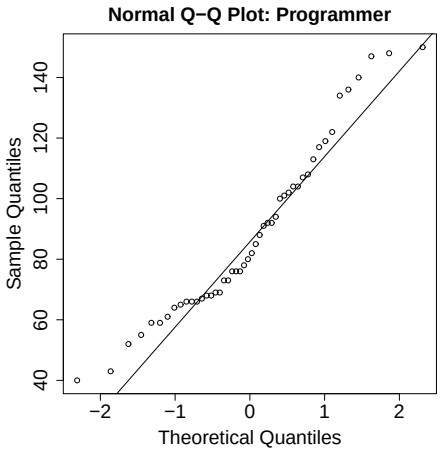
Figure 114: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



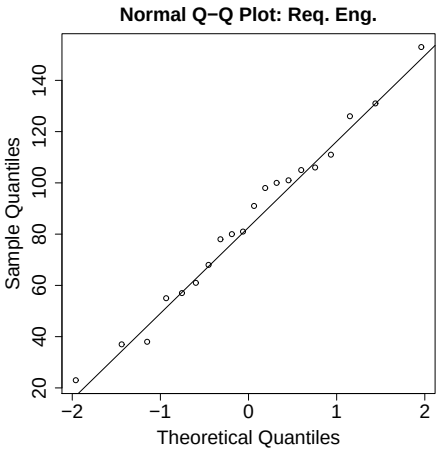
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9449731,$$
$$p = 0.02531208$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

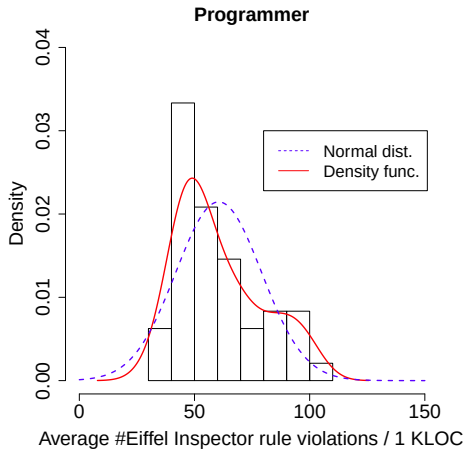


(d) QQ-Plot for requirements engineer-written requirements documents.

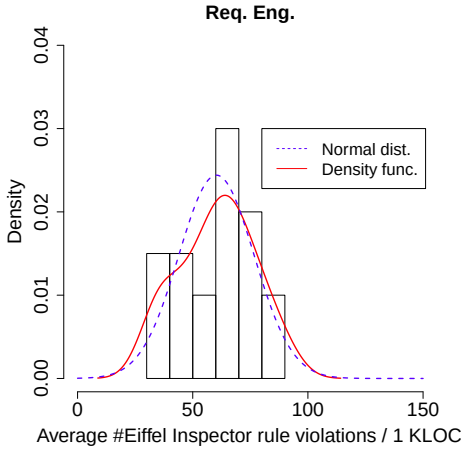
$$W = 0.9846575,$$
$$p = 0.979391$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

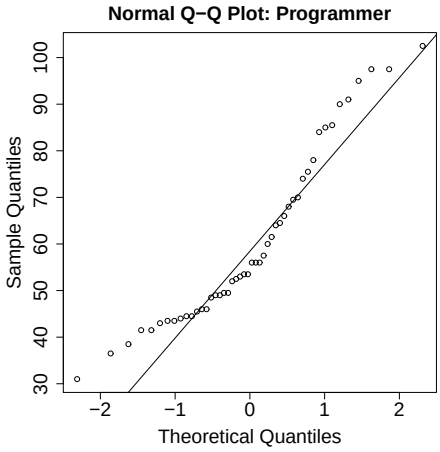
Figure 115: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



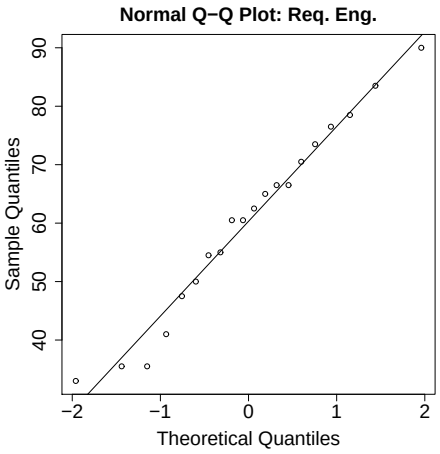
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9184665,$$
$$p = 0.002625928$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.



(d) QQ-Plot for requirements engineer-written requirements documents.

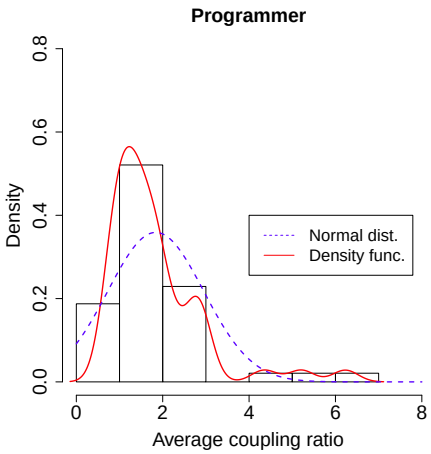
$$W = 0.9707123,$$
$$p = 0.7698443$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

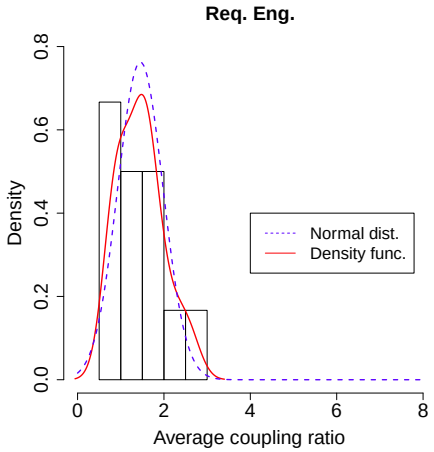
Figure 116: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.

B.2.2 Main analysis (2009 - 2012 vs. 2013)

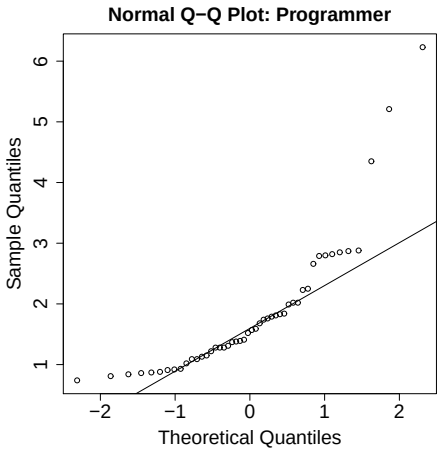
Figure 117 to 126 show the normality tests performed for the main analyis of research question RQ.2 comparing the projects from 2009 to 2012 with the projects from 2013.



(a) Histogram for programmer-written requirements documents.



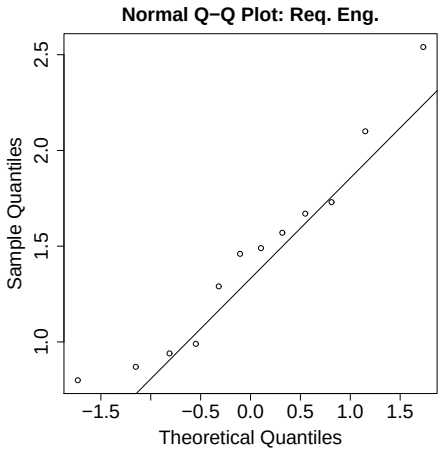
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.7787101,$$
$$p = 4.430485\text{e-}07$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

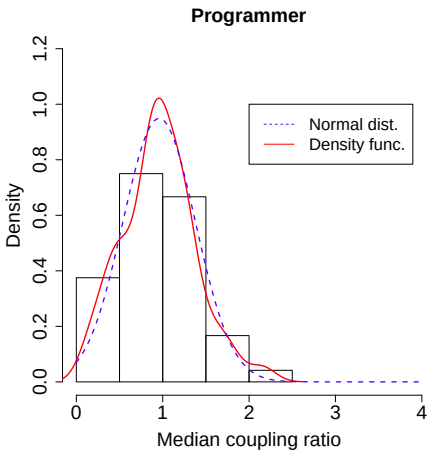


(d) QQ-Plot for requirements engineer-written requirements documents.

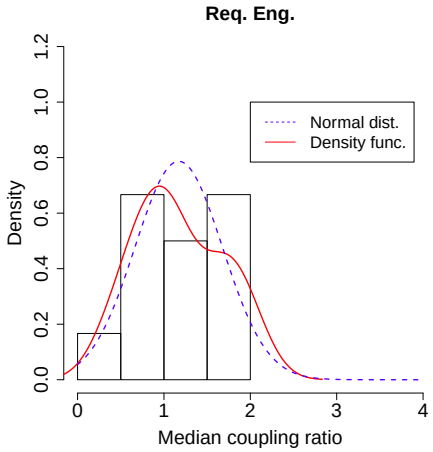
$$W = 0.9397359,$$
$$p = 0.4946585$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

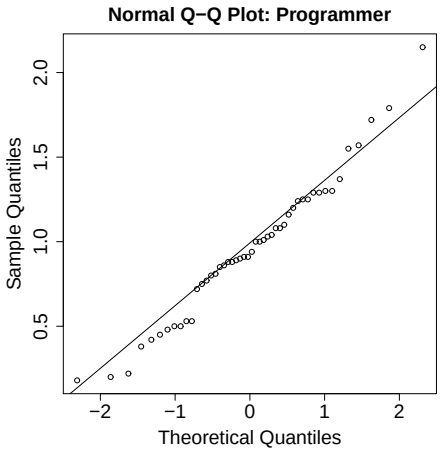
Figure 117: Normality tests for average coupling ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



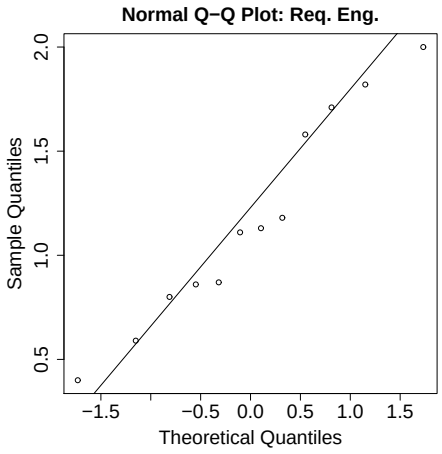
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9764807,$$
$$p = 0.4418487$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

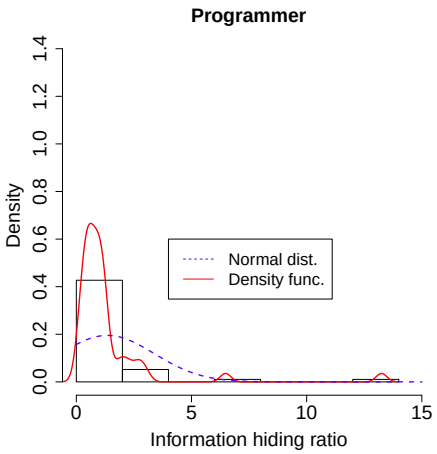


(d) QQ-Plot for requirements engineer-written requirements documents.

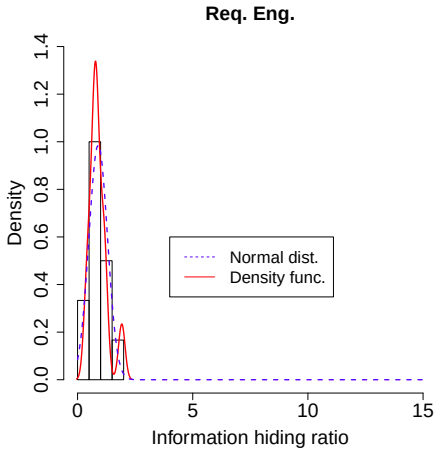
$$W = 0.953185,$$
$$p = 0.6839076$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

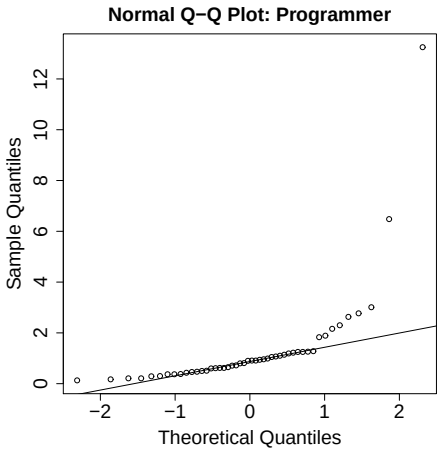
Figure 118: Normality tests for median coupling ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



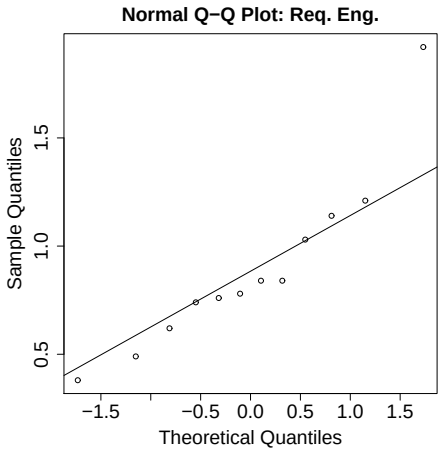
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.4700603,$$
$$p = 6.293918\text{e-}12$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

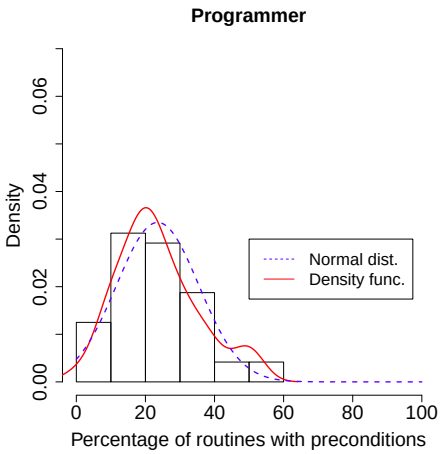


(d) QQ-Plot for requirements engineer-written requirements documents.

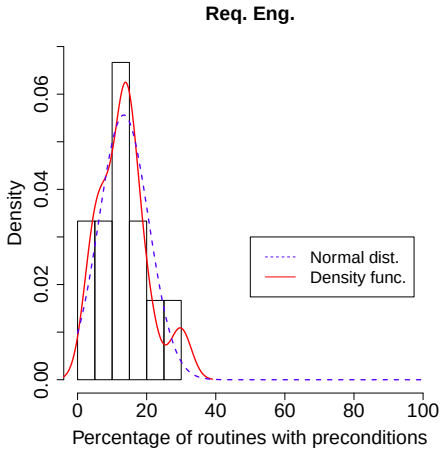
$$W = 0.8843561,$$
$$p = 0.09968543$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

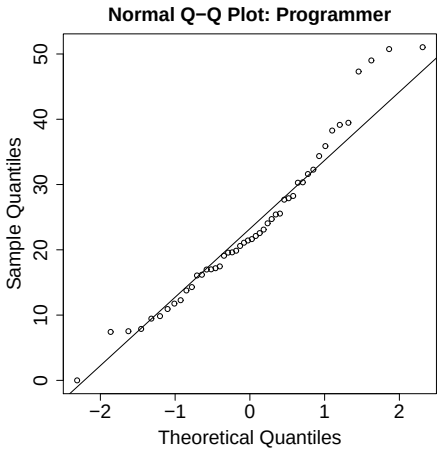
Figure 119: Normality tests for information hiding ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



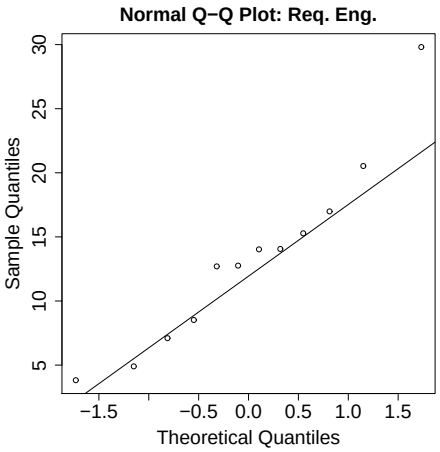
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.958913,$$
$$p = 0.09124195$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

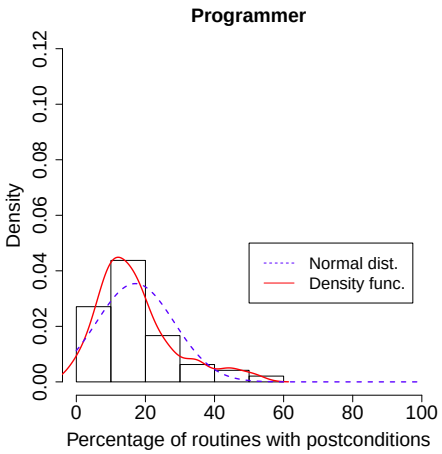


(d) QQ-Plot for requirements engineer-written requirements documents.

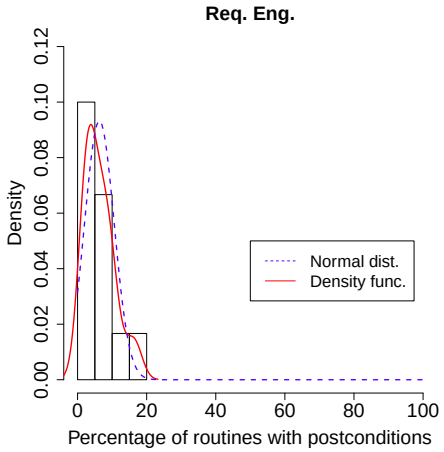
$$W = 0.9336334,$$
$$p = 0.4202194$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

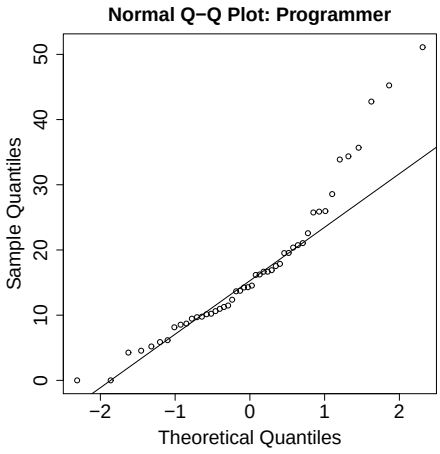
Figure 120: Normality tests for percentage of routines with preconditions of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



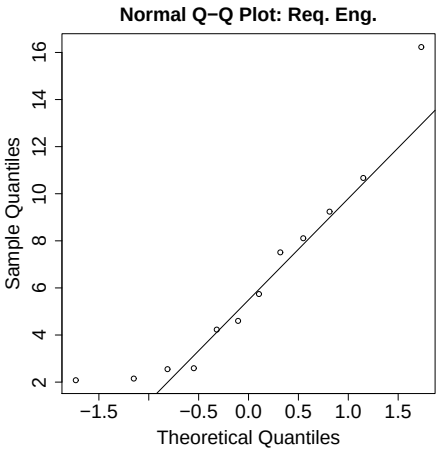
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9105844,$$
$$p = 0.00140638$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

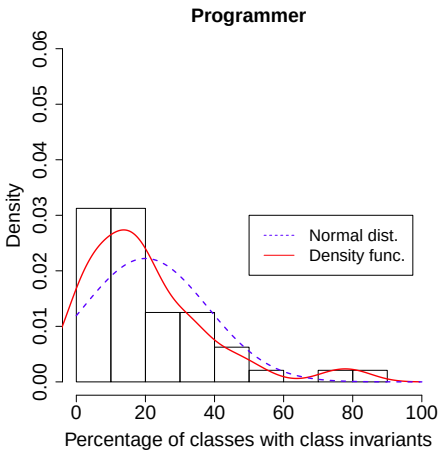


(d) QQ-Plot for requirements engineer-written requirements documents.

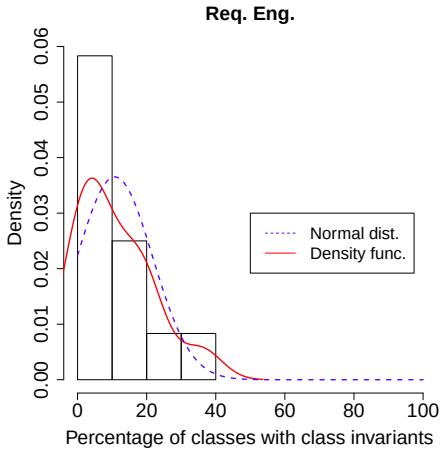
$$W = 0.8865686,$$
$$p = 0.1064361$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

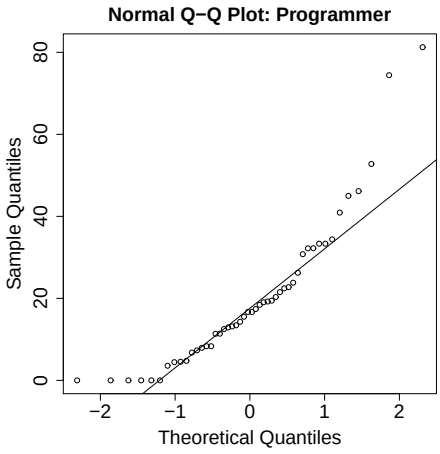
Figure 121: Normality tests for percentage of routines with postconditions of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



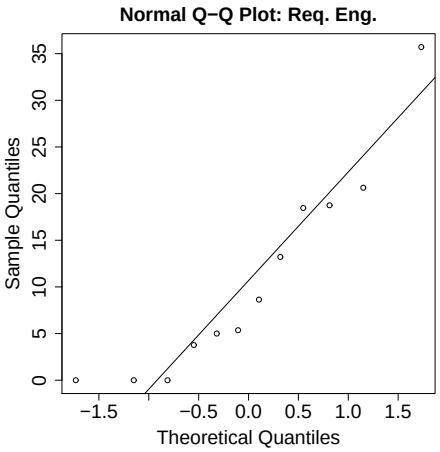
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.8651632,$$
$$p = 5.636879\text{e-}05$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

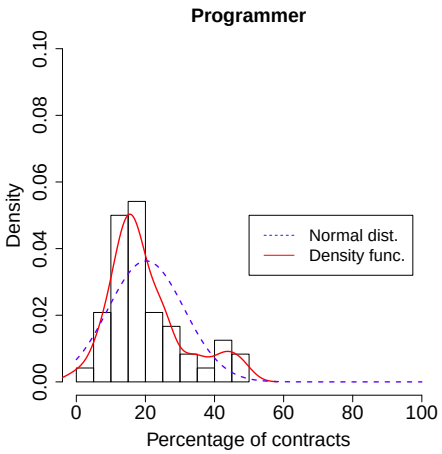


(d) QQ-Plot for requirements engineer-written requirements documents.

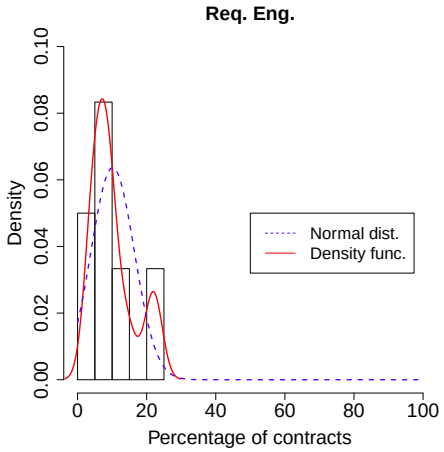
$$W = 0.8810316,$$
$$p = 0.09035545$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

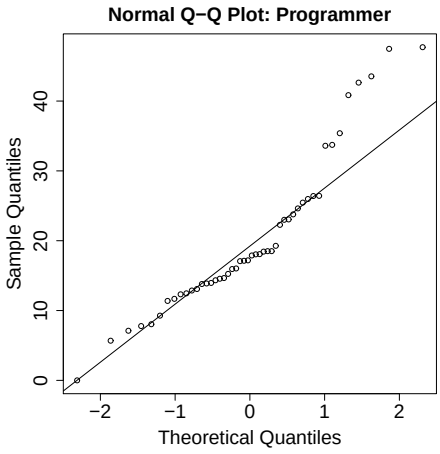
Figure 122: Normality tests for percentage of classes with class invariants of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



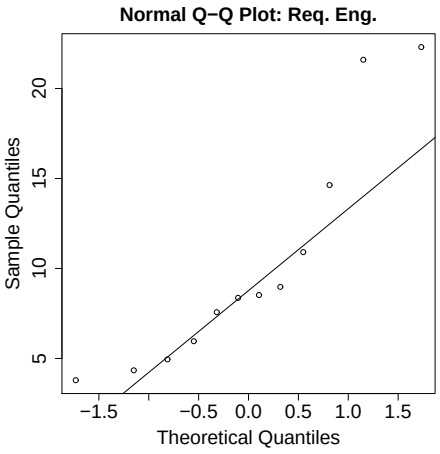
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9088658,$$
$$p = 0.001230903$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

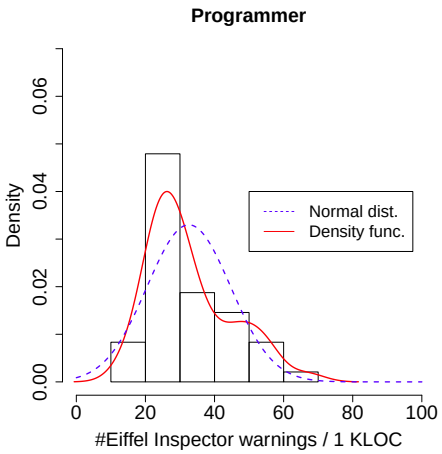


(d) QQ-Plot for requirements engineer-written requirements documents.

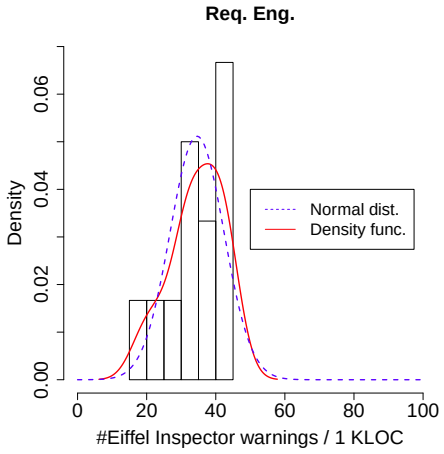
$$W = 0.8413115,$$
$$p = 0.02872399$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

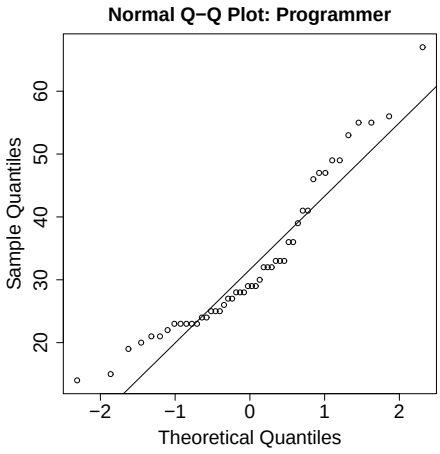
Figure 123: Normality tests for average percentage of contracts of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



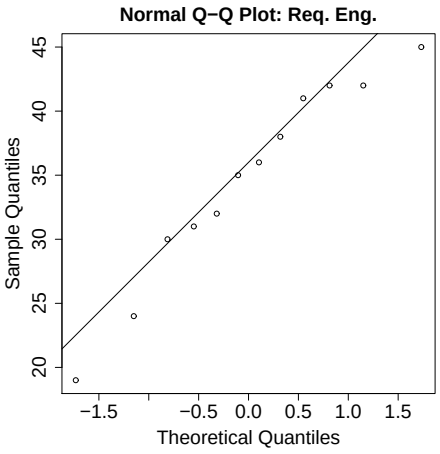
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9135019,$$
$$p = 0.001767548$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

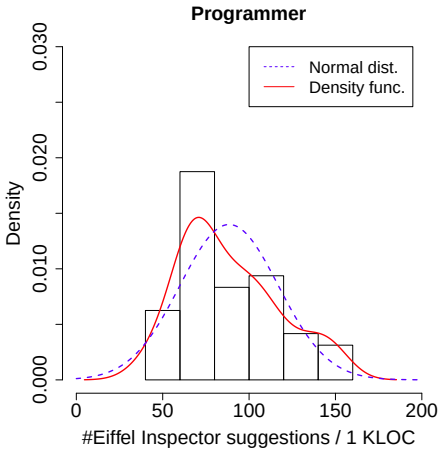


(d) QQ-Plot for requirements engineer-written requirements documents.

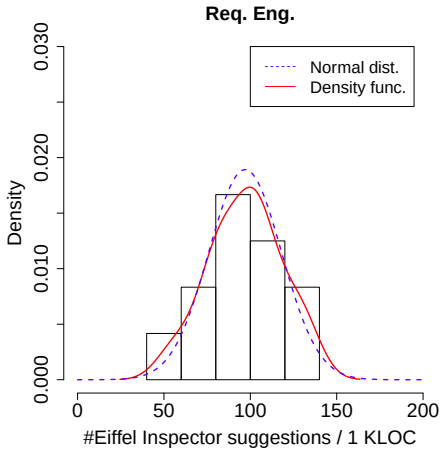
$$W = 0.9499293,$$
$$p = 0.6359519$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

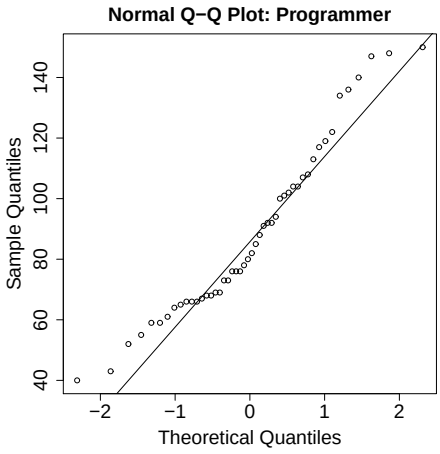
Figure 124: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



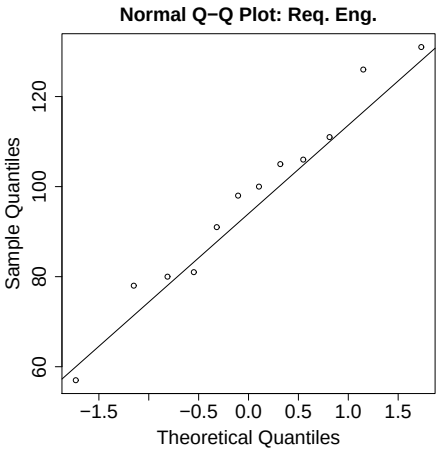
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9449731,$$
$$p = 0.02531208$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

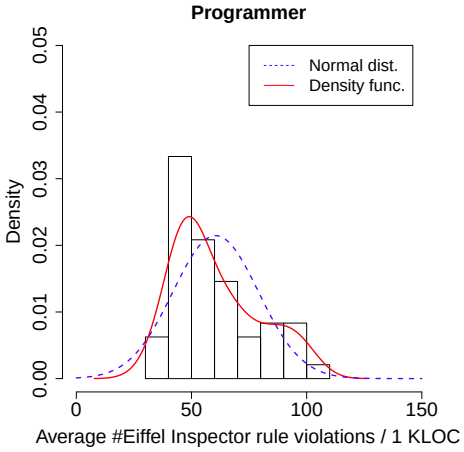


(d) QQ-Plot for requirements engineer-written requirements documents.

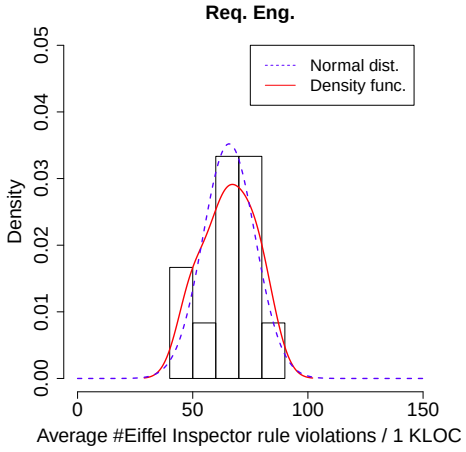
$$W = 0.9756132,$$
$$p = 0.9598853$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

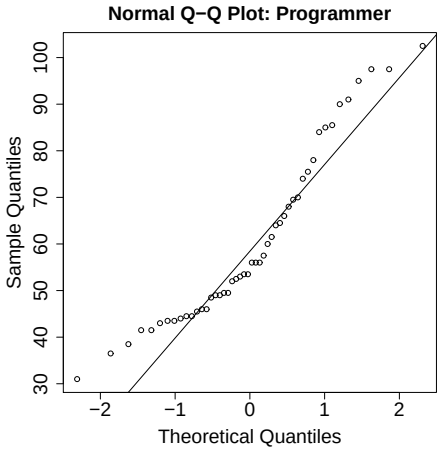
Figure 125: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



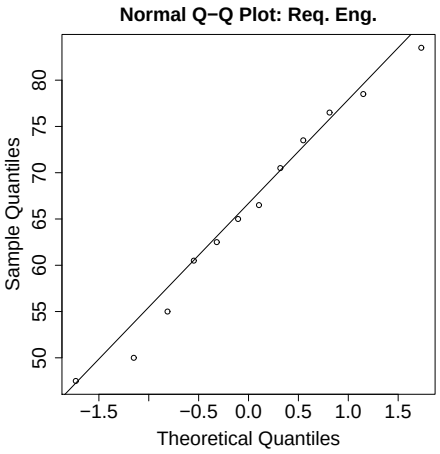
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9184665,$$
$$p = 0.002625928$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.



(d) QQ-Plot for requirements engineer-written requirements documents.

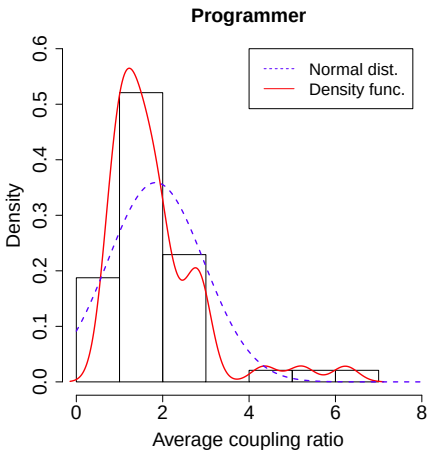
$$W = 0.9738655,$$
$$p = 0.9467902$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

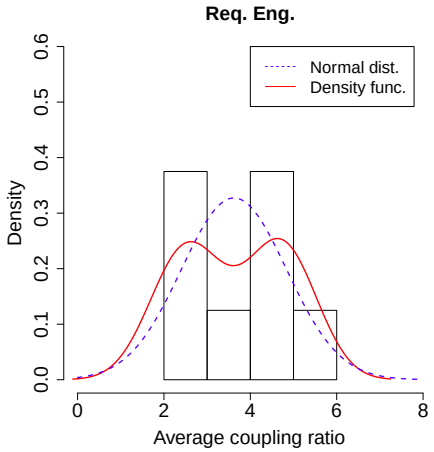
Figure 126: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.

B.2.3 Main analysis (2009 - 2012 vs. 2014)

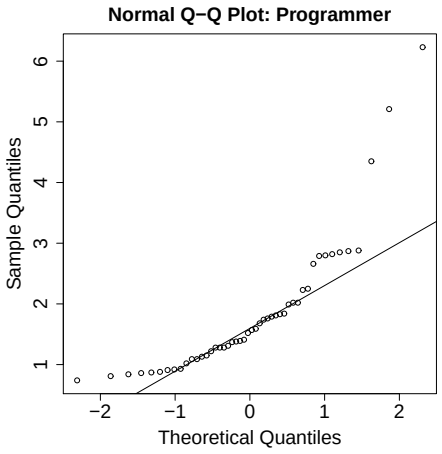
Figure 127 to 136 show the normality tests performed for the main analyis of research question RQ.2 comparing the projects from 2009 to 2012 with the projects from 2014.



(a) Histogram for programmer-written requirements documents.



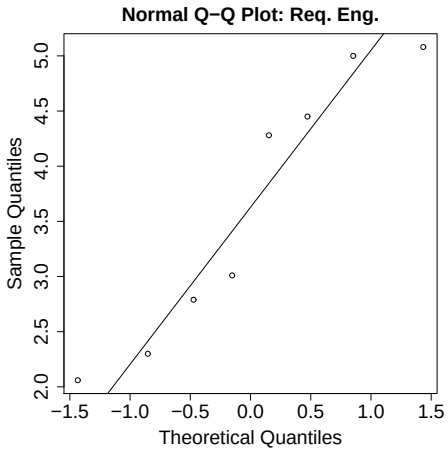
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.7787101,$$
$$p = 4.430485\text{e-}07$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

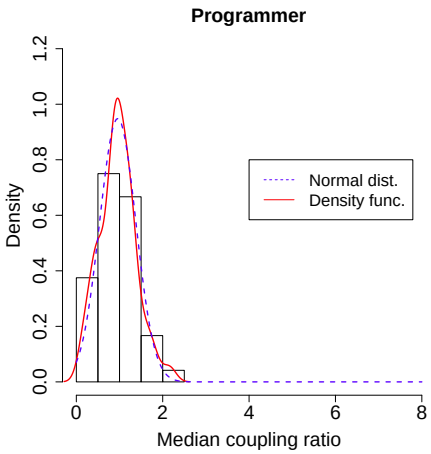


(d) QQ-Plot for requirements engineer-written requirements documents.

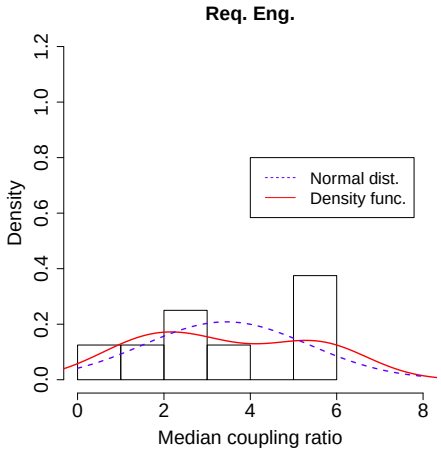
$$W = 0.8910062,$$
$$p = 0.2391208$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

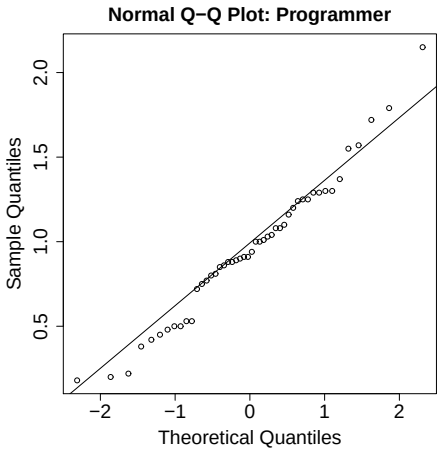
Figure 127: Normality tests for average coupling ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



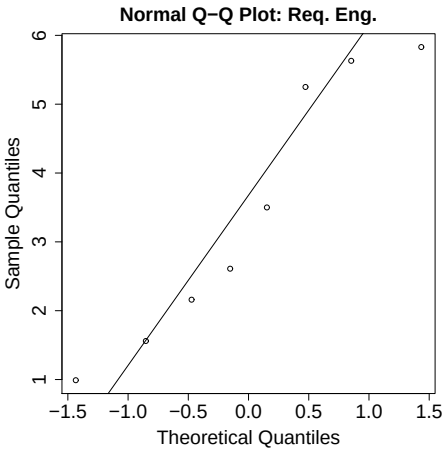
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9764807,$$
$$p = 0.4418487$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

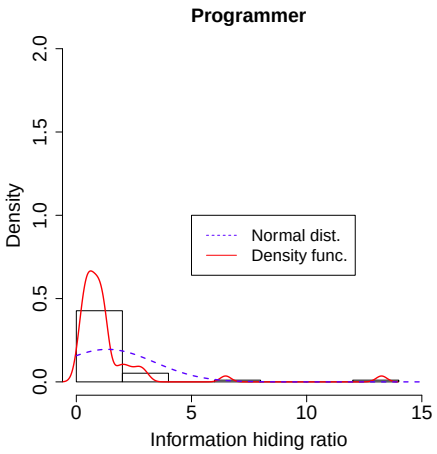


(d) QQ-Plot for requirements engineer-written requirements documents.

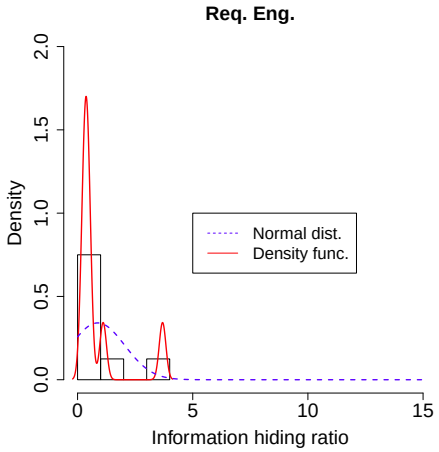
$$W = 0.9008719,$$
$$p = 0.2942188$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

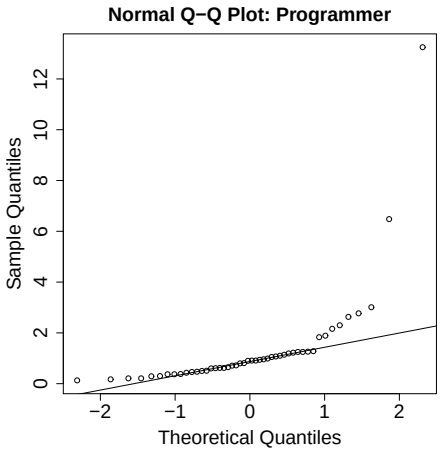
Figure 128: Normality tests for median coupling ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



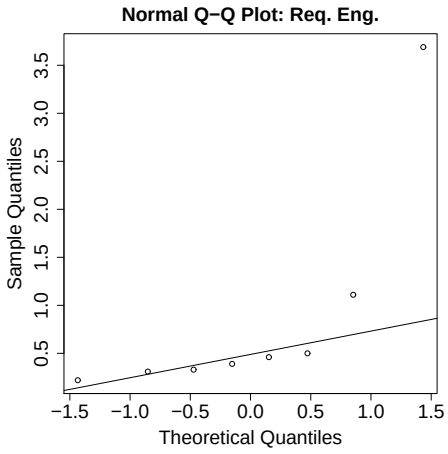
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.4700603,$$
$$p = 6.293918\text{e-}12$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

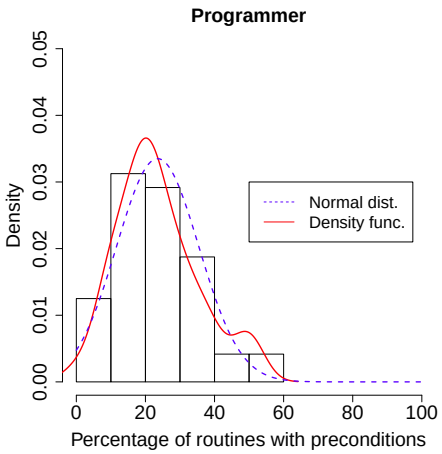


(d) QQ-Plot for requirements engineer-written requirements documents.

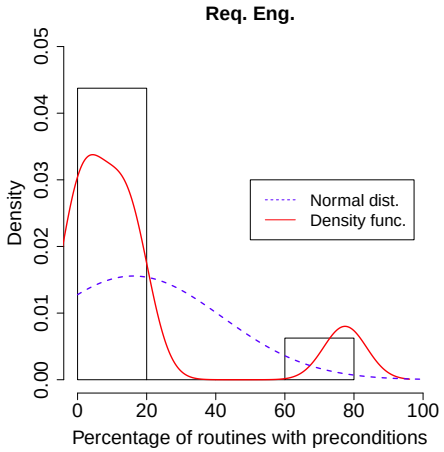
$$W = 0.5948687,$$
$$p = 0.0001384617$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

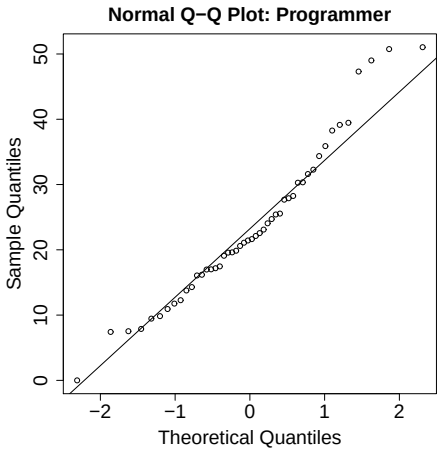
Figure 129: Normality tests for information hiding ratio of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



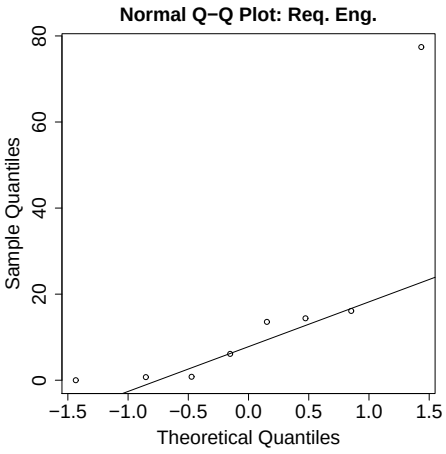
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.958913,$$
$$p = 0.09124195$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

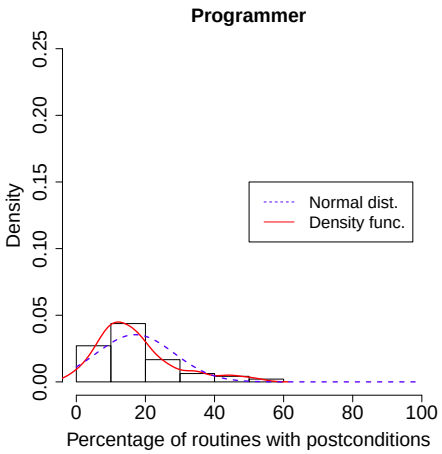


(d) QQ-Plot for requirements engineer-written requirements documents.

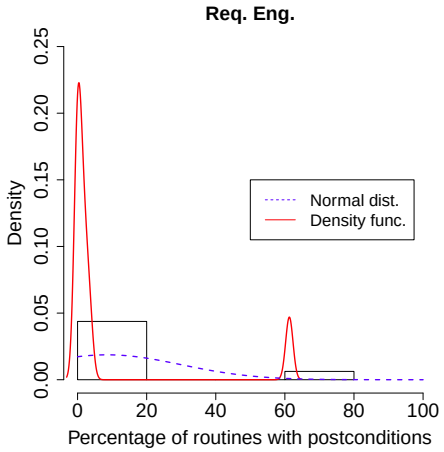
$$W = 0.6460561,$$
$$p = 0.0005450104$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

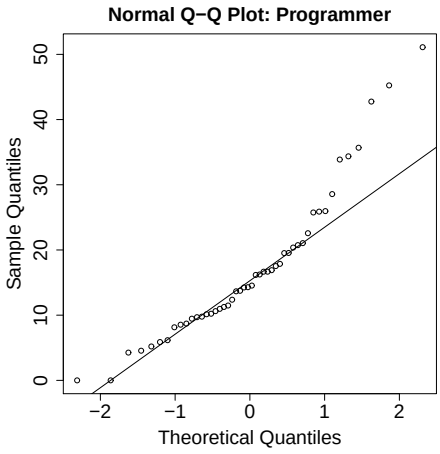
Figure 130: Normality tests for percentage of routines with preconditions of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



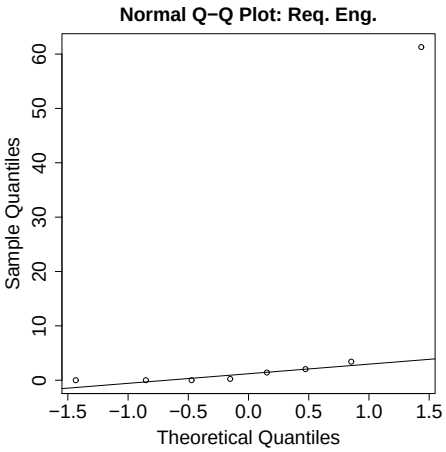
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9105844,$$
$$p = 0.00140638$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

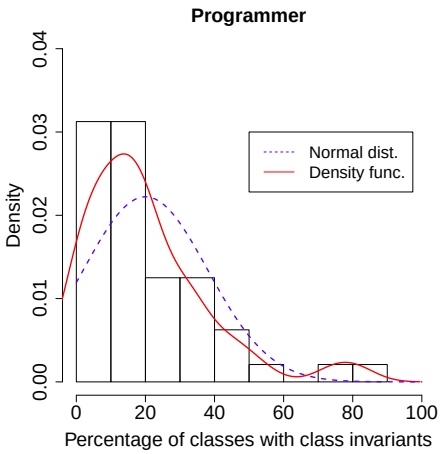


(d) QQ-Plot for requirements engineer-written requirements documents.

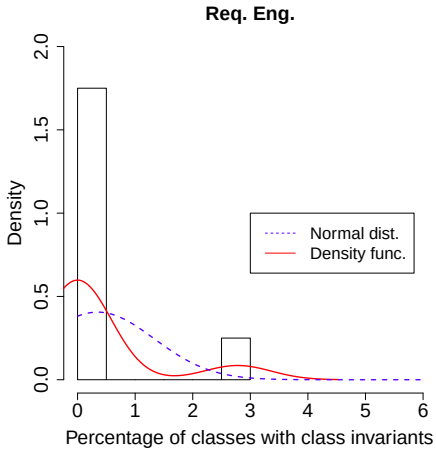
$$W = 0.4667864,$$
$$p = 4.096073\text{e-}06$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

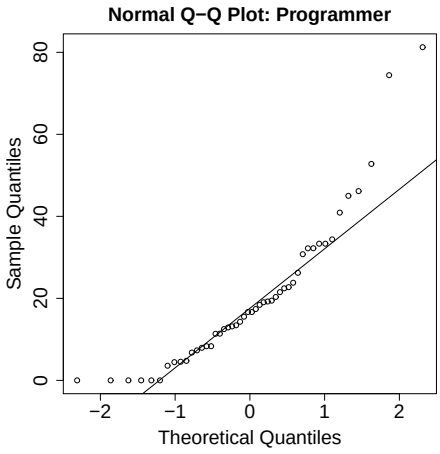
Figure 131: Normality tests for percentage of routines with postconditions of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



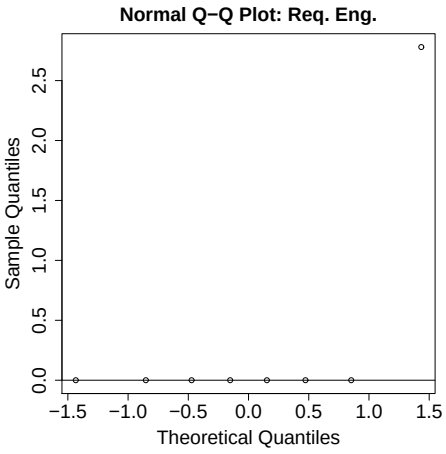
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.8651632,$$
$$p = 5.636879\text{e-}05$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

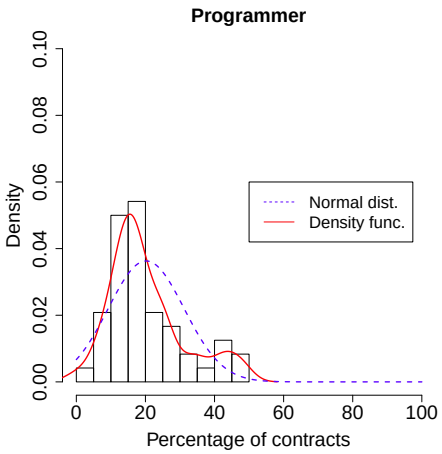


(d) QQ-Plot for requirements engineer-written requirements documents.

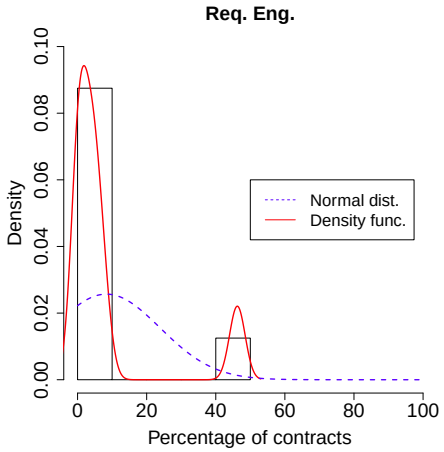
$$W = 0.4183984,$$
$$p = 1.047225\text{e-}06$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

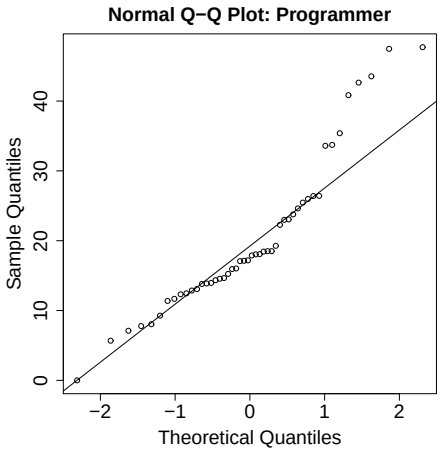
Figure 132: Normality tests for percentage of classes with class invariants of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



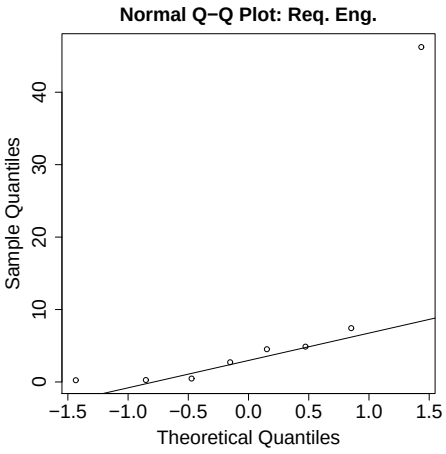
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9088658,$$
$$p = 0.001230903$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

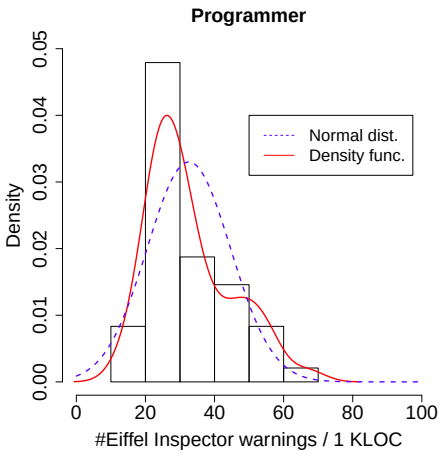


(d) QQ-Plot for requirements engineer-written requirements documents.

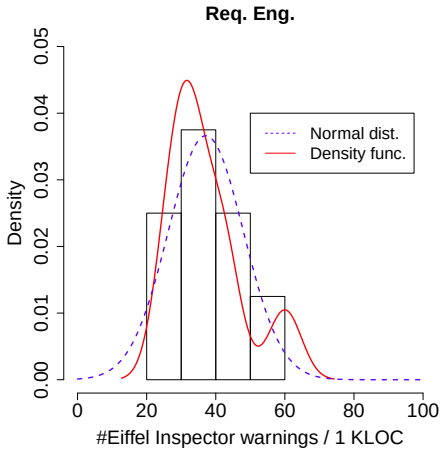
$$W = 0.568247,$$
$$p = 6.732357\text{e-}05$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

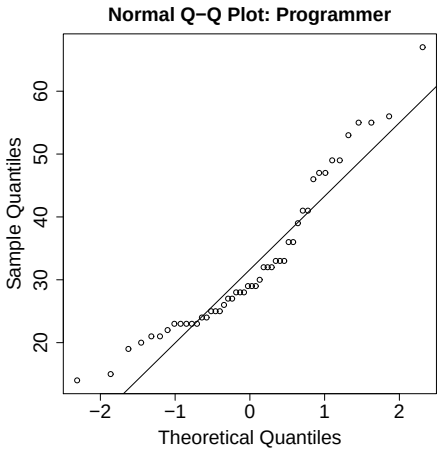
Figure 133: Normality tests for average percentage of contracts of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



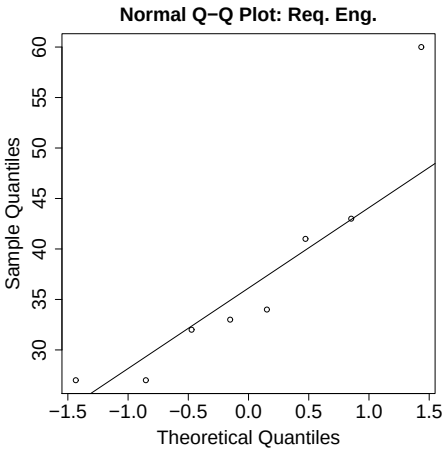
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9135019,$$
$$p = 0.001767548$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

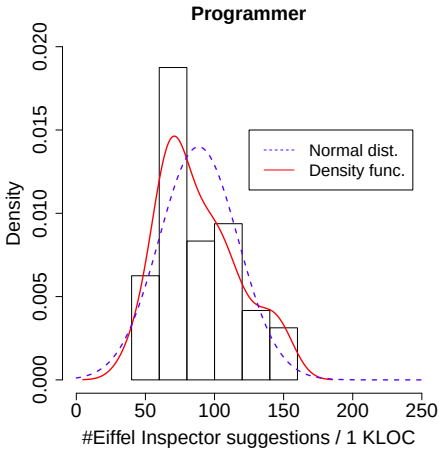


(d) QQ-Plot for requirements engineer-written requirements documents.

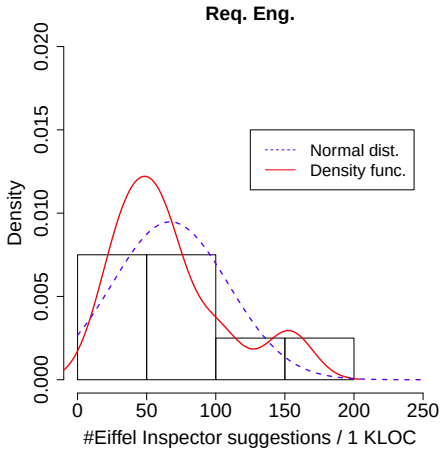
$$W = 0.855451,$$
$$p = 0.1081279$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

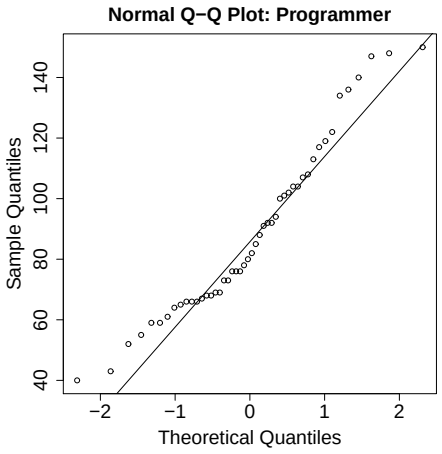
Figure 134: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



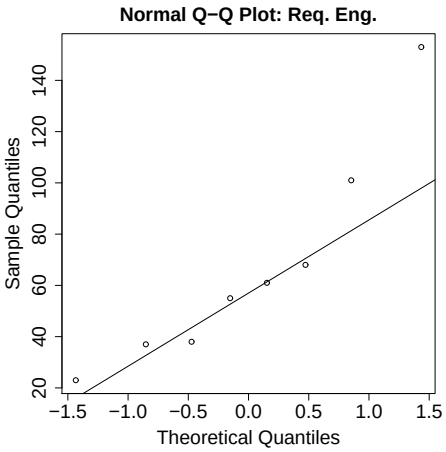
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9449731,$$
$$p = 0.02531208$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.

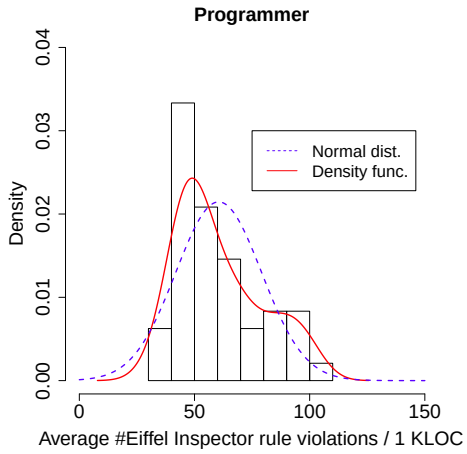


(d) QQ-Plot for requirements engineer-written requirements documents.

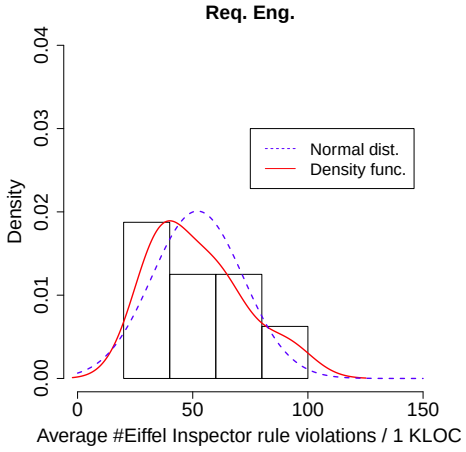
$$W = 0.8798461,$$
$$p = 0.1876921$$

(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

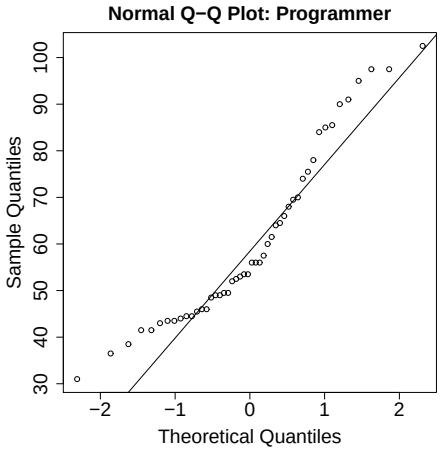
Figure 135: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.



(a) Histogram for programmer-written requirements documents.



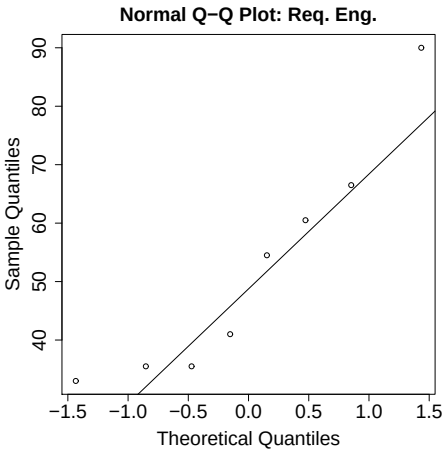
(b) Histogram for requirements engineer-written requirements documents.



(c) QQ-Plot for programmer-written requirements documents.

$$W = 0.9184665,$$
$$p = 0.002625928$$

(e) Shapiro-Wilk normality test for programmer-written requirements documents.



(d) QQ-Plot for requirements engineer-written requirements documents.

$$W = 0.8856615,$$
$$p = 0.2131334$$

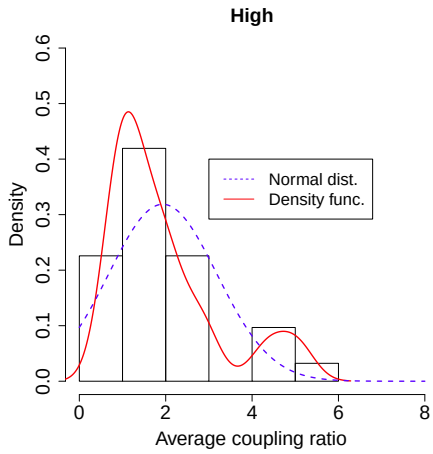
(f) Shapiro-Wilk normality test for requirements engineer-written requirements documents.

Figure 136: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with programmer-written requirements documents and projects with requirements engineer-written requirements documents.

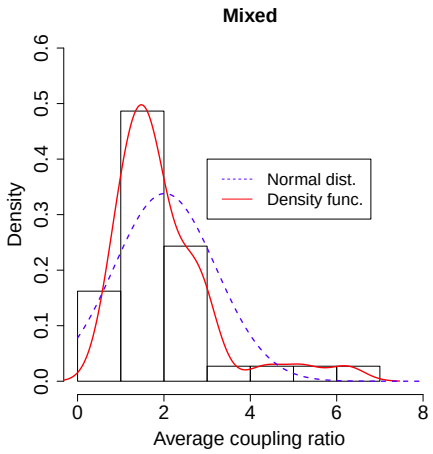
B.3 Research question RQ.3

B.3.1 Main analysis (2009 - 2014)

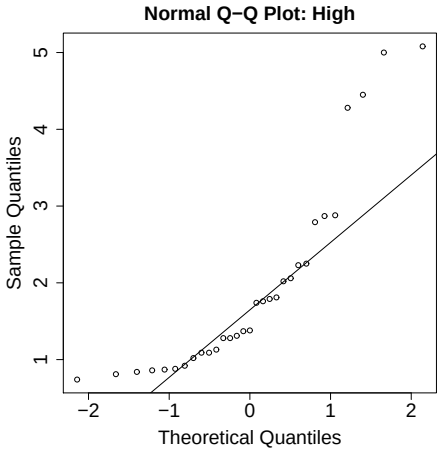
Figure 137 to 146 show the normality tests performed for the main analyis of research question RQ.3 using all projects.



(a) Histogram for high-context culture groups.



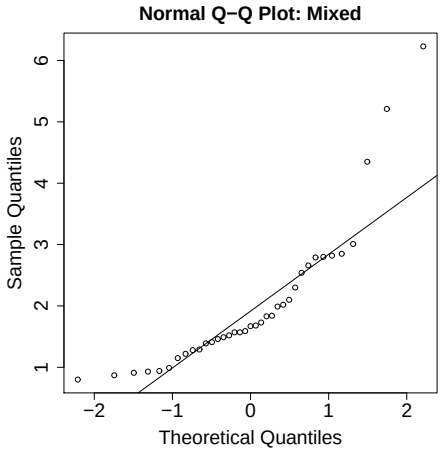
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.8081011,$$
$$p = 7.317087\text{e-}05$$

(e) Shapiro-Wilk normality test for high-context culture groups.

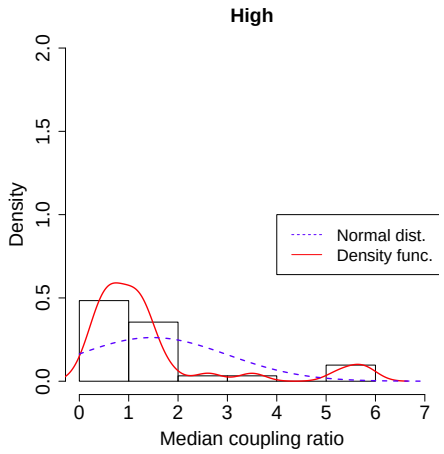


(d) QQ-Plot for mixed-context culture groups.

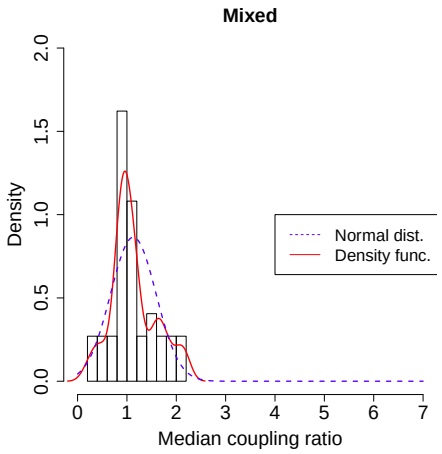
$$W = 0.7973817,$$
$$p = 1.154935\text{e-}05$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

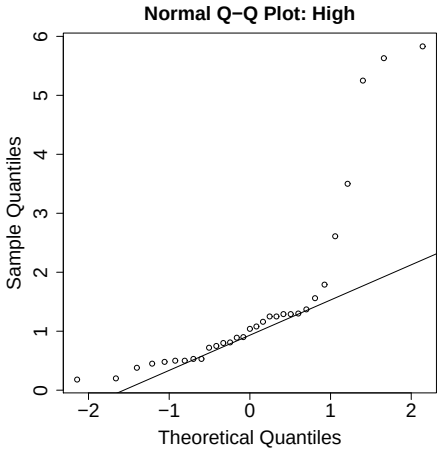
Figure 137: Normality tests for average coupling ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



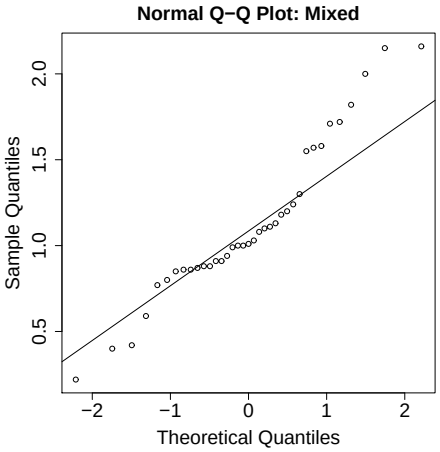
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

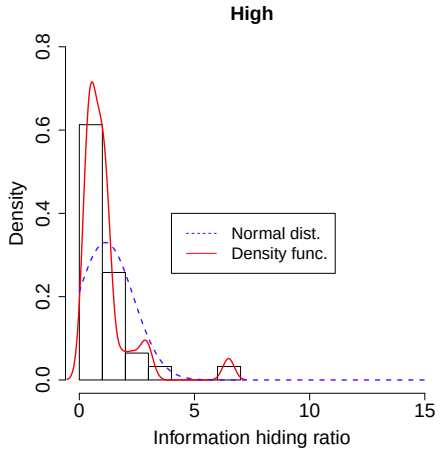
$$W = 0.6911152,$$
$$p = 8.616463\text{e-}07$$

(e) Shapiro-Wilk normality test for high-context culture groups.

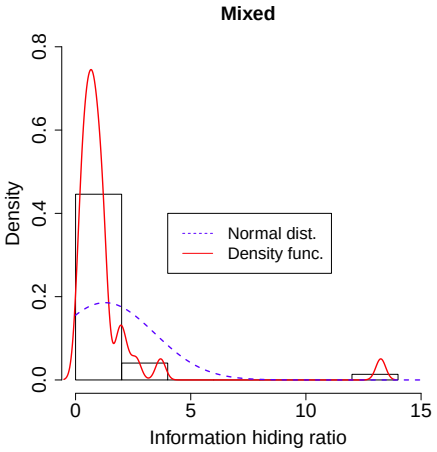
$$W = 0.9404252,$$
$$p = 0.04758538$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

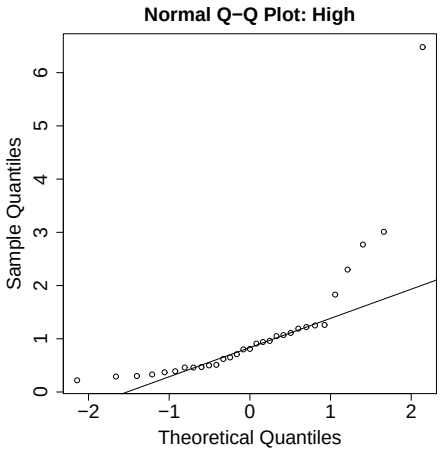
Figure 138: Normality tests for median coupling ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



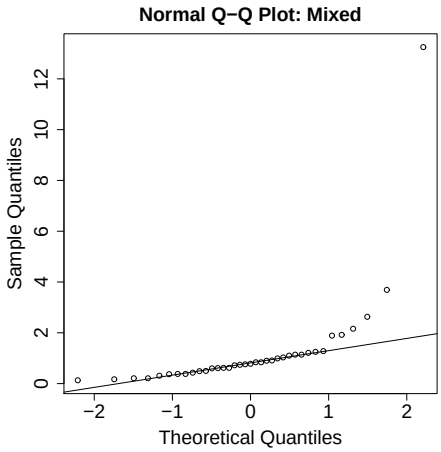
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.637516,$$
$$p = 1.569022\text{e-}07$$

(e) Shapiro-Wilk normality test for high-context culture groups.

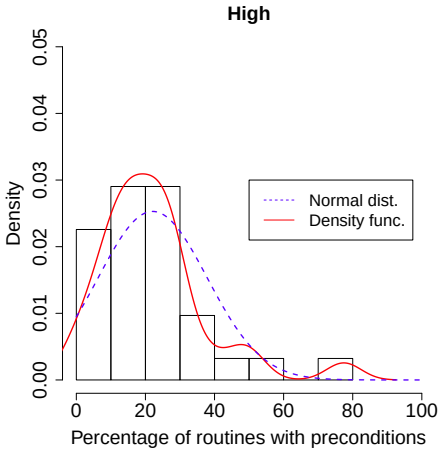


(d) QQ-Plot for mixed-context culture groups.

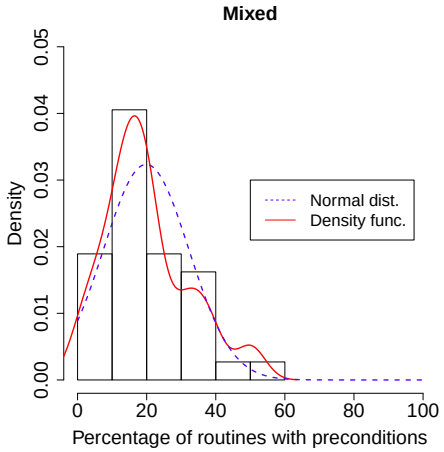
$$W = 0.4118016,$$
$$p = 5.07042\text{e-}11$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

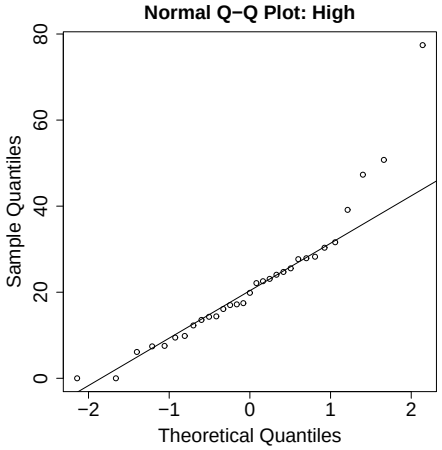
Figure 139: Normality tests for information hiding ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



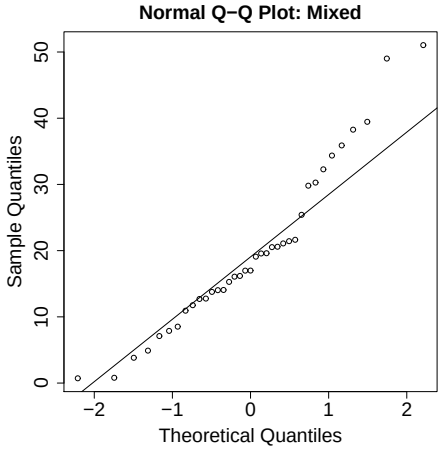
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

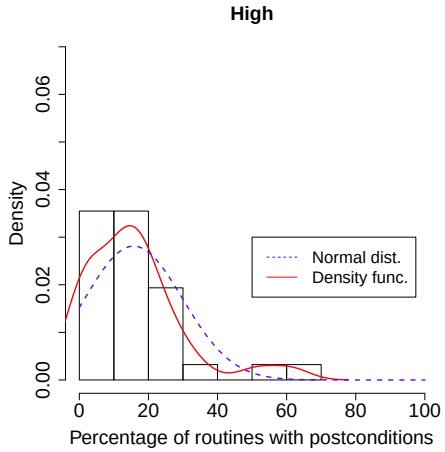
$$W = 0.8816508,$$
$$p = 0.00258874$$

(e) Shapiro-Wilk normality test for high-context culture groups.

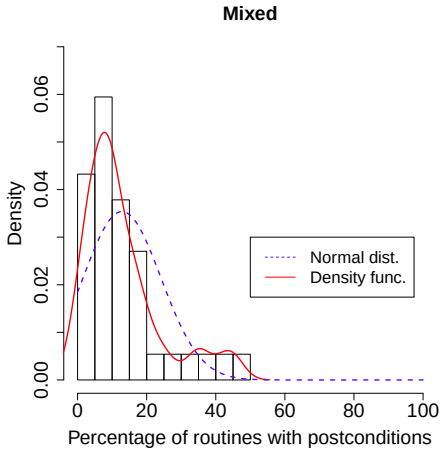
$$W = 0.9429703,$$
$$p = 0.05736539$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

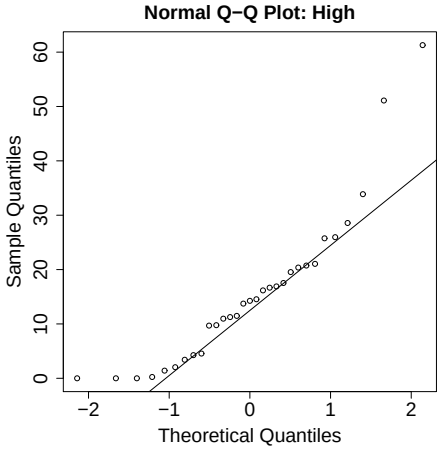
Figure 140: Normality tests for percentage of routines with preconditions of projects with only high-context culture groups and projects with mixed-context culture groups.



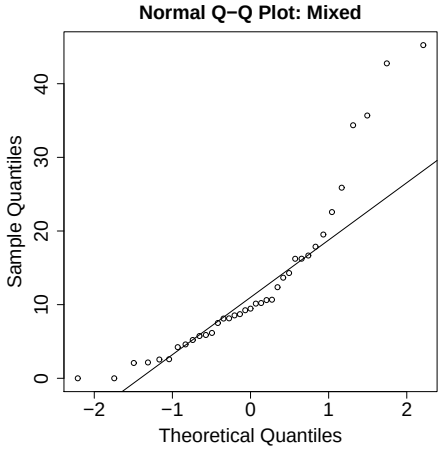
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

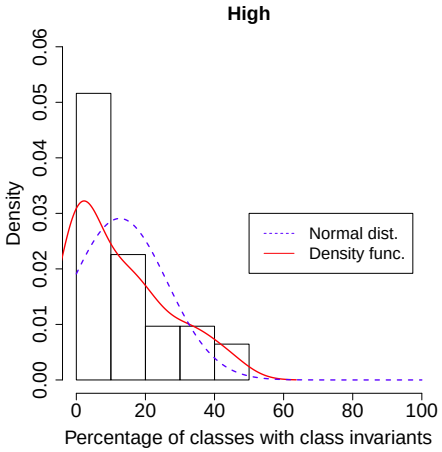
$$W = 0.8626568,$$
$$p = 0.0009540699$$

(e) Shapiro-Wilk normality test for high-context culture groups.

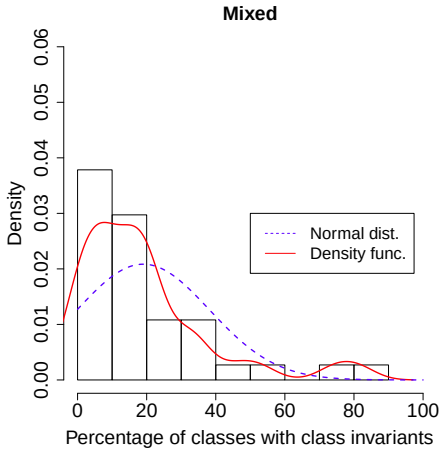
$$W = 0.8405321,$$
$$p = 9.53537\text{e-}05$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

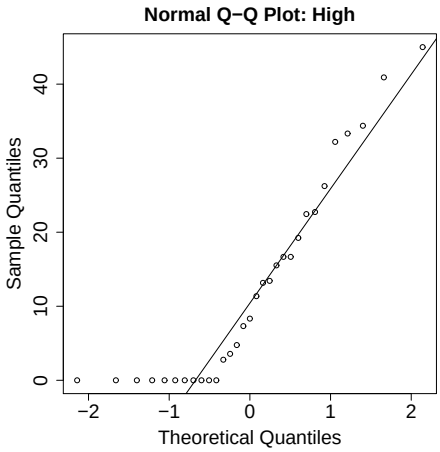
Figure 141: Normality tests for percentage of routines with postconditions of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



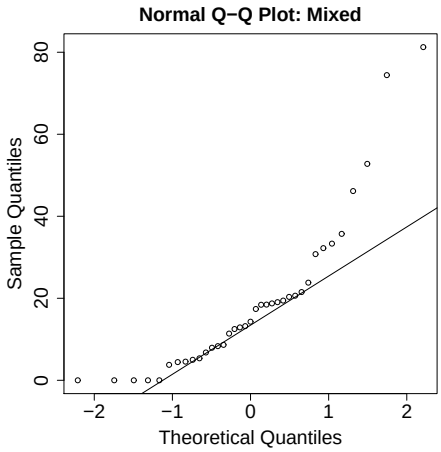
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.8537333,$$
$$p = 0.0006090788$$

(e) Shapiro-Wilk normality test for high-context culture groups.

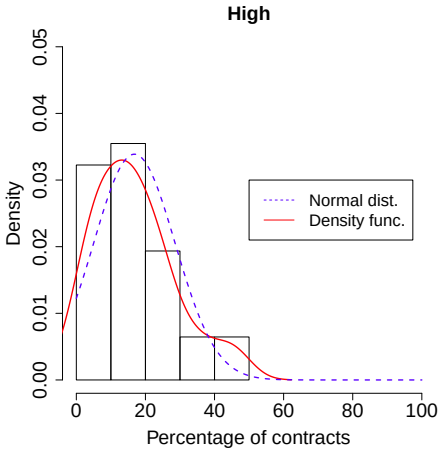


(d) QQ-Plot for mixed-context culture groups.

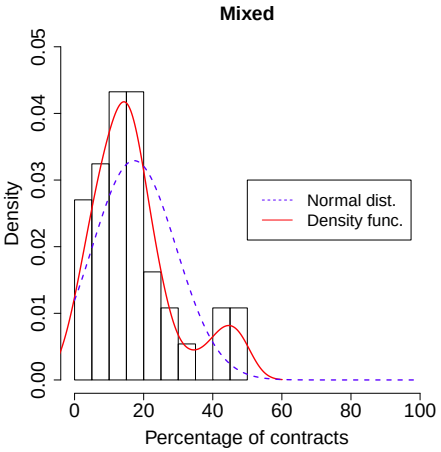
$$W = 0.8169213,$$
$$p = 2.907226\text{e-}05$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

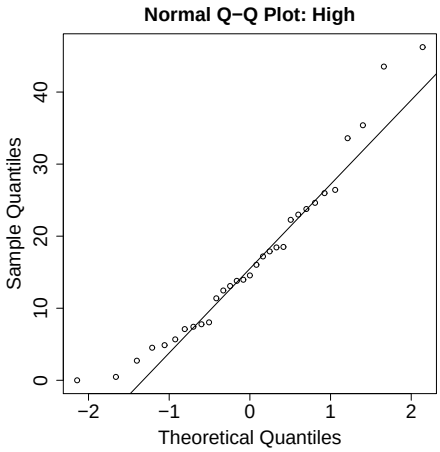
Figure 142: Normality tests for percentage of classes with class invariants of projects with only high-context culture groups and projects with mixed-context culture groups.



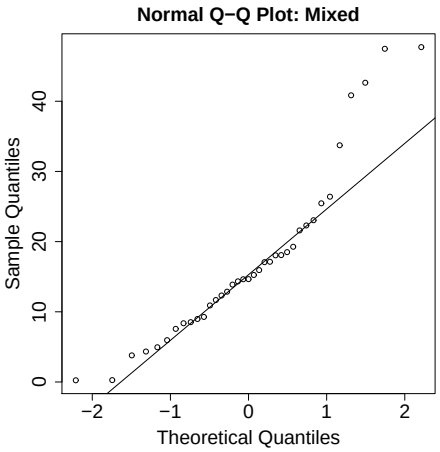
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

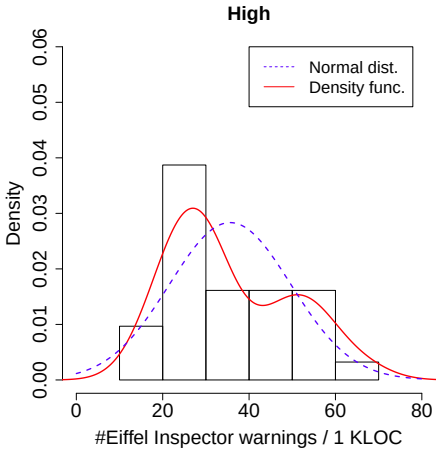
$$W = 0.9403896,$$
$$p = 0.08459119$$

(e) Shapiro-Wilk normality test for high-context culture groups.

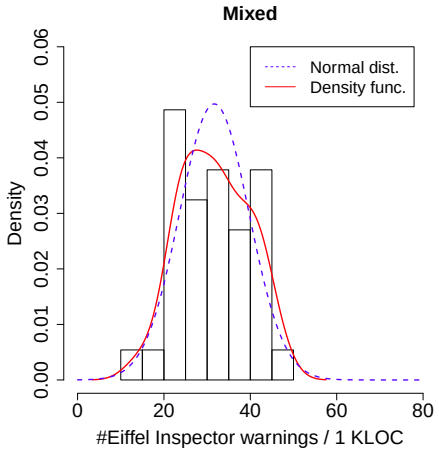
$$W = 0.8927861,$$
$$p = 0.00186904$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

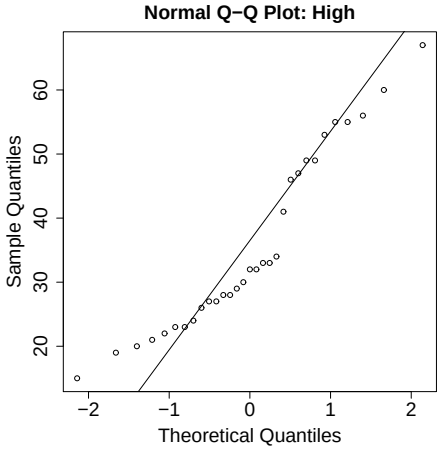
Figure 143: Normality tests for average percentage of contracts of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



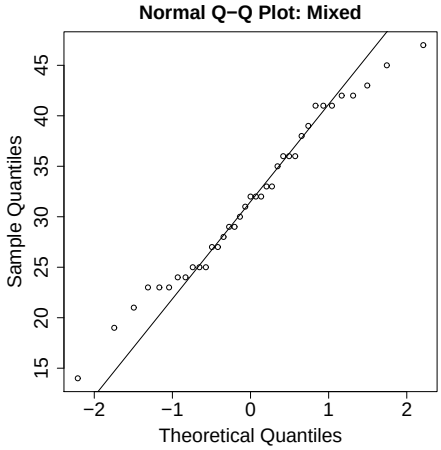
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9182023,$$
$$p = 0.02115637$$

(e) Shapiro-Wilk normality test for high-context culture groups.

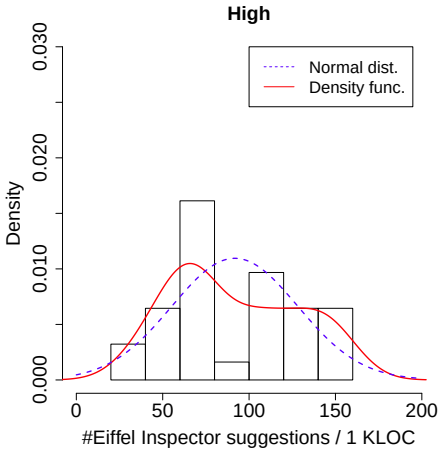


(d) QQ-Plot for mixed-context culture groups.

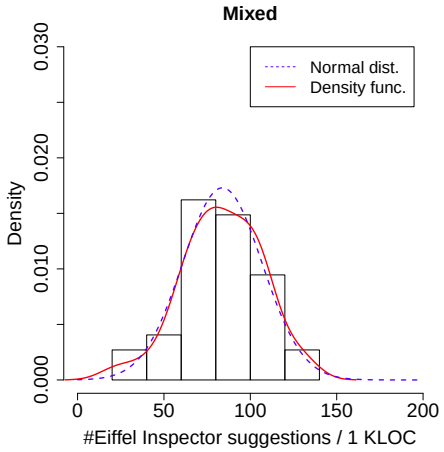
$$W = 0.9772082,$$
$$p = 0.6344855$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

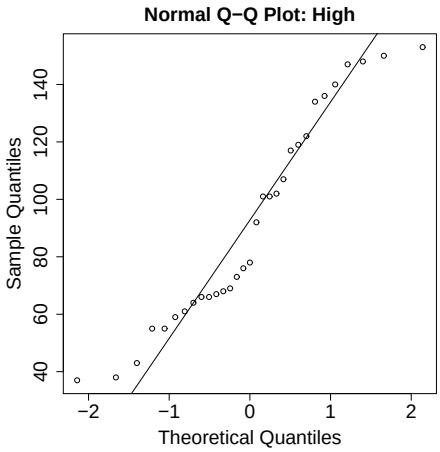
Figure 144: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



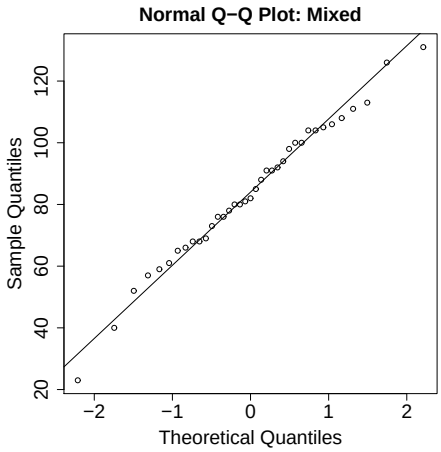
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9247784,$$
$$p = 0.03167116$$

(e) Shapiro-Wilk normality test for high-context culture groups.

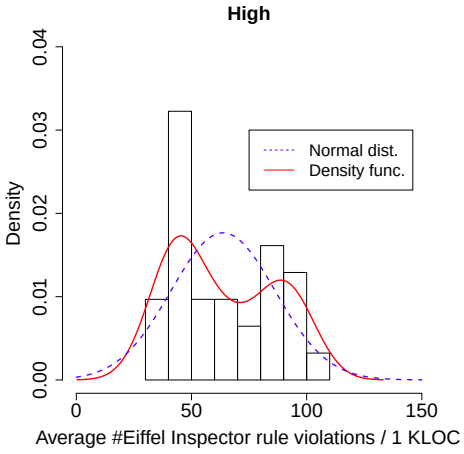


(d) QQ-Plot for mixed-context culture groups.

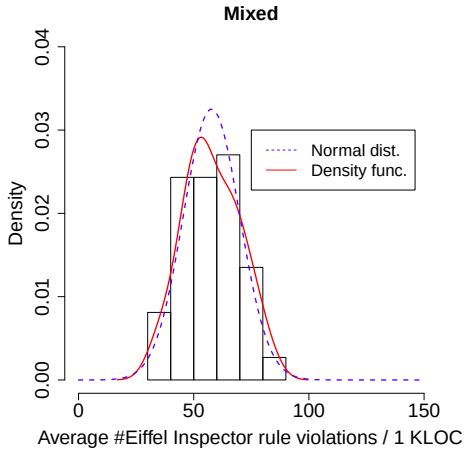
$$W = 0.9880314,$$
$$p = 0.9547597$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

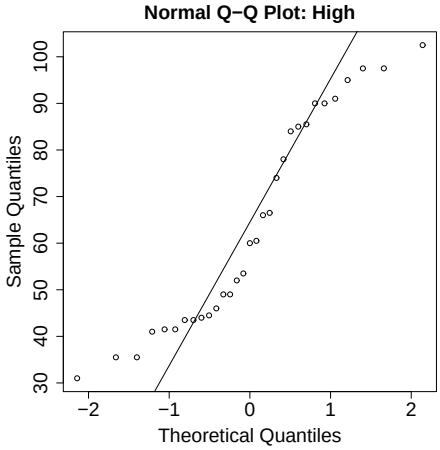
Figure 145: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



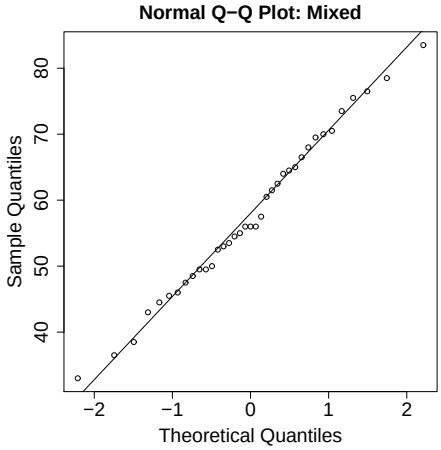
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9049574,$$
$$p = 0.009606407$$

(e) Shapiro-Wilk normality test for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

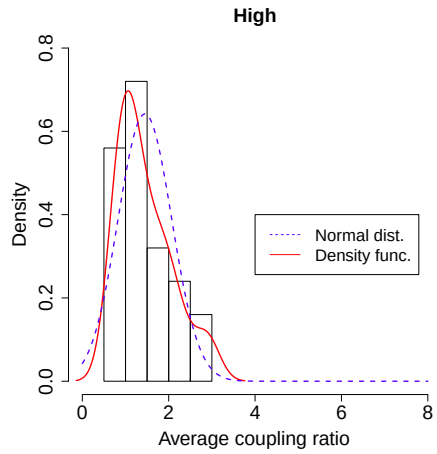
$$W = 0.9878832,$$
$$p = 0.9522527$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

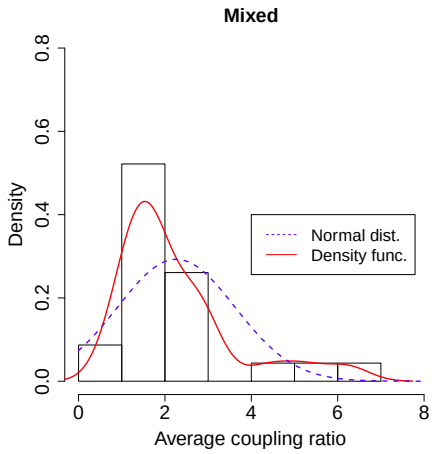
Figure 146: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.

B.3.2 Main analysis (2009 - 2012)

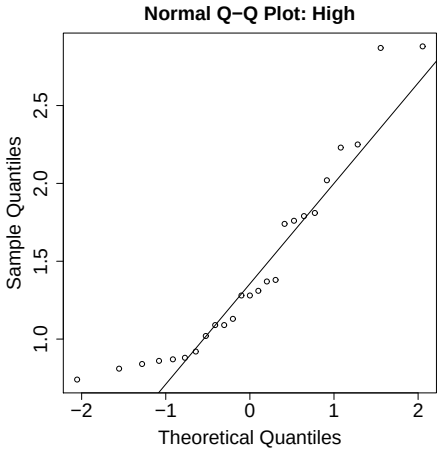
Figure 147 to 156 show the normality tests performed for the main analyis of research question RQ.3 using the projects from 2009 to 2012.



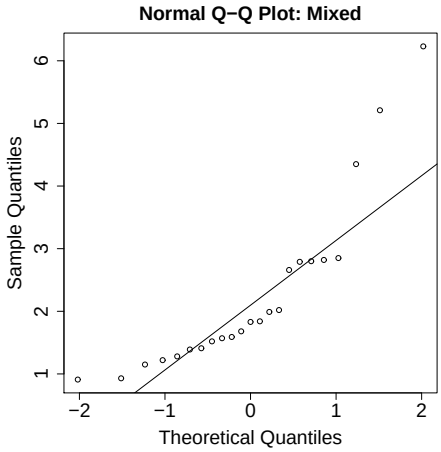
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

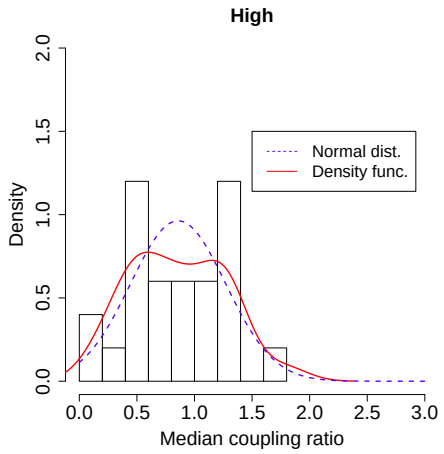
$$W = 0.8844678,$$
$$p = 0.008555478$$

(e) Shapiro-Wilk normality test for high-context culture groups.

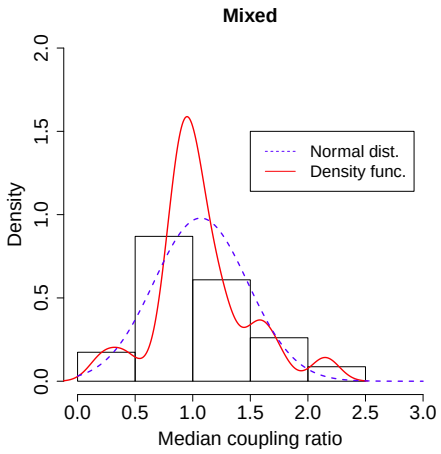
$$W = 0.8048286,$$
$$p = 0.0004582308$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

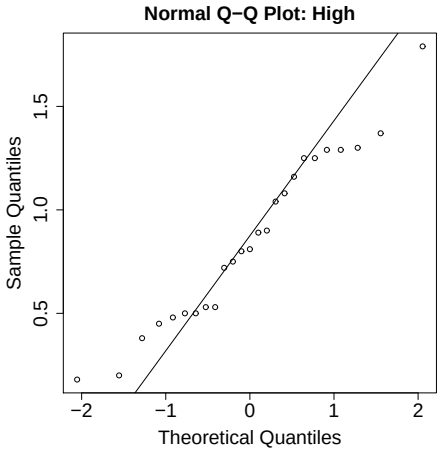
Figure 147: Normality tests for average coupling ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



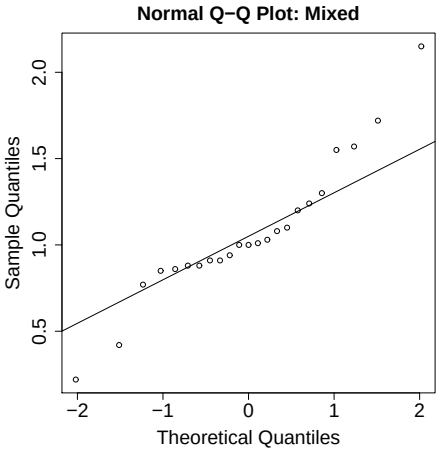
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

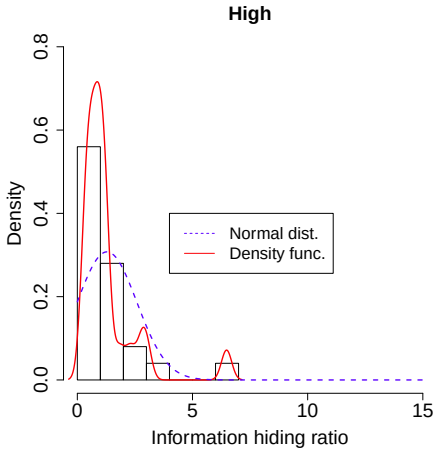
$$W = 0.9593799,$$
$$p = 0.4022613$$

(e) Shapiro-Wilk normality test for high-context culture groups.

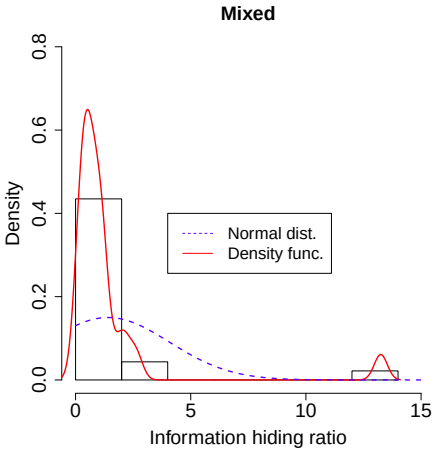
$$W = 0.9298052,$$
$$p = 0.10831$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

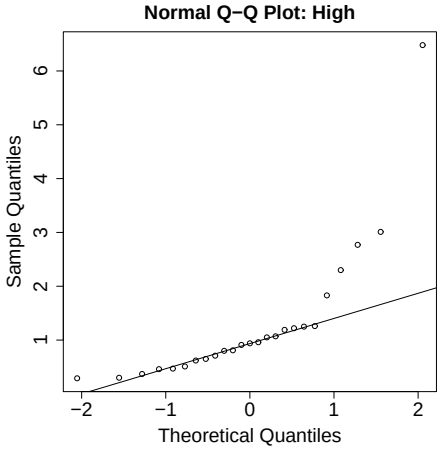
Figure 148: Normality tests for median coupling ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



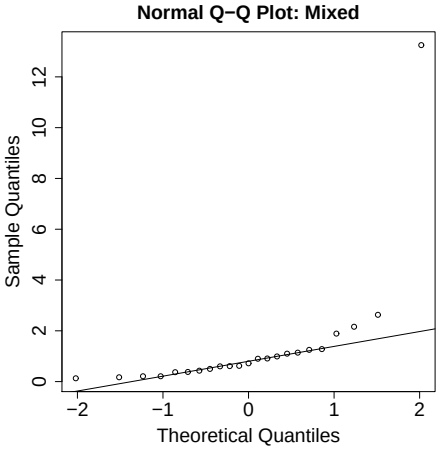
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

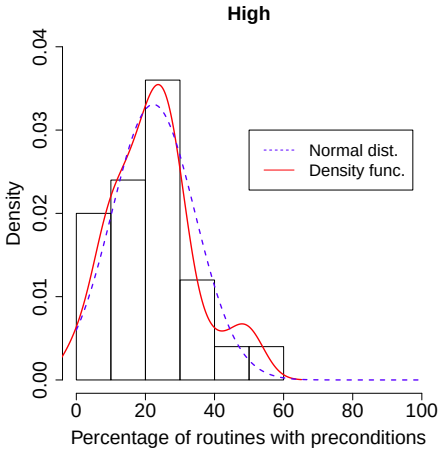
$$W = 0.6584753,$$
$$p = 2.112356\text{e-}06$$

(e) Shapiro-Wilk normality test for high-context culture groups.

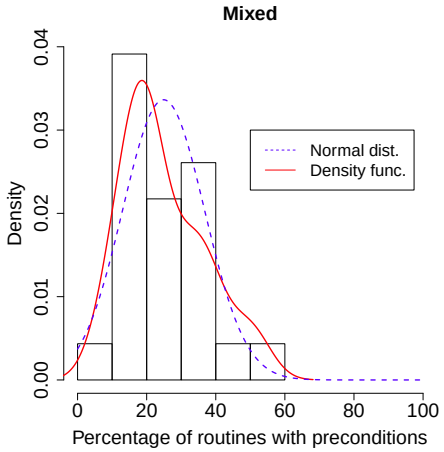
$$W = 0.4172333,$$
$$p = 1.517451\text{e-}08$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

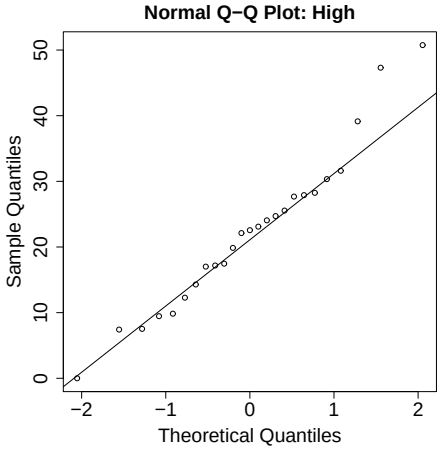
Figure 149: Normality tests for information hiding ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



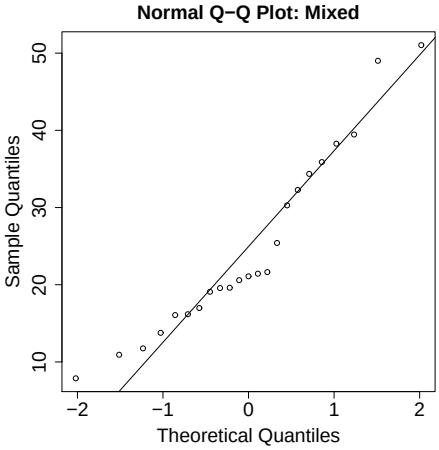
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9644528,$$
$$p = 0.5100571$$

(e) Shapiro-Wilk normality test for high-context culture groups.

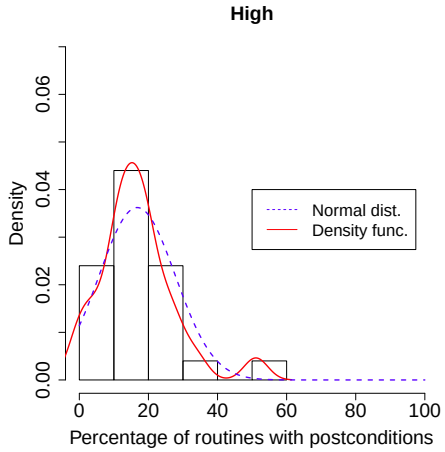


(d) QQ-Plot for mixed-context culture groups.

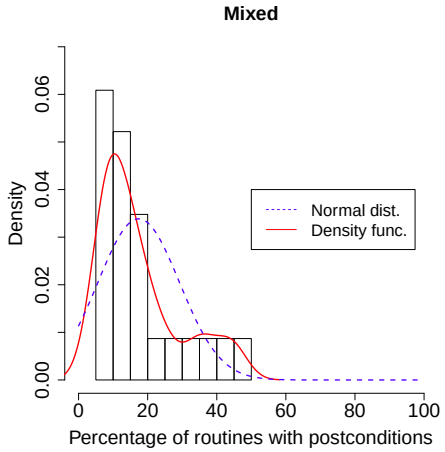
$$W = 0.9252787,$$
$$p = 0.08649837$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

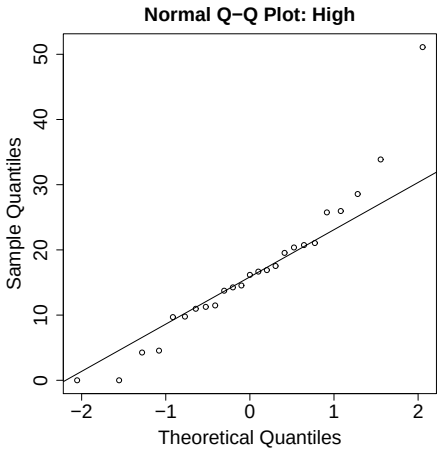
Figure 150: Normality tests for percentage of routines with preconditions of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



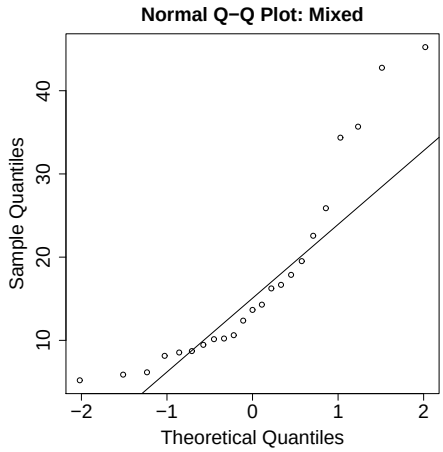
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9226232,$$
$$p = 0.0588086$$

(e) Shapiro-Wilk normality test for high-context culture groups.

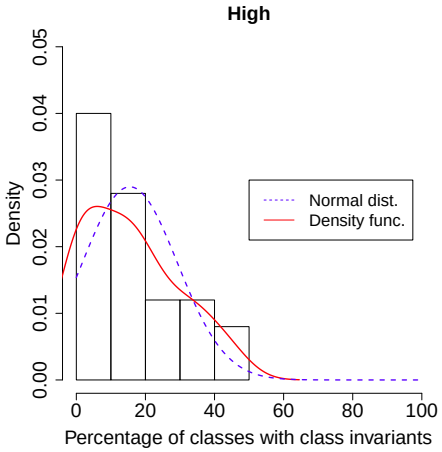


(d) QQ-Plot for mixed-context culture groups.

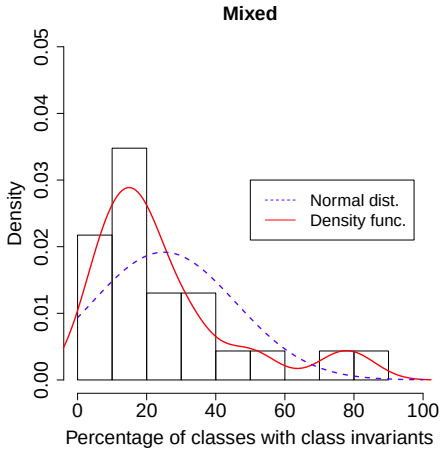
$$W = 0.8425767,$$
$$p = 0.002001541$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

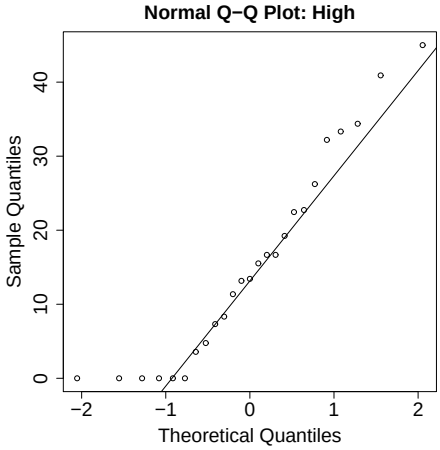
Figure 151: Normality tests for percentage of routines with postconditions of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



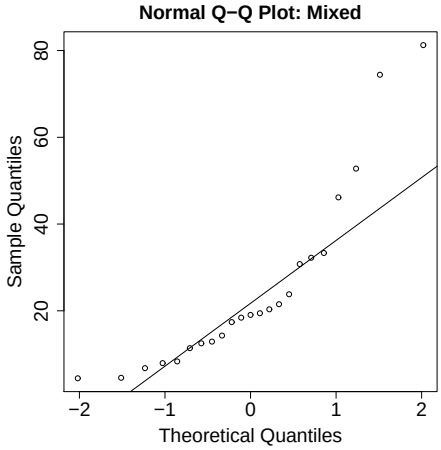
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9153825,$$
$$p = 0.04025624$$

(e) Shapiro-Wilk normality test for high-context culture groups.

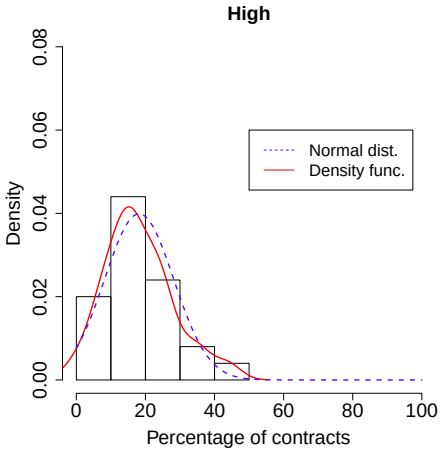


(d) QQ-Plot for mixed-context culture groups.

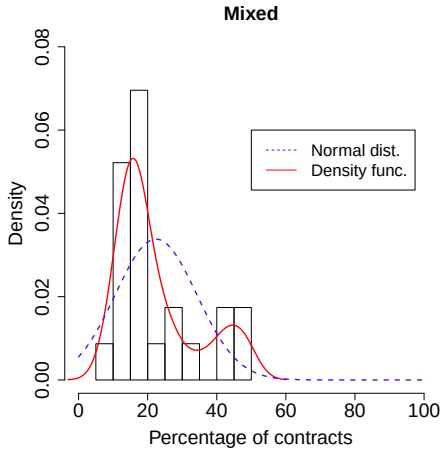
$$W = 0.815909,$$
$$p = 0.0006964003$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

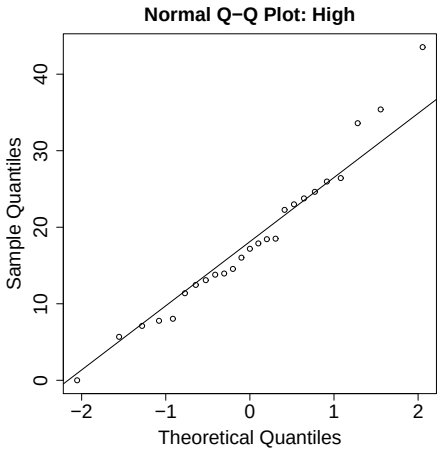
Figure 152: Normality tests for percentage of classes with class invariants of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



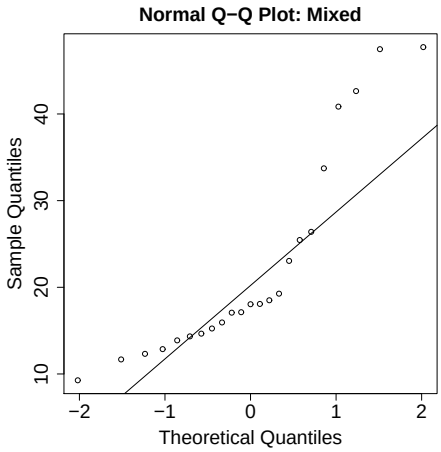
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9681826,$$
$$p = 0.5994068$$

(e) Shapiro-Wilk normality test for high-context culture groups.

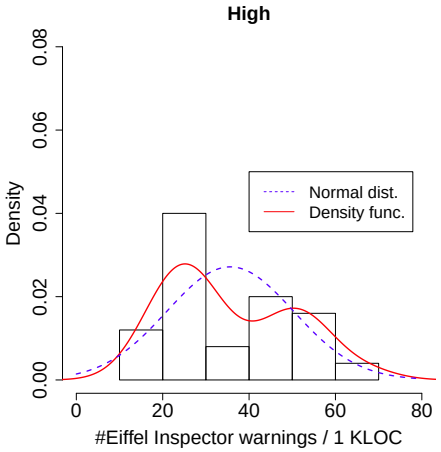


(d) QQ-Plot for mixed-context culture groups.

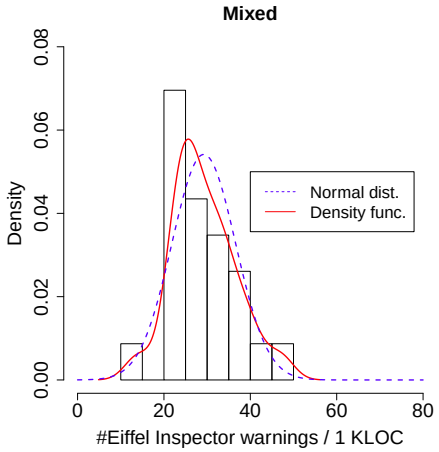
$$W = 0.8194473,$$
$$p = 0.0007978983$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

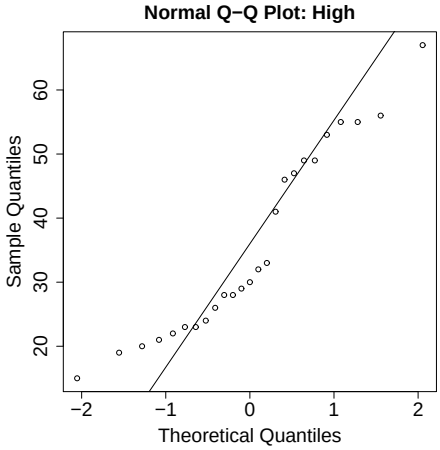
Figure 153: Normality tests for average percentage of contracts of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



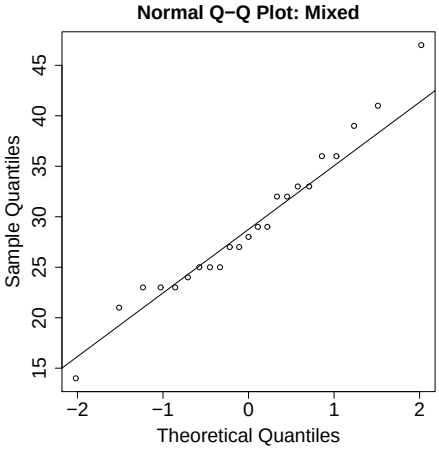
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9143764,$$
$$p = 0.03820688$$

(e) Shapiro-Wilk normality test for high-context culture groups.

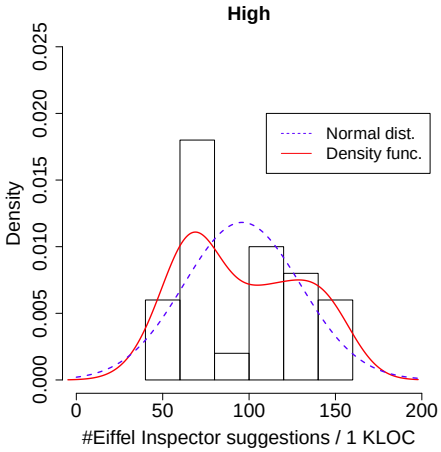


(d) QQ-Plot for mixed-context culture groups.

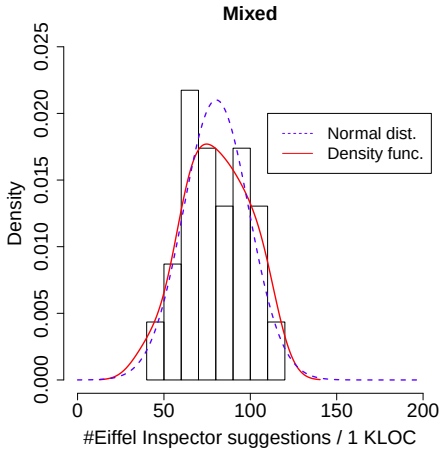
$$W = 0.9691859,$$
$$p = 0.6695199$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

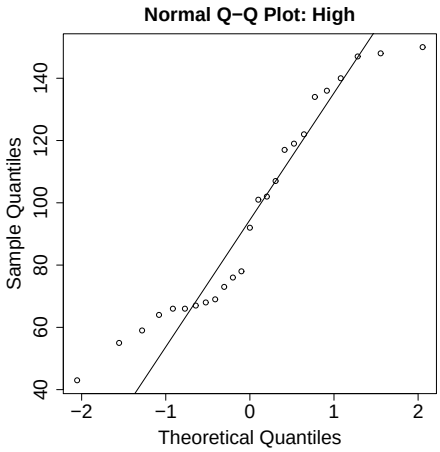
Figure 154: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



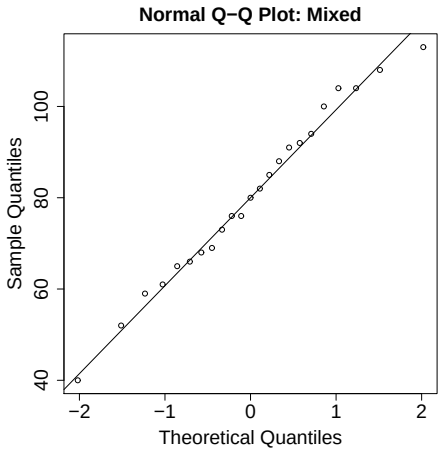
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.9149578,$$
$$p = 0.03937759$$

(e) Shapiro-Wilk normality test for high-context culture groups.

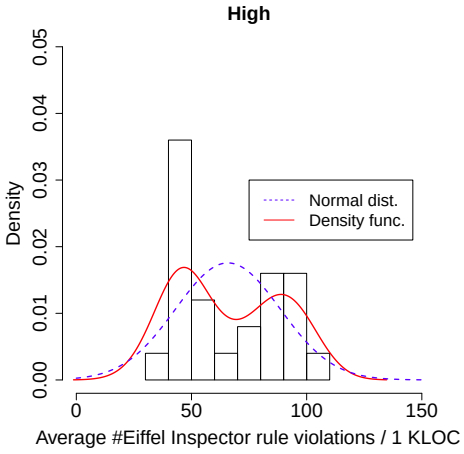


(d) QQ-Plot for mixed-context culture groups.

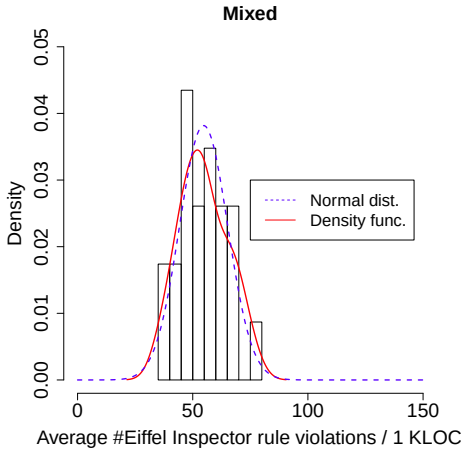
$$W = 0.9839119,$$
$$p = 0.9608059$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

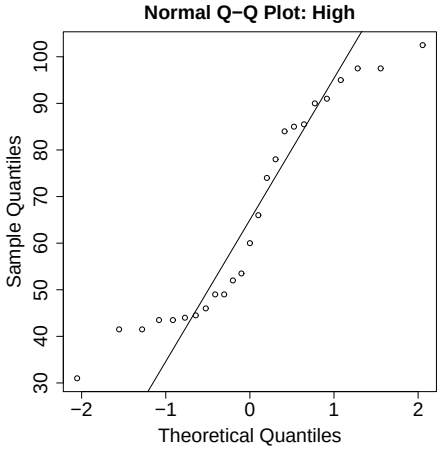
Figure 155: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



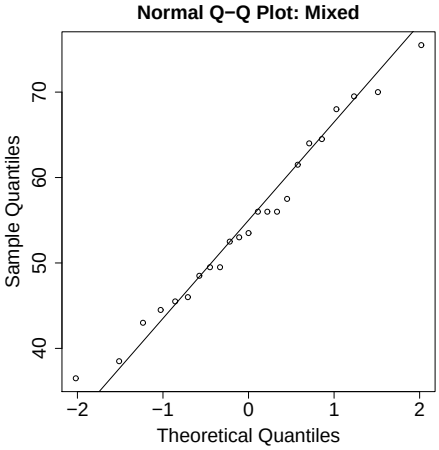
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.8945535,$$
$$p = 0.01398944$$

(e) Shapiro-Wilk normality test for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

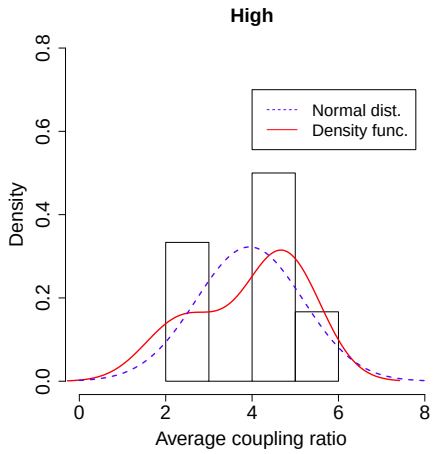
$$W = 0.9770158,$$
$$p = 0.8496011$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

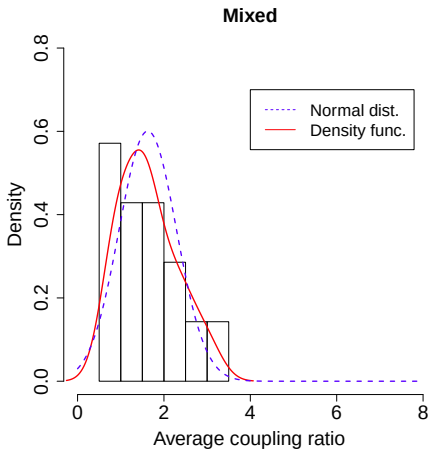
Figure 156: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.

B.3.3 Main analysis (2013 - 2014)

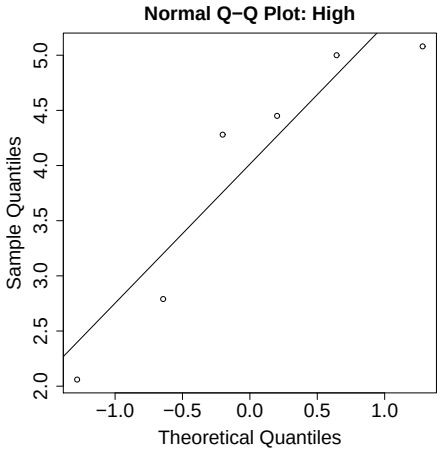
Figure 157 to 166 show the normality tests performed for the main analyis of research question RQ.3 using the projects from 2013 and 2014.



(a) Histogram for high-context culture groups.



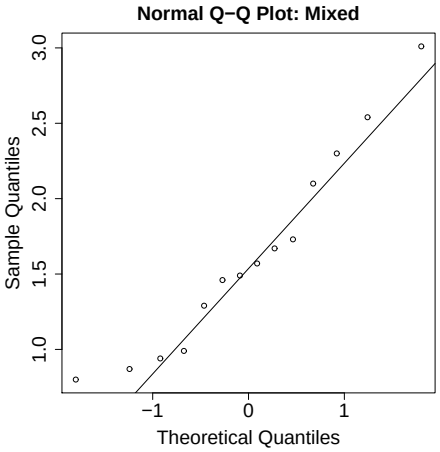
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.8675465,$$
$$p = 0.2165993$$

(e) Shapiro-Wilk normality test for high-context culture groups.

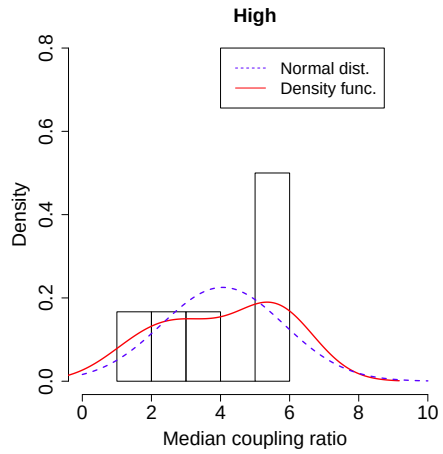


(d) QQ-Plot for mixed-context culture groups.

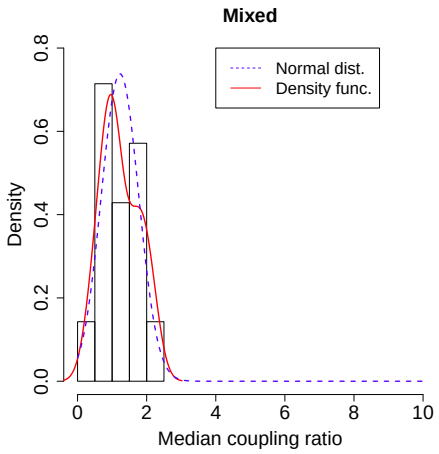
$$W = 0.9393494,$$
$$p = 0.4100507$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

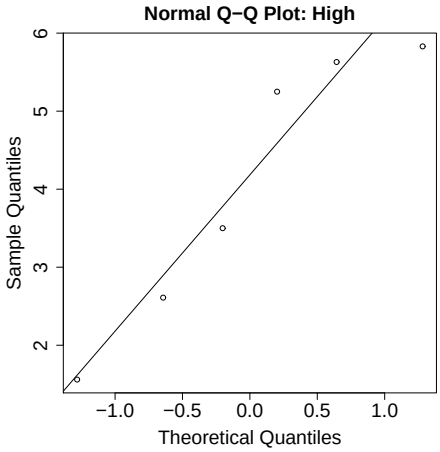
Figure 157: Normality tests for average coupling ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



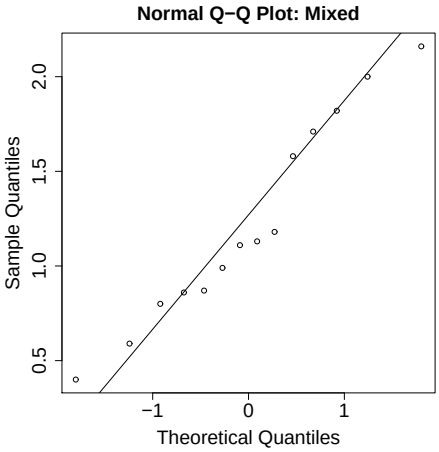
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

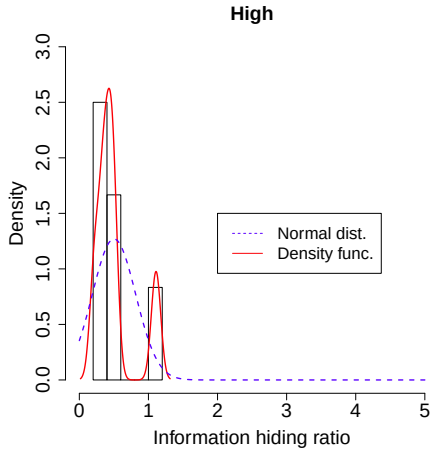
$$W = 0.895738,$$
$$p = 0.3493508$$

(e) Shapiro-Wilk normality test for high-context culture groups.

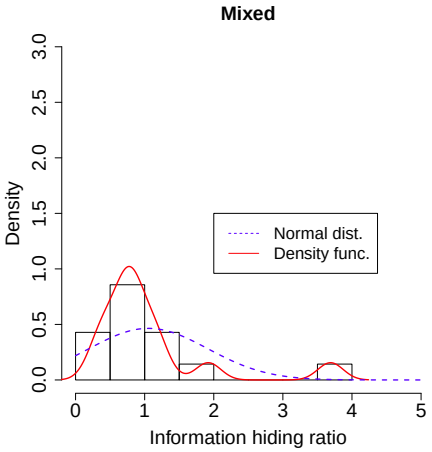
$$W = 0.9501225,$$
$$p = 0.562609$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

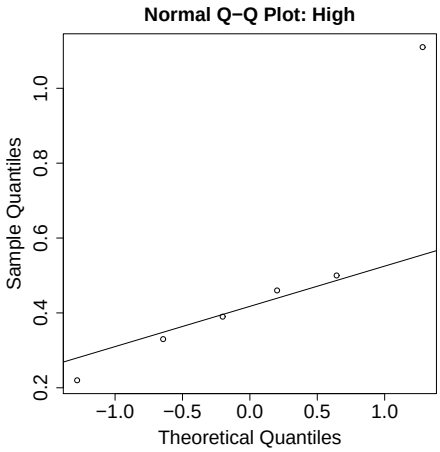
Figure 158: Normality tests for median coupling ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



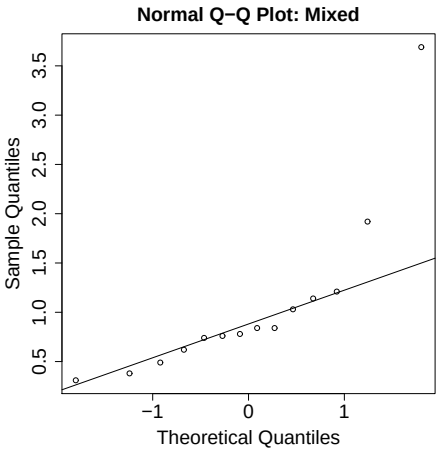
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

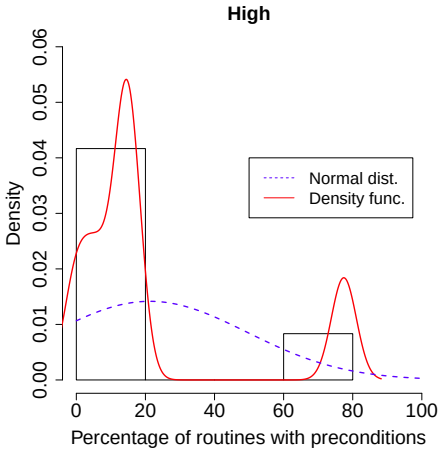
$$W = 0.7951096,$$
$$p = 0.05308719$$

(e) Shapiro-Wilk normality test for high-context culture groups.

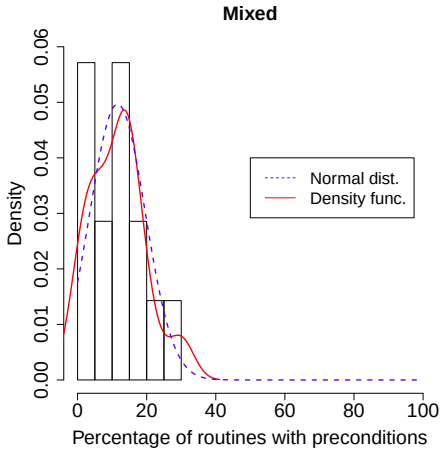
$$W = 0.7042539,$$
$$p = 0.0004090505$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

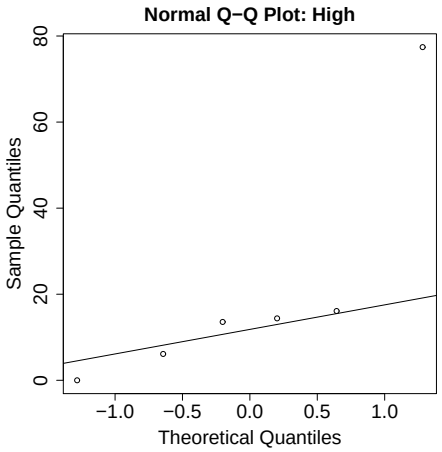
Figure 159: Normality tests for information hiding ratio of projects with only high-context culture groups and projects with mixed-context culture groups.



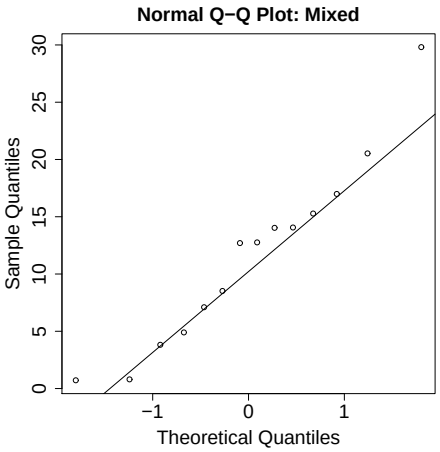
(a) Histogram for high-context culture groups.



(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

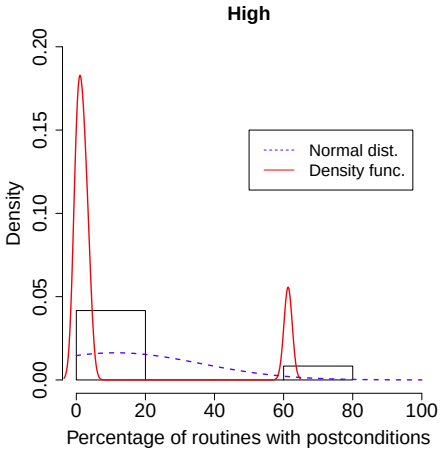
$$W = 0.6986431,$$
$$p = 0.005981248$$

(e) Shapiro-Wilk normality test for high-context culture groups.

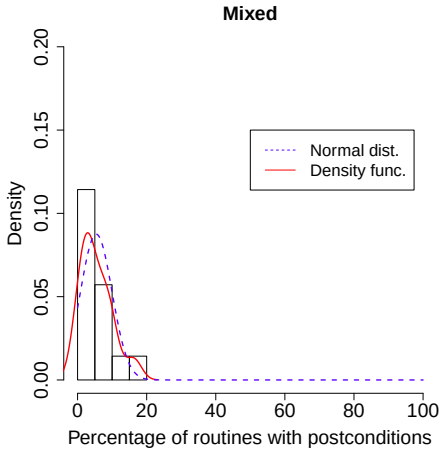
$$W = 0.9480409,$$
$$p = 0.5307245$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

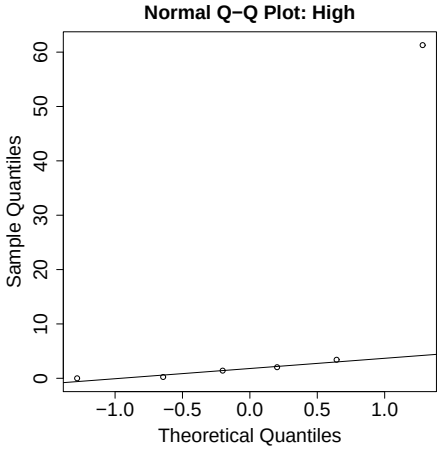
Figure 160: Normality tests for percentage of routines with preconditions of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



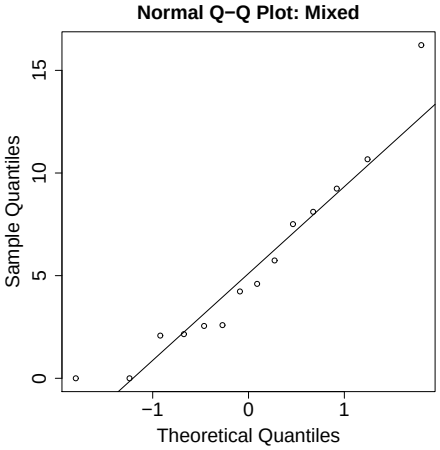
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.543741,$$
$$p = 9.046506\text{e-}05$$

(e) Shapiro-Wilk normality test for high-context culture groups.

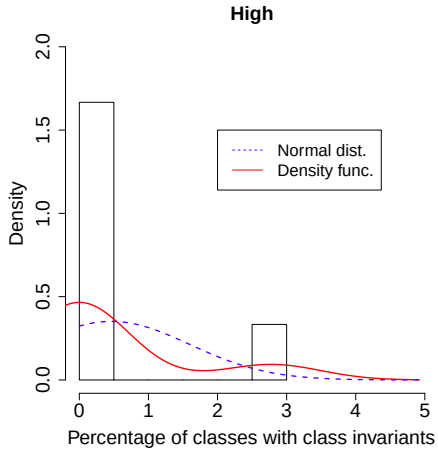


(d) QQ-Plot for mixed-context culture groups.

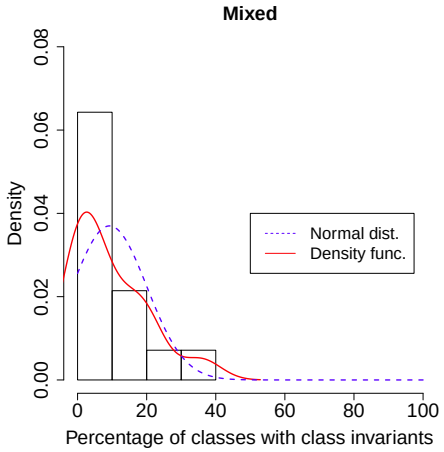
$$W = 0.9200526,$$
$$p = 0.2202782$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

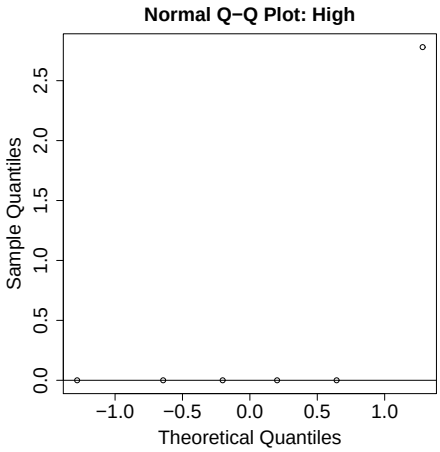
Figure 161: Normality tests for percentage of routines with postconditions of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



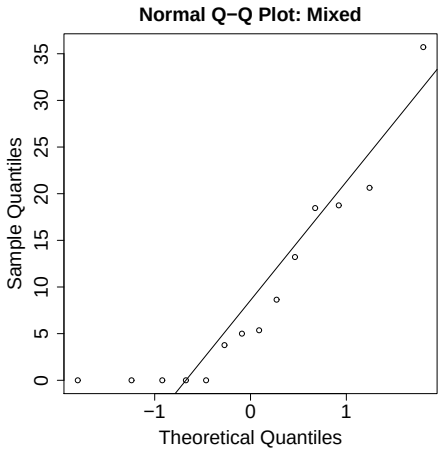
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.4960943,$$
$$p = 2.072914\text{e-}05$$

(e) Shapiro-Wilk normality test for high-context culture groups.

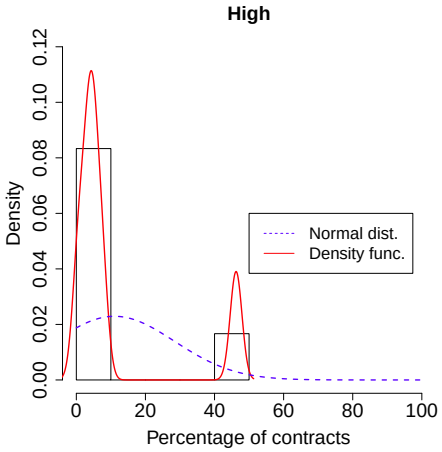


(d) QQ-Plot for mixed-context culture groups.

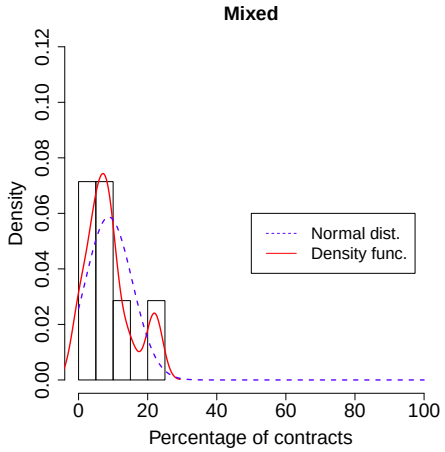
$$W = 0.8352436,$$
$$p = 0.01408785$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

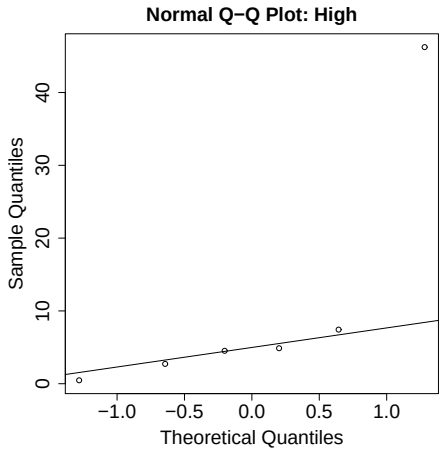
Figure 162: Normality tests for percentage of classes with class invariants of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



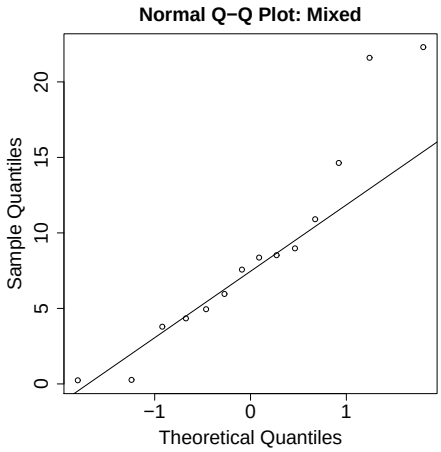
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.6260735,$$
$$p = 0.0009352391$$

(e) Shapiro-Wilk normality test for high-context culture groups.

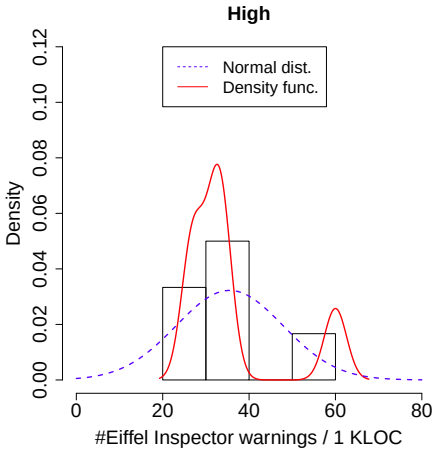


(d) QQ-Plot for mixed-context culture groups.

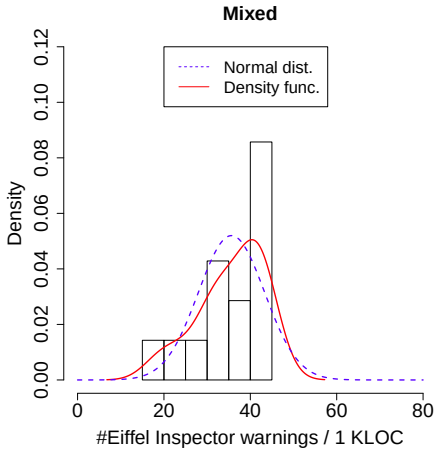
$$W = 0.8980282,$$
$$p = 0.1055957$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

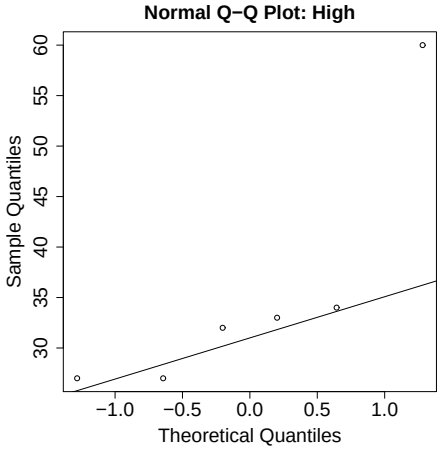
Figure 163: Normality tests for average percentage of contracts of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



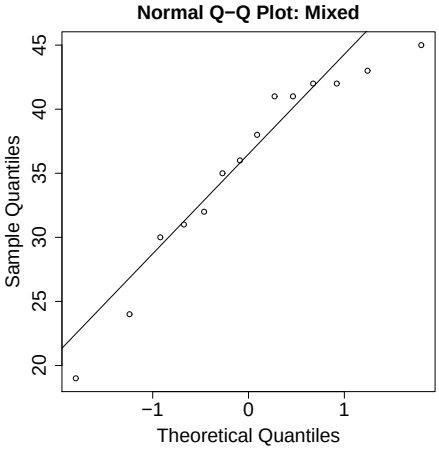
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.70745,$$
$$p = 0.007399402$$

(e) Shapiro-Wilk normality test for high-context culture groups.

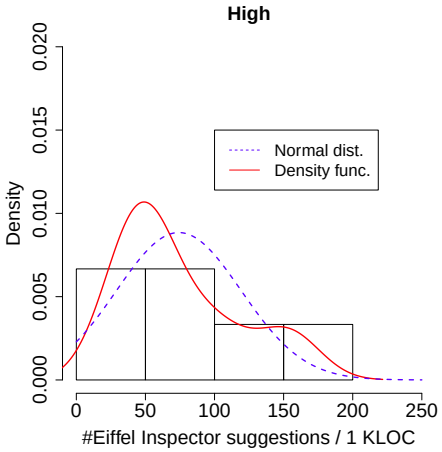


(d) QQ-Plot for mixed-context culture groups.

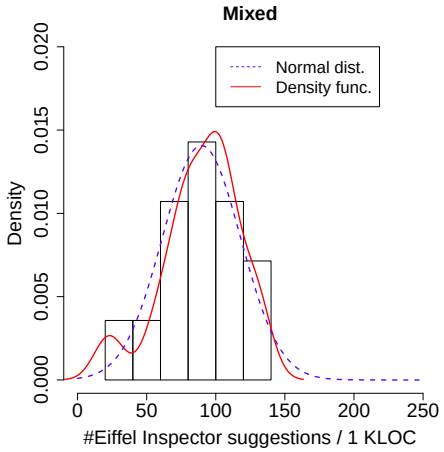
$$W = 0.9171671,$$
$$p = 0.2001578$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

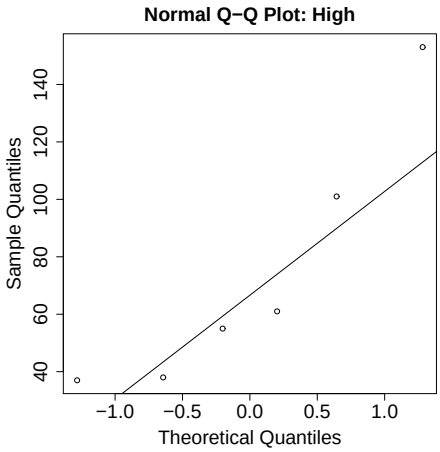
Figure 164: Normality tests for number of Eiffel Inspector warnings per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



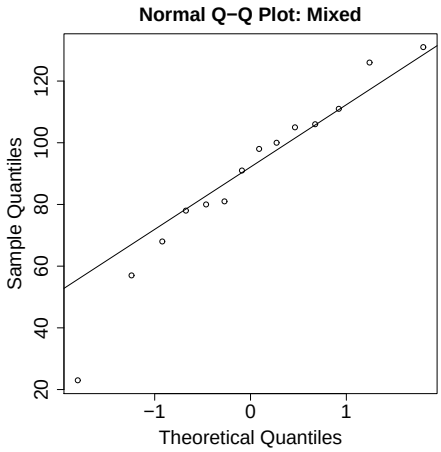
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.8472019,$$
$$p = 0.149388$$

(e) Shapiro-Wilk normality test for high-context culture groups.

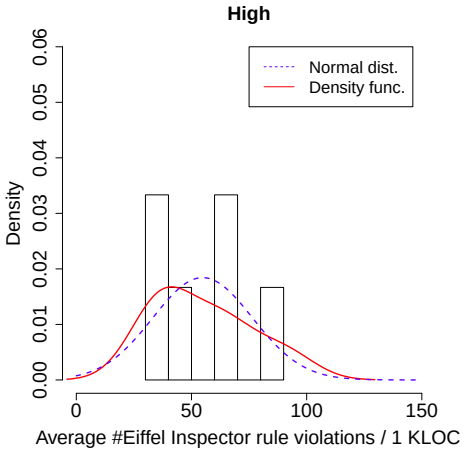


(d) QQ-Plot for mixed-context culture groups.

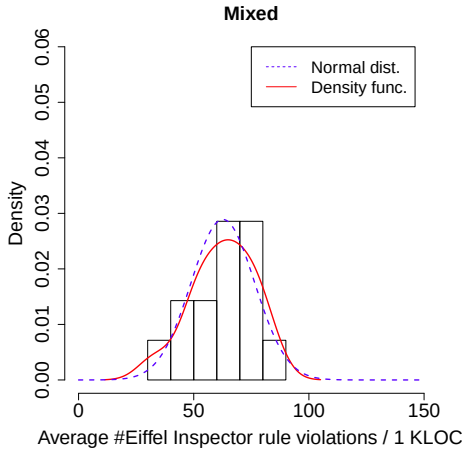
$$W = 0.954644,$$
$$p = 0.6348281$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

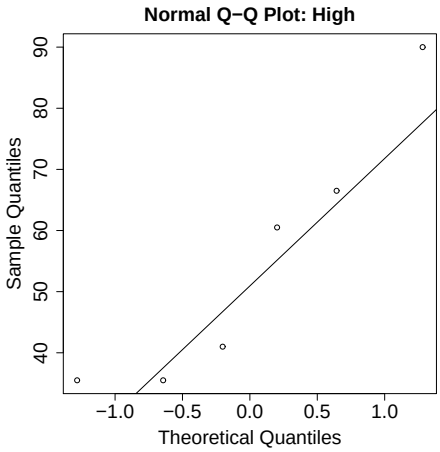
Figure 165: Normality tests for number of Eiffel Inspector suggestions per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.



(a) Histogram for high-context culture groups.



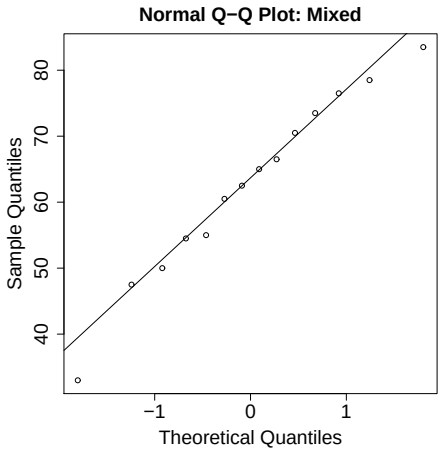
(b) Histogram for mixed-context culture groups.



(c) QQ-Plot for high-context culture groups.

$$W = 0.8819149,$$
$$p = 0.2779664$$

(e) Shapiro-Wilk normality test for high-context culture groups.



(d) QQ-Plot for mixed-context culture groups.

$$W = 0.976214,$$
$$p = 0.9468847$$

(f) Shapiro-Wilk normality test for mixed-context culture groups.

Figure 166: Normality tests for average number of Eiffel Inspector rule violations per 1000 LOC of projects with only high-context culture groups and projects with mixed-context culture groups.

References

[1] IEEE recommended practice for software requirements specifications. *IEEE Std 830-1998*, pages 1–40, Oct 1998.

[2] T. J. Allen. *Managing the Flow of Technology.* MIT Press, Cambridge, MA, 2nd edition, 1997.

[3] Andrea Arcuri and Lionel Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 1–10, New York, NY, USA, 2011. ACM.

[4] Deborah J. Barrett. *Leadership communication.* McGraw-Hill/Irwin, New York, NY, USA, 2nd edition, 2008.

[5] Victor R. Basili and Barry T. Perricone. Software errors and complexity: An empirical investigation0. *Commun. ACM*, 27(1):42–52, January 1984.

[6] R.D. Battin, R. Crocker, J. Kreidler, and K. Subramanian. Leveraging resources in global software development. *Software, IEEE*, 18(2):70–77, Mar 2001.

[7] T. E. Bell and T. A. Thayer. Software requirements: Are they really a problem? In *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE '76, pages 61–68, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[8] Christian Bird, Nachiappan Nagappan, Premkumar Devanbu, Harald Gall, and Brendan Murphy. Does distributed development affect software quality? an empirical case study of windows vista. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 518–528, Washington, DC, USA, 2009. IEEE Computer Society.

[9] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE '76, pages 592–605, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[10] B.W. Boehm, J.R. Brown, H. Kaspar, M. Lipow, G.J. MacLeod, and M.J. Merritt. *Characteristics of Software Quality.* North-Holland Publishing Company, 1978.

[11] I. Bosnic, I. Cavrak, M. Zagar, R. Land, and I. Crnkovic´. Customers' role in teaching distributed software development. In *Software Engineering Education and Training (CSEE T), 2010 23rd IEEE Conference on*, pages 73–80, March 2010.

[12] B. Bruegge, Allen H. Dutoit, R. Kobylinski, and G. Teubner. Transatlantic project courses in a university environment. In *Software Engineering Conference, 2000. APSEC 2000. Proceedings. Seventh Asia-Pacific*, pages 30–37, 2000.

[13] E. Carmel and R. Agarwal. Tactical approaches for alleviating distance in global software development. *Software, IEEE*, 18(2):22–29, Mar 2001.

[14] Marcelo Cataldo and Sangeeth Nambiar. On the relationship between process maturity and geographic distribution: An empirical analysis of their impact on software quality. In *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ESEC/FSE '09, pages 101–110, New York, NY, USA, 2009. ACM.

[15] Joseph P. Cavano and James A. McCall. A framework for the measurement of software quality. In *Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues*, pages 133–139, New York, NY, USA, 1978. ACM.

[16] S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493, Jun 1994.

[17] Lennie Copeland and Lewis Griggs. *Going International: How to Make Friends and Deal Effectively in the Global Marketplace*. 1986.

[18] I. Crnkovic, I. Bosnic´, and M. Zagar. Ten tips to succeed in global software engineering education. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 1225–1234, June 2012.

[19] Daniela Damian, Allyson Hadwin, and Ban Al-Ani. Instructional design and assessment strategies for teaching global software development: A framework. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 685–690, New York, NY, USA, 2006. ACM.

[20] J.A. Espinosa, Ning Nan, and E. Carmel. Do gradations of time zone separation make a difference in performance? a first laboratory study.

In *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*, pages 12–22, Aug 2007.

[21] H.-Christian Estler, Martin Nordio, Carlo A. Furia, and Bertrand Meyer. Collaborative debugging. In *8th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2013.

[22] H.-Christian Estler, Martin Nordio, Carlo A. Furia, and Bertrand Meyer. Unifying configuration management with awareness systems and merge conflict detection. In *22nd Australasian Software Engineering Conference (ASWEC)*. IEEE, 2013.

[23] H.-Christian Estler, Martin Nordio, Carlo A. Furia, and Bertrand Meyer. Awareness and merge conflicts in distributed software development. In Yuanfang Cai, Jude Fernandez, and Wenyun Zhao, editors, *Proceedings of the 9th International Conference on Global Software Engineering (ICGSE)*, pages 26–35. IEEE Computer Society, August 2014. Best paper award.

[24] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[25] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[26] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 2002.

[27] Carlo Ghezzi and Dino Mandrioli. The challenges of software engineering education. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 637–638, New York, NY, USA, 2005. ACM.

[28] Peter Gloor, Maria Paasivaara, Casper Lassenius, Detlef Schoder, Kai Fischbach, and Christine Miller. Teaching a global project course: Experiences and lessons learned. In *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*, CTGDSD '11, pages 1–5, New York, NY, USA, 2011. ACM.

[29] Olly Gotel, Vidya Kulkarni, Long Chrea Neak, Christelle Scharff, and Sopheap Seng. Introducing global supply chains into software engineering education. In *Proceedings of the 1st International Conference on Software Engineering Approaches for Offshore and Outsourced Development*, SEAFOOD'07, pages 44–58, Berlin, Heidelberg, 2007. Springer-Verlag.

[30] Olly Gotel, Vidya Kulkarni, Christelle Scharff, and Longchrea Neak. Students as partners and students as mentors: An educational model for quality assurance in global software development. In Kay Berkling, Mathai Joseph, Bertrand Meyer, and Martin Nordio, editors, *Software Engineering Approaches for Offshore and Outsourced Development*, volume 16 of *Lecture Notes in Business Information Processing*, pages 90–106. Springer Berlin Heidelberg, 2009.

[31] E. T. Hall. *Beyond culture*. Anchor Press, Garden City, NY, 1976.

[32] M.J. Hawthorne and D.E. Perry. Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 643–644, May 2005.

[33] J.D. Herbsleb and D. Moitra. Global software development. *Software, IEEE*, 18(2):16–20, Mar 2001.

[34] H. Holmstrom, E.O. Conchuir, P.J. Agerfalk, and B. Fitzgerald. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. In *Global Software Engineering, 2006. ICGSE '06. International Conference on*, pages 3–11, Oct 2006.

[35] R.R. Lutz. Analyzing software requirements errors in safety-critical, embedded systems. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 126–133, Jan 1993.

[36] J.A. McCall, P.K. Richards, and G.F. Walters. *Factors in Software Quality. Volume I. Concepts and Definitions of Software Quality*. 1977.

[37] Marshall Kirk McKusick and George V. Neville-Neil. *The Design and Implementation of the FreeBSD Operating System*. Pearson Education, 2004.

[38] B. Meyer. The unspoken revolution in software engineering. *Computer*, 39(1):124, 121–123, Jan 2006.

[39] Bertrand Meyer. Applying "design by contract". *Computer*, 25(10):40–51, October 1992.

[40] Bertrand Meyer. *Eiffel: The Language*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[41] Bertrand Meyer. *Object-oriented Software Construction (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[42] Bertrand Meyer. *Touch of Class: Learning to Program Well with Objects and Contracts*. Springer Publishing Company, Incorporated, 1 edition, 2009.

[43] A. Mockus and J. Herbsleb. Challenges of global software development. In *Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*, pages 182–184, 2001.

[44] Martin Nordio, Christian Estler, Bertrand Meyer, Nazareno Aguirre, Elisabetta Di Nitto, Rafael Prikladnicki, and Anthony Savidis. An experiment on teaching coordination in a globally distributed software engineering class. In *Proceedings, 27th Conference on Software Engineering Education and Training (CSEE&T 2014)*, 2014.

[45] Martin Nordio, H.-Christian Estler, Bertrand Meyer, Julian Tschannen, Carlo Ghezzi, and Elisabetta Di Nitto. How do distribution and time zones affect software development? a case study on communication. In *6th International Conference on Global Software Engineering*. IEEE, 2011.

[46] Martin Nordio, Carlo Ghezzi, Bertrand Meyer, Elisabetta Di Nitto, Giordano Tamburrelli, Julian Tschannen, Nazareno Aguirre, and Vidya Kulkarni. Teaching software engineering using globally distributed projects: the dose course. In *Collaborative Teaching of Globally Distributed Software Development - Community Building Workshop (CTGDSD)*. ACM, 2011.

[47] Martin Nordio, Roman Mitin, and Bertrand Meyer. Advanced hands-on training for distributed and outsourced software engineering. In *ICSE '10 Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, 2010.

[48] Martin Nordio, Roman Mitin, Bertrand Meyer, Carlo Ghezzi, Elisabetta Di Nitto, and Giordano Tamburelli. The Role of Contracts in

Distributed Development. In O. Gotel, M. Joseph, and B. Meyer, editors, *Software Engineering Advances For Offshore and Outsourced Development*, volume 35 of *Lecture Notes in Business and Information Processing*, pages 117–119, 2009.

[49] Rosalie Ocker, Mary Beth Rosson, Dana Kracaw, and S. Roxanne Hiltz. Training students to work effectively in partially distributed teams. *Trans. Comput. Educ.*, 9(1):6:1–6:24, March 2009.

[50] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058, December 1972.

[51] S.L. Pfleeger and J.M. Atlee. *Software Engineering: Theory and Practice*. Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition, 2006.

[52] Roger Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, USA, 6 edition, 2005.

[53] John A. Rice. *Mathematical statistics and data analysis*. 3rd edition, 2007.

[54] Ita Richardson, Allen E. Milewski, Neel Mullick, and Patrick Keil. Distributed development: An education perspective on the global studio project. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 679–684, New York, NY, USA, 2006. ACM.

[55] Paul Rook (Editor). *Software Reliability Handbook*. Elsevier Science Inc., New York, NY, USA, 1990.

[56] Christelle Scharff. An evolving collaborative model of working in students' global software development projects. In *Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development*, CTGDSD '11, pages 11–15, New York, NY, USA, 2011. ACM.

[57] Diomidis Spinellis. Global software development in the freebsd project. In *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner*, GSD '06, pages 73–79, New York, NY, USA, 2006. ACM.

[58] Eleni Stroulia, Ken Bauer, Michelle Craig, Karen Reid, and Greg Wilson. Teaching distributed software engineering with ucosp: The undergraduate capstone open-source project. In *Proceedings of the 2011 Com-*

munity Building Workshop on Collaborative Teaching of Globally Distributed Software Development, CTGDSD '11, pages 20–25, New York, NY, USA, 2011. ACM.

[59] Marco Trudel, Carlo A. Furia, Martin Nordio, and Bertrand Meyer. Really automatic scalable object-oriented reengineering. In *European Conference on Object-Oriented Programming (ECOOP)*. Springer, 2013.

[60] Stefan Zurfluh. Rule-based code analysis, 2014.