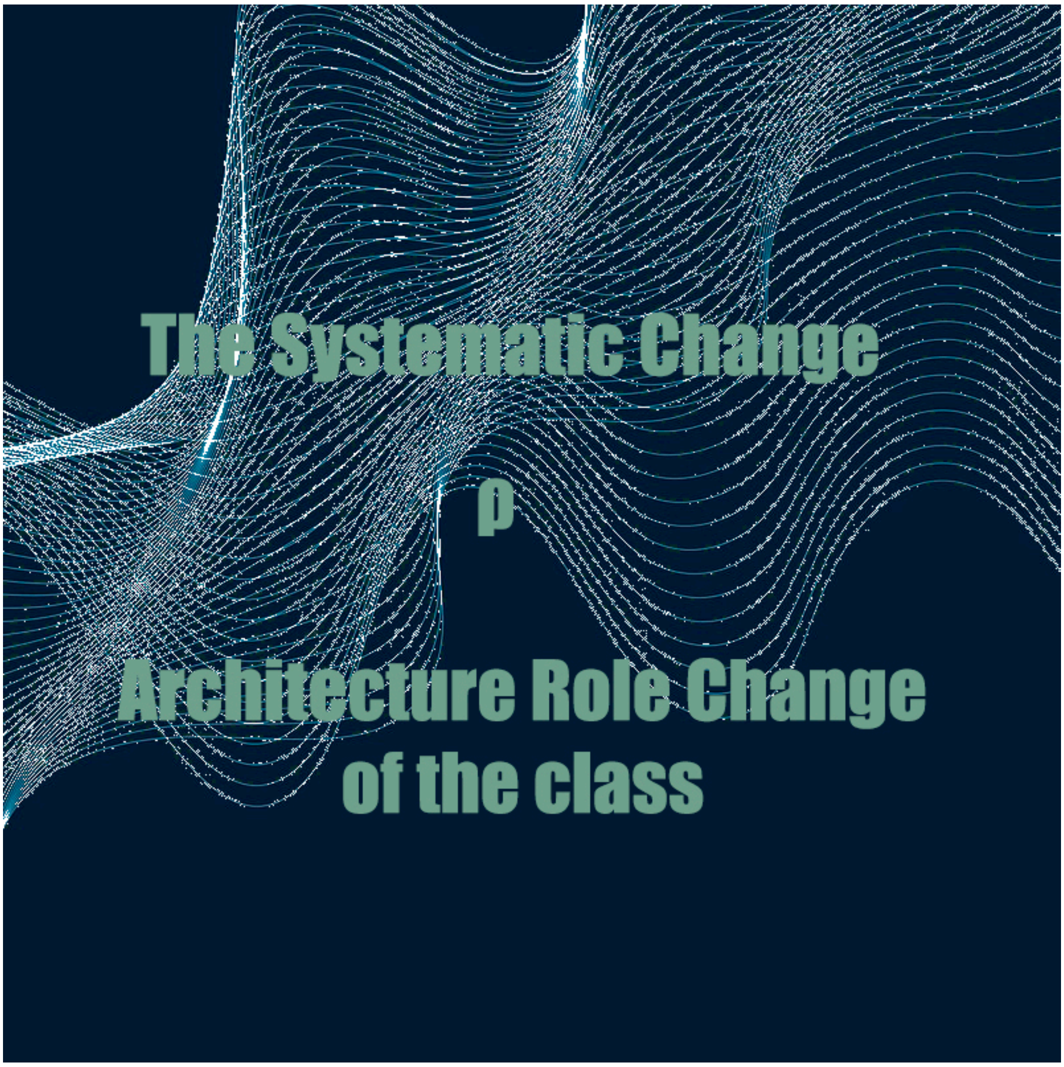
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



The relationship between "issues" with a specific label and Architecture role change of class in the Github’s repositories
—- An Empirical Case Study

Bachelor of Science Thesis in Software Engineering and Management

Wei Li

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

The relationship between "issues"with a specific labels in the Github repository and Architecture role change of the class
—An empirical case study

Supervisor: Ho-Quan Truong
Examiner: Richard Berntsson Svensson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
The image of correlation between "Systematic change " and " Architecture role change of class"

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

The relationship between the“issues” with a specific label and Architecture role change of the class in Github's repositories

----*An empirical case study*

Software engineering and management program

Department of Computer Science and Engineering

University of Gothenburg

Wei Li

Supervisor: Truong Ho-Quan

Abstract-- In Github repositories, "issues" are used as a mechanism to trace the program enhancements, bugs, architecture. Besides, developers can share their opinions and discuss the problems with the user community in the " issues section" in every Git repository, such as bug fixes, feature additions. In addition, many types of changes occur in software systems every day. However, Not all of them have an impact at the architectural level. In other words, not all types of changes affect architecture role change in class. Moreover, we learned the taxonomy of the class role stereotypes from Rebecca Wirfs-Brocks. Besides, We(the researchers) inherited the evolution of class role changes over k9-mail, a Github repository, from the paper written by Fröding and Nguyen-Ngoc. [4]. Nevertheless, The problems still remain! Whether or not "closed issues" with a specific label are associated with systematic changes? Are they further associated with class role change? Does systematic change reflected by bug-fix issues subject to any particular type of class role change? Which one?

Due to no prior studies tackling those problems, This study aims to provide empirical evidence for the correlation between systematic code change and architecture role change through a case study on K9-mail. Furthermore, we extracted the data from the "issues" with a specific label from k9-mail. Then, run the "correlation test" by "R," a statistical tool, between several pairs of entities to discover the correlations amid them.

The results show robust positive correlations between the above-investigated entities in the repository. Further, we extend our discussion over those problems in a broader context after the results presented, including validity threats. This paper contributes to unveiling the relationships between those entities through a case study Also helps software developers and software testers with software understanding, software improvement, software prediction, and software maintenance. However, Due to the single case study, the generalization of the results into common scenarios is limited.

Keywords--issues, closed issues, Labels, types of code change, architecture role change of class, class role change.

I.Introduction

This study aims to find empirical proof on the hypothesis that the "bug-fix issues" [1]with a specific label from Github's repositories are

[1] The definition of "issue": https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues#integrated-with-github

correlated with "architecture role change of the class" over selected releases. Furthermore, we believe clarifying the "correlation" is vital. It will help us improve software prediction and software understanding: Under what circumstance, we carry out the "refactoring and restructuring" activities. As [1][2] revealed, early refactoring is better than later reengineering. Refactoring can slow down program degeneration, avoid reengineering, avoid architectural degeneration, reduce program complexity, lower maintenance costs, and enhance software quality. We observe architecture change via the "class role change" to see whether it will induce bug-fix commits through the refactoring.

In this study, The "issues," which are labeled as "bug" and introduce the "bug-fix commits," are regarded as 'bug-fix issues."Also, a quantifiable representation of "systematic change." (figure 1)

Furthermore, To discover the solid evidence on the hypothesis in mind, We quantify/observe the architectural change via changes in the role/responsibility of classes over releases. We observe the code changes related to bug-fix by considering the "issues" marked with bug-fix in selected releases. (figure 1) Thus, We can measure the correlation between architecture change and code change in a quantifiable manner.

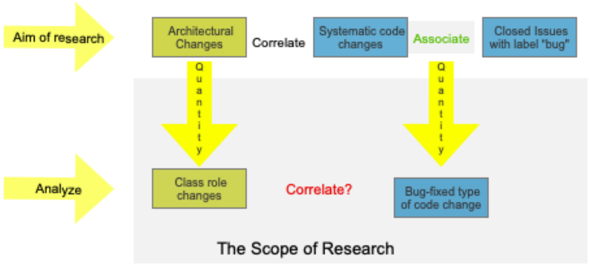


Figure 1: The aim of the research

Many changes are happening every day in software system.[2, p. 33] The changes might have different purposes, scopes, and impacts on the system. For instance, "type *of change": the perfective modifications* resulting from changed or new requirements improve system performance; *The corrective change* happens in response to defects;[2, p. 33]. *Adaptive change occurs* when the program moves to a new environment or accommodates a new standard.[2, p. 33] Moreover, The changes can also be classified by the differential impacts on the system architecture. For instance, the "*cosmetic change"*(trivial change) includes class name, method name, and comments. These mentioned above don't affect the structure and semantics of the system. Others may lead to "significant impacts" on the design, namely. "*global change*," such as code *refactoring*, [2, p. 38]"the refactoring is a process of modifying a program to improve its structure, to reduce its complexity, to make it easier to understand."[1, p. 250] However, not all of them are "*systematic changes,"* Besides "bug-fix commits" induced by "refactoring."

On the other hand, knowing "architecture changes" are crucially important in understanding "class role stereotypes" and their changes. As [5] mentioned, There are six class role stereotypes in total. In this study, the "class role change" refers to the object/class shifting from one class role stereotype into another, Implying responsibility alteration between them.[4][5] Besides, the" architecture change," the abstract representation of the"class role change," (figure 1)refers to the deletion/addition of software modules and connections. [2]. Further, It impacts the system architecture. "the architecture changes include *refactoring* and *restructuring* the system architecture to improve the software quality."[2, p. 38] According to [3], " The conclusion from the study verifies the findings from M. Di. Penta et al.: The commits implementing refactoring actions have higher odds to induce bug-fix in the system." Therefore, the association between "architecture changes" reflected by "class role change" and "systematic changes" reflected by "bug-fix commits" get initial theoretical proof but require further investigation due to no empirical evidence found so far.

Even though the existing works of literature provide some insights and knowledge foundation for our research, the gaps remain! There is a lack of empirical evidence for the correlation/relationship between code change and change at an architectural level.

In this study, we want to draw special attention to the change at the architectural level. Meaning, to discover whether systematic code change correlates with architectural role changes in class? To fill the knowledge gap, we will conduct an empirical case study on a chosen GitHub repository, k9-mail. Further, examine all bug-fix issues labeled as "bug" over selected releases in the repository to explore the associations between those mentioned entities. We collected data related to "bug-fix issues/commits" by manually extracting data from k9-mail—further, we inherited data regarding "class role changes" from materials provided by Fröding and Nguyen-Ngoc. Relying on "R," a statistical tool, we run the correlation tests on several corresponding pairs to discover the evidence to support/refute the hypothesis.

This study contributes to filling in the blanks in the software field and deepening our understanding of whether "architecture change "by "class role change" is associated with" systematic change" mirrored by the "bug-fix commits/issues?" In turn, help software developers to predict whether "systematic change" resulting from" bug-fix commit" have impacts on the architecture change via "refactoring activities."? Besides, to clarify whether the bug-fix commits/issues are prone to one particular type of class role stereotype? In addition, It provides empirical evidence to back up the findings from previous work by M. Di. Penta et al.[3] and to verify our hypothesis. Thus, we can enhance software understanding and software prediction. then improve software maintenance via "early refactoring".

The structure of the remaining paper is presented below: In section II, we provide readers with the background knowledge needed to understand the research topic. In section II, we also summarize related works to the research topic. Section III presents the research questions(RQs)and the hypothesis, elaborated on the research methodology. Section IV shows the main findings of the study. We discuss the results to formulate answers for the proposed research questions and also verify the hypothesis in section V. Finally, we conclude our study then outline the future studies in section VI.

II. Background

A. Background knowledge

In this paragraph, We will briefly introduce some background knowledge: There are "issues", "open issues", "closed issues", and "labels" as well as "role of class". In many circumstances, software development produces software programs with two warnings: (1). It is imperfect to certain features incompletion. (2) It generates bugs in the system. [9]. Therefore, software developers and users wish to report those issues to track software activities, such as advising new features, reporting bugs and explaining code changes[12]. In Github, a super repository of millions of open source projects [9], There is an ad-hoc section for every repository, "*the issues*,"[2] serving just for those purposes. Besides, there are two types of issues in the "*issues section.*" "*Open issues*" indicate the reported problems undergoing investigation. Contrastly, the resolved issues are listed under the "*closed issues*" tab.

Unlike other bug tracking systems, the developers use Github's *issues* tracker primarily to tracing bugs, enhancements, and changing features through tagging systems [12]. Besides, the main goals for *issues i*n GitHub are to

[2] The definition of " issue": https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues#integrated-with-github

promote collaboration works on reproducing bugs, discussing the additional features or changes on source codes. Indeed, filling in the issues in Github isn't a difficult task to perform. Anyone who has a Github account can provide feedback. Moreover, the repository's administrator can create the issue and pull request template, encouraging contributors to open issues, fill in issues, and request features[12].

Further, The additional key components in Github's *issues* are

1. Labels: To facilitate the systematic classification of issues throughout a repository, the users can create or use in-house provided labels[10]. Usually, The different color-coded labels help users quickly identify the *issue'*s topic and filter out the desired issues [10].
2. Milestones: is a group of issues relating to features, code changes, periods[12].
3. The assignee: responsible for editing the issues[12].
4. The contributors: responsible for opening issues, filling in the issues, commenting issues[12].

On the other hand, according to Rebecca Wirfs-Brock et al.[5]. There are six types of class role stereotypes: information holder, service provider, structure, coordinator, interfacer, controller. Therefore, all classes and objects in different commits can be assigned either one or more role stereotypes while the software development iteration evolves. Further, every role stereotype has its special responsibility. Thus, the class/object with different role stereotypes shall collaborate to perform the specific contract tasks. Besides, In the paragraph below, We will discuss the roles and responsibilities.

- Information Holder(IH): Responsible for knowing information and providing information to other objects.[5]
- Service Provider(SP): Responsible for performing works and offering service to others.[5]
- Structurer(ST): Responsible for maintaining relationships between objects and information about these relationships [5].
- Interfacer(IT): Responsible for transforming request and information between different parts of the system.[5]
- Coordinator(CD): Responsible for delegating works to others.[5]
- Controller: (CT): an object designed to make decisions and control a complex task.[5]

B. Related works

We chose some most relevant papers which help us to understand the background of the research topic and to realize the gaps between current knowledge and the aim of our research then presented a literature review below:

- B. J. Williams and J. C. Carver[2] present a study on the cause and effect of software changes through a systematic literature review(SLR) in a paper "characterize software architecture changes." Besides discussions about the different classification criteria of software changes and their impacts on the software architecture, they introduce a scheme(SACCS)based on the results from the SLR to identify the various effects upon high-level architecture resulting from software changes. Despite a detailed explanation of the causal relationship between multiple software changes and their impacts, It lacks an explicit description of "systematic changes." Nevertheless, we can use their findings to explain many concepts in our study to improve our initial understanding of the abstract view of our research's goal. Such as "type of change," "architecture change," etc.
- M. Di. Penta et al.[3] have replicated their

previous work (3 open-source projects) in the larger sample set(103 Apache repositories) to verify the relationship between refactoring operations and bugs-fix in the system by a better toolchain. The results strengthen their previous findings on the correlation but deserve further study on which type of refactoring operations most likely introduce the fix in the system. However, their conclusion has contradicted the results from some previous studies in the same domain. Therefore, further examination may be necessary. Nevertheless, their works provide a trustworthy theoretical foundation to our study.

- Fröding and Nguyen-Ngoc[4] use the classifier to classify and study the evolution of class roles in three open-source projects by color graphs, including K9 mail, sweet home 3D, BitCoin wallet. Moreover, their data collection strategy is to choose three open-source projects from GitHub randomly. All of the committed codes from the three projects are written in Java language. Our study uses the data of class role changes in k9-mail created in the tables by Fröding and Nguyen-Ngoc. to sum up the number of class role changes in selected releases from k9-mail. Thus, we can learn which releases of the class role changes are more prominent. Further, to investigate whether bug-fix issues in k9-mail are subject to one specific class role stereotype.
- Rebecca Wirfs-brock introduces her taxonomy version on class role stereotypes in the "characterize classes" [5]. There are information holders(IH), Service provider (SP), Interfacer(IF), Structurer(ST), Coordinator(CO) and Controller(CT). Further, the main focus is the responsibility of each role stereotype and the collaborative works among different objects with various roles. Besides, the author points out the class characterization serves two purposes on object design. These are to clarify the important aspects of the class's expected behaviors and communicate its design intention with others [5]. Our research uses six role stereotypes specified in Wirfs-Brock's paper to examine their possible correlations with bug-fix issues in k9-mail. Further, to explore which individual class role type is more prone to bug-fix issues in the repository.
- In their joint paper, Truong Ho-Quan et al. introduces an automated machine learning approach to class role stereotypes classification [6]. This research collected data from an open-source project, K9-mail, from GitHub then extracted features from source code committed in several releases. Next, they established a ground truth as classification benchmarks for machine learning tools. Then comparing feature performance resulting from three different ML algorithms. Thus, they can avoid validity threats from all counts due to method triangulations[16] applied. Moreover, this paper contributes to the field by introducing the machine learning approach to class role classification rather than the rule-based approach. Further, we can use the knowledge from their paper to explain some phenomena that occurred in our research.
- Y. Perez-Riverol et al. briefly present ten simple rules for using Git and GitHub in the research project.[12] They mentioned some background knowledge regarding how users, teams, and organizations use "Git" to track the project. Further, How "forking and branching" help in collaborative development. Also, they discuss "continuous integration" and "automation" using tools to integrate and test repositories hosted on Github, such

as Travis CI. Primarily, they introduce the interaction among Github's contributors through the "*issues.* ". Moreover, Rule number 7 from their paper focuses on the differences between Github's issue tracker and other tracker systems, classifying the issues via the color-coded tagging system. Thus, their paper contributes mainly to data collection practice in our research. For example, we can easily filter out desired issues through the tagging system in k9-mail.

III. Research Methodology

A. The rationale on selection of research methodology :

This research aims to study a case using the exploratory and descriptive approach to observe, further portray the relationship between the "bug-fixed issues" and "class role change" in a software repository over time.

The case study is "an empirical method aimed at investigating contemporary phenomena in their context."[7] In this case, the "contemporary phenomena" refers to "the correlation" between "class role change and bug-fix issues" within most recent releases. Then, the "their context" refers to "selected releases in software repositories." Even though the data regarding "class role change" for most recent releases are not concluded from the previous work by Fröding and Nguyen-Ngoc,[4], the "deductive reasoning," a process of from general to specific[16], provides answers to that. More specifically, We can predict the phenomena appearing in the most recent releases through the multiple observations upon the phenomena from non-contiguous/contiguous releases in the timeline.

We believe that the "single case embedded study" [7]is most suitable to serve our purpose because:

- The single case study using quantitative data provides an in-depth analysis of a particular phenomenon within its context.[14]
- A single case study can create a more complicated theory than a multi-case study because researchers can fit their theory with many details in a particular case.[13]
- The single case study gains a deeper understanding of circumstances where the particular phenomena occurred. Thus, the "generalization" results tend to be more reliable from a single case study than from multi-case studies. (external validity and reliability)[13]

B. Case description:

1.The Case selection:

In this research, we study K9-Mail[3] which is an independent email application designed for the Android system. It has had multiple stable releases over a decade.

2.The Case Inclusion /Exclusion criteria :

Fröding and Nguyen-Ngoc[4] have concluded the evolution of class roles for three open-source projects in Github. We chose only one out of three projects for the current study, k9-mail. It had almost 700 "closed issues"labeled as "bug." In comparison with the other two projects, the "Bitcoin wallet" had only 9 "closed issues "relating to "bug"; the "sweet home 3D" had no "issues" at all. Therefore, the likelihood of extracting sizable "issues" relating to "bug-fix" from the k9-mail is relatively high.

Further, since there is no corresponding data relating to "class role change"readily available, extracting "issues" from a randomly selected repository at Github becomes absurd.

[3] K9-Mail homepage: https://github.com/k9mail/k-9

C. The rationale on selection of data collection methods:

On the other hand, P. Runeson and M. Höst proposed that a case study may contain elements of other research methods. e.g., "archival analyses may be part of its data collection."[7] Further, Dr. Layder argues, "archival data "also has its place in contemporary-oriented research threefold. (1) It adds "empirical depth" to a project by providing extra data to verify the data from the other sources. (2). Archival data mainly explain the process of change and evolution. such as "class role change in k9-mail."(3). It can be used to challenge the existing theory.[15] In this study, Due to most of raw data can be obtained either through a Github's repository or archival data presented by other researchers, the "archival analysis "sounds suitable for doing the data collection jobs.

In this section, we will give a brief introduction to the case. Then, formulate the research questions (RQs) and present hypotheses. Next, elaborate on the data collection and data analysis procedures in detail.

D.Research Questions (RQs) and Hypothesis:

Table 1: Research Questions

RQ1:	*Are there any correlations between "closed issues" with the label "bug" and architectural role changes of class in software repositories?*
RQ1.1:	*Are there any correlations between "closed issues" with the label "bug" and systemic change in software repositories?*
RQ1.2:	*Whether the amount of bug-fixed issues are more subject to one particular type of class role change ? Which stereotype ?*

E.Data collection process:

We rely on the data of role changes collected/identified by *Fröding and Nguyen-Ngoc*[4]. Specifically, the authors classified role changes from the number of 31 selected releases of K9-Mail. To collect issues that are relevant for bug-fix, we use the following steps:

- Step one: Go into the issue list[4]
- Step two: Filter closed issues with label "bug" by choosing the color-coded label "bug" from the label list.[5]
- Step three: Click "amount of closed issues"[6] under the "search field.
- Step four: Sort the "issues" ascendly.

Next, We will describe how various types of data are extracted from k9-mail. there are four units of analysis for this research. In other words, four data types:

1. Bug-fix issues;
2. Amount changes of class role stereotypes;
3. Total counts on "closed issues" with label "bug";
4. The changing frequency of every class role stereotype for each release.[7].

1. The data regarding the number of "bug-fix issues"in k9-mail.

We will extract the raw data by manually reviewing all of the *"closed issues" with* the label "bug" in K-9 mail to conclude the number of bug-fix issues in the selected releases. The

[4] Issue list K9-Mail: https://github.com/k9mail/k-9/issues

[5] The label list in k9-mail: https://github.com/k9mail/k-9/labels

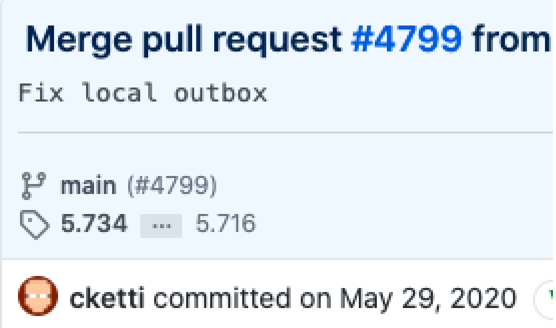
[6] The location of where are the number of closed Issues :https://github.com/k9mail/k-9/issues?q=label%3Abug+is%3Aclosed

rationale behind this choice is that there is no readily available tool to extract bug-fix commits from the " issues" so far. Especially when the commit message has no clear indication, such as "bugs-fix" or "fix" entailed in the"issues." Further, it is very difficult to run batch operations for "feature extraction."

In this study, The closed issues with the label "bug" don't always refer to the bug-fix issues .Implies, bug-fix commits can't be found in the issue's body of content —for example, #749 from Table III. Contrarily, if the developers can find the bug-fixed commits in the issue's body of context, the issue shall be classified as a bug-fix issue. For instance: #828 in Table III. Particularly In this research, we aim to explore how bug-fix issues are associated with architectural role changes of class in k9-mail. Therefore, we have to filter out the closed issues with the label "bug," which are considered as bug-fix issues.

Meanwhile, to classify bug-fix issues and none bug-fix issues in k9-mail, We will pay attention to some most frequently appearing terms/phrases/labels in the corpus of every *"closed issue."* For instance: "fix the bug in commit 2d67b49''[7], "purple-colored *Merge label*."[8] "The assignee closed this in commit #",[9] " *The assignee* closed the issues at #2367."[10] etc. According to our observation, Those terms are always associated with bug-fix commits in k9-mail.

Further, We will quickly figure out the most appropriate "release" for every "issue" from the tag list after identifying the *commit id and commit date linking to the "issue"(see figure 2).* So far, We have closely examined more than 450 closed issues with the label "bug." Then, categorize them into two groups: "bug-fix issues" and "none bug-fix issues." Further, We assort the "bug-fix issues" on a release basis.(see the "Table II" in the appendix)



Merge pull request #4799 from

Fix local outbox

main (#4799)

5.734 ... 5.716

ckettti committed on May 29, 2020

Figure 2; the commit and matched release

2. Amount changes of class role stereotypes for every release in k9-mail:

Relying on the Table "k9-change-numbers.csv" concluded by Fröding and Nguyen-Ngoc [4], We apply the "R" commands "sum" to calculate the number of class roles changes for every attribute column in the Table from the year 2014 to the year 2020. Then match the summation of data from each attribute column to a corresponding "release." Moreover, The rationale behind this selection is that "bug-fix issues" are not found before the year 2014 except "2012-02-02". Further, We use the following "R" code to summarize the number of "class role changes" in every period from 2014 to 2020. For instance:

sum(k9.changes.numbers.1$V17)

Additionally, At "Table III" in the appendix, It is not difficult to obtain the summation of the amount for every class role type over selected release.

3. Total counts on "closed issues" with label "bug"in k9-mail:

To answer RQ1.1, we have to obtain the total counts for both bug-fixed and none bug-fixed issues for every selected release from k9-mail to explore their correlation with bug-fixed issues/commits. (Systematic Changes). The concrete steps are following:

1. Based on the various releases from k9-mail, We sorted and categorized "bug-fixed issues" and "none bug-fixed

[7] Fix the bugs in the commit#: https://github.com/k9mail/k-9/issues/828
[8] Purple-colored merge label: https://github.com/k9mail/k-9/issues/744
[9] The assignee closed this in the commit ### https://github.com/k9mail/k-9/issues/1151
[10] The assignee closed the issue at #XXXX https://github.com/k9mail/k-9/issues/811

issues" separately by the " close date" column in Table II. Then, transfer the results to Table IV. into the columns of "bug-fix issues/commits" and "none bug-fix issues."

2. Based on the different "releases," We have merged the redundant rows into one for the column "bug-fix/release" Thus, we obtained the number of "bug-fix issues" for every release. Then transfer the column into Table IV.
3. Table II, We match the "close date" of "none bug-fixed issues" with the date in/between releases for "bugs-fixed issues." For instance: the close date for issue #660[11] is "16/05/2015," which matches the date between release V.5.106 and release V. 5.006. Therefore, #660 shall be categorized into the V.5.006. Before implant the results into Table IV, We sum the amount of "none bug fix issues" release by release.
4. Add up the number of "bug-fixed issues" and "none bug-fixed issues" from k9-mail to get values for "total counts" column in Table IV.

4. The changing frequency of every class role stereotype for each release:(RQ1.2)

Depending on the data concluded from the table "k9-change-names.csv" by Fröding and Nguyen-Ngoc [4], We apply "R, "a software for statistical analysis," to calculate the number of classes that have changed role types in each release. Further, we use the following "R commands" to trace which class has changed roles between two adjacent attribute columns to know from which role type shifts to which role type. Then, get summation of class role changes at the ending column (see code snippet 1). By constant comparison between two adjacent attribute columns from the table, we will discover the changing frequency of each class role type for all selected releases. Further, the variation from "unidentified role type" to any identified role type [5] won't be taken into account in this research also vice versa. We are only interested in alterations that occurred between the identified class role types.[12]

Moreover, We will demonstrate how "this comparison" works in “code snippet 1” below. We will run the "R" commands to exemplify the change that occurred on every class role stereotype between column 5 and column 6 in the table, In other words, from "2011-11-01" to "2012-02-02".

Code Snippet 1:

```
k9.changes.name.1$V5<-sample(c("Information Holder", "Coordinator", "Service Provider", "Controller","Interfacer", " Structurer"),2634, replace= T);
k9.changes.name.1$V6<-sample(c("Information Holder", "Coordinator", "Service Provider", "Controller","Interfacer", " Structurer"),2634, replace= T);
Df2;
```

F. Data Analysis

A. Motive to perform the "correlation test"

Based on the data collected from the previous phase, We summarized the results from each attribute column from Table III. So far, there are 217 "closed issues" out of 465 marked with the label "bug" associated with the bug-fix commits. Further, one thing that deserves to mention is that all of the bug-fix issues revealed after the release of V5.709 will be excluded from this research because the relevant data for "class role changes" is not found after the release of V.5.709. Therefore, It is difficult to correlate them mathematically. Moreover, the number of classes that had role changes for every selected release is shown in Table III by the "role change/release" column. Meanwhile, the number of bug-fix issues

[11] #660: https://github.com/k9mail/k-9/issues/660

[12] The identified class role type refers to Structure, Information Holder, Coordinator, Controller, Service Provider, Interfecer.

appearing in every selected release is concluded in that Table as well, by the column of "bug issue/release". Further, the data relating to "total counts'' on the amount of" bug-fixed issues" and "none bug-fixed issues" for every selected release from k9-mail is presented in Table IV by the column "Total counts."

As yet, We discovered a considerable number of class role changes in several releases, such as V5.300, V5.500, V 5.700, and V5.709. However, not too many class role changes in the other releases. Likewise, plenty of bug-fix issues appeared in those releases above but not in the others. So, It seems that the changing trend is concordant. On the contrary, the significant number of bug-fix issues found in other releases are not always associated with a drastic number of "class role changes" in matched releases, such as V5.007 and V5.114. Therefore, there might be some confounding factors other than "bug-fix " affecting class role changes. Thus, We believe that it is necessary to perform "the correlation test" to discover reliable evidence to support/refute the Hypothesis.

B. Tool introduction:

To perform the "correlation test" on the datasets from Table III and Table IV, we shall brief the " software tool" we use to run the test first. We chose "R,"[13] a programming language and environment for statistical computing and graphics, developed by Bell Laboratories. It provides a variety of statistical and graphical techniques including mathematical modeling, statistical test, and analysis. Etc. With extensive easy install add-ons, It has become an increasingly popular choice to run statistical tests and graphic modeling.

C. The inclusive/exclusive criteria for the type of statistical test:

Before we run any "correlation" tests on any datasets from Table III and Table IV, we shall closely examine their "Normality" and "Monotonic" to determine which test most suits the role. Therefore, the "Shapiro-wiki test"[14] is ideal for checking the "Normality" of the data set with small sample size. (n<50) Likewise, the "Mann-Kendall test" is good for checking the "monotonic trend" of the data set. Thus, we run both tests via "R" for all input datasets(see code snippet 0). As both the" P" value from the "Shapiro test" and "S" value from the "Mann-Kendall test" indicated, all of the datasets from both tables don't follow a normal distribution pattern. Therefore, they are non-parametric. (fig3-fig11) Further, they all follow a monotonic upward trend if the "S" value is positive. Otherwise, a downwards monotonic trend if the "S" is negative. [15] Further verification is presented from fig12 to fig 20 in the appendix. Besides, The "R" command below will exemplify the "Mann-Kendall test" and "Shapiro-Wiki" test performed on the "Interfecer" column from Table III.

Code snippet 0:

```
MK.test(monotonic_checker$IT)[16]
Shapiro.test(monotonic_checker$IT)
```

In addition, There are several non-parametric candidates available to test the dissimilarity between the two independent samples. The Mann-Whitney U test [17]is used to test whether or not the distribution from both samples has no difference. In contrast, Fisher's exact[18] suit to find the difference between two nominal categorical groups. Further, same as Fisher's exact, Chi-square[19] applies to the situation where

[13] What is "R": https://www.r-project.org/about.html

[14] What is "Shapiro-Wiki": https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test

[15] What is " S" value in Mann-kendall trend test: https://help.healthycities.org/hc/en-us/articles/233420187-Mann-Kendall-Test-for-Trend-Overview

[16] The MK test refers to the " Mann-Kendall trend test".

[17] Mann-Whitney U test: https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test

[18] Fisher' exact test: https://en.wikipedia.org/wiki/Fisher%27s_exact_test

[19] Chi-square test: https://en.wikipedia.org/wiki/Chi-squared_test#Chi-squared_test_for_variance_in_a_normal_population

the variables come from two categorical groups and "test statistics" approach to " $\chi2$" distribution. Kruskal-Wallis test[20] is only suited for comparisons on more than two unpaired groups. Therefore, only the "Spearman ranked test" is suitable for detecting the monotonic association between two independent samples, ranked-categorical variables. .Besides, the "ρ "[21]value shows how strong two samples tie to each other.[22]

Moreover, In the book "Empirical research in software engineering: concepts, analysis, and applications," R. Malhotra[11] mentioned that if the distributions of data sets are highly skewed, the non-parametric technique (Spearman's "ρ") for measuring relationships can be used.[8] Besides, We want to discover the evidence through Spearman's test to verify the hypothesis in mind and answer the research questions. In addition, We present the null hypothesis and alternative hypothesis below:

H_0 =There are no correlations between bug-fixed issues and class role change in k9-mail.

H_1 = There are correlations between bug-fixed issues and class role change in k9-mail .

H_0 =There are no correlations between "closed issues" with the label " bug" and systematic change in k9-mail.

H_1 = There are correlations between "closed issues" with the label " bug" and systematic change in k9-mail.

D. To perform the "Spearman Ranked test"

We will run the "Spearman ranked test" [23]by "R" for seven pairs of attribute columns from table III. Additionally, one pair of attribute columns from table IV to discover the answers to research questions and verify the hypothesis. Hence,

The first pair is between the "total bug-fix issues/release" column and the "total class role changes/release" column. Meaning, the correlation between the " total bug-fix issues in every selected release" column and the column of "the number of classes which have changed roles in every selected release":

Code snippet 2:

```
cor.test(data_role_bug$`role
change/release`,data_role_bug$`bug
issues/relea`,method"=
spearman",exact= F)
```

The second pair is between "IT" and " total bug-fix issues/release". Meaning, between the column of "Interfacer" class role type and the column of "total bug-fix issues in every release."

Code snippet 3:

```
cor.test(data_role_bug$IT,data_role_bug$
bug_issues/relea`,method=
"spearman", exact = F)
```

The third pair is between "CD" and " total bug-fix issues/release". Meaning, between the column of "Coordinator" class role type and the column of "total bug-fix issues in every release."

Code snippet 4:

```
cor.test(data_role_bug$CD,
data_role_bug$`bug_issues/relea`,
method= "spearman", exact = F)
```

The fourth pair is between "CT" and " total bug-fix issues/release". Meaning, between the column of "Controller" class role type and the column of "total bug-fix issues in every release."

Code snippet 5:

```
cor.test(data_role_bug$CT,
data_role_bug$`bug_issues/relea`,
method= "spearman", exact = F)
```

[20] Kruskal-Wallis test: https://en.wikipedia.org/wiki/Kruskal%E2%80%93Wallis_one-way_analysis_of_variance

[21]" ρ " refers to " correlation".

[22] Which test to choose? https://www.healthknowledge.org.uk/sites/default/files/documents/publichealthtextbook/statistics/parametric2.png

[23] What is "Spearman ranked Test"? https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/spearman-rank-correlation/

The fifth pair is between "ST" and " total bug-fix issues/release". Meaning, between the column of "Structurer" class role type and the column of "total bug-fix issues in every release."

Code snippet 6:

```
cor.test(data_role_bug$ST,
data_role_bug$`bug_issues/relea`,
method= "spearman", exact = F)
```

The sixth pair is between "IH" and " total bug-fix issues/release". Meaning, between the column of "information Holder" class role type and the column of "total bug-fix issues in every release."

Code snippet 7:

```
cor.test(data_role_bug$IH,
data_role_bug$`bug_issues/relea`,
method= "spearman", exact = F)
```

The seventh pair is between "SP" and " total bug-fix issues/release". Meaning, between the column of "Service Provider" class role type and the column of "total bug-fix issues in every release."

Code snippet 8:

```
cor.test(data_role_bug$SP,
data_role_bug$`bug_issues/relea`,
method= "spearman", exact = F)
```

Additionally, one more pair from table IV is between "bug-fix issues/commits" and "total counts on the amount of bug-fixed issues and none bug-fixed issues". In other words, between the column of "bug-fixed issues/commits" and the column of "total counts" for every selected release from k9-mail. The reason is that we want to discover the correlation between a labeled dataset(buggy issues from k9-mail) and systematic code change in the repository.

Code snippet 9:

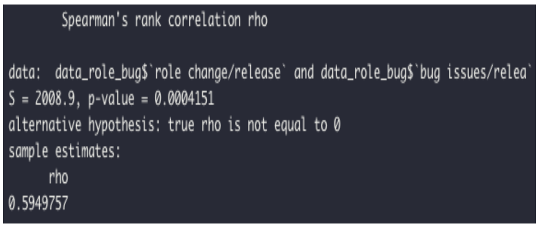
```
cor.test(data_total_counts$`bug-fix
issues/commits`,data_total_counts$`total
counts`, method = "spearman", exact = F)
```

IV. Result

RQ1: *Are there any correlations between "closed issues"with the label "bug" and architectural role changes of class in software repositories?*

H_0: There are no correlation between “closed issues” with the label “bug” and architectural role changes of class in software repositories.

H_1: There are correlation between “closed issues” with the label “bug” and architectural role changes of class in software repositories.

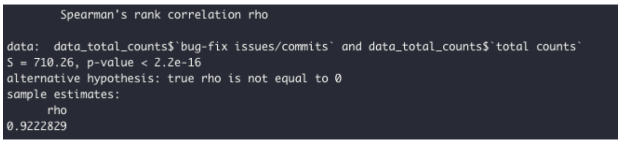


```
        Spearman's rank correlation rho

data:  data_role_bug$`role change/release` and data_role_bug$`bug issues/relea`
S = 2008.9, p-value = 0.0004151
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.5949757
```

Fig 21: the relationship between the column of " bug-fixed issues/release"and column of " role change/release" from k9-mail.

As figure 21 indicates, according to calculation from code snippet 2, the extremely low positive "p-value" of "0.000415", which is slightly greater than 0, implies statistical significance between two attribute columns from table III. The column of " bug-fixed issues/release" and the column of " role change/release." In addition, the "ρ"cofficience from the "Spearman test" is "0.5949757," which is more than "0.5." Meaning a strong positive correlation between the two columns.Thus, the alternative hypothesis is confirmed, and the null hypothesis is declined.

RQ1.1: *Are there any correlations between "closed issues" with the label"bug” and systemic change in software repositories?*

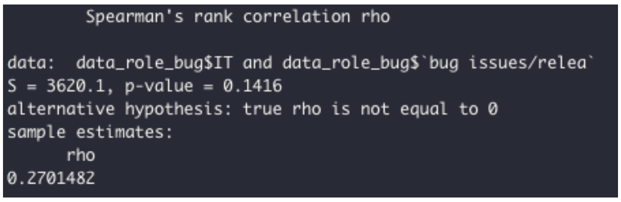


```
        Spearman's rank correlation rho

data:  data_total_counts$`bug-fix issues/commits` and data_total_counts$`total counts`
S = 710.26, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.9222829
```

Fig22: the relationship between column of " bug-fixed issues/commits" and column of "total counts" in k9-mail

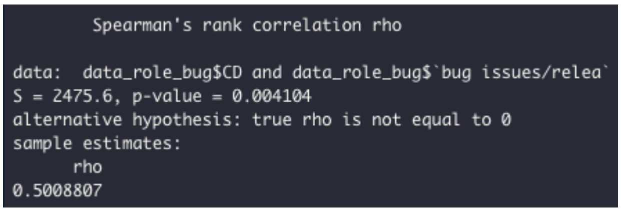
As figure 22 indicates, according to the calculation by code snippet 9, the "P-value", which is less than "2.2e-16" and approximately to "0", implies robust statistical significance between two attribute columns in table IV. There are " bug-fix issues/commits" and "total counts". In addition, the "ρ" coefficient from the "Spearman test" is "0.9222829," which is close to "1", Meaning a robust positive correlation between the two columns. Thus, the alternative hypothesis is confirmed, and the null hypothesis is rejected.

RQ1.2: *Whether the amount of bug-fixed issues are more subject to one particular type of class role change in software repositories?Which stereotype ?*



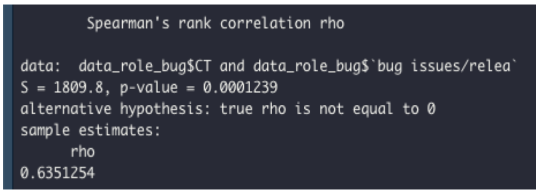
```
        Spearman's rank correlation rho

data:  data_role_bug$IT and data_role_bug$`bug issues/relea`
S = 3620.1, p-value = 0.1416
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.2701482
```

Fig23: relationship between column of "IT"and column of "bug issues/release"



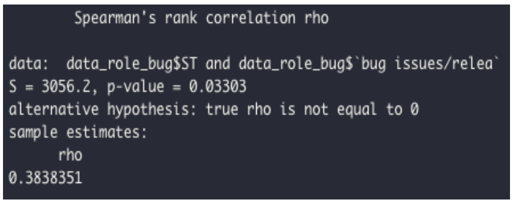
```
        Spearman's rank correlation rho

data:  data_role_bug$CD and data_role_bug$`bug issues/relea`
S = 2475.6, p-value = 0.004104
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.5008807
```

Fig 24: the relationship between column of "CD "and column of " bug issues/release"



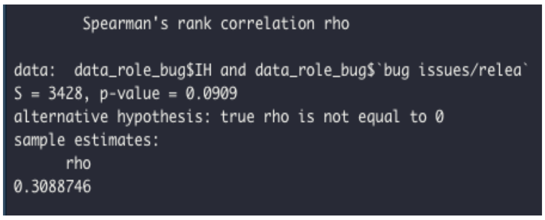
```
        Spearman's rank correlation rho

data:  data_role_bug$CT and data_role_bug$`bug issues/relea`
S = 1809.8, p-value = 0.0001239
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.6351254
```

Fig 25: the relationship between column of "CT"and column of " bug issues/release"



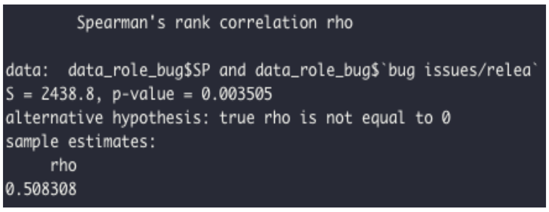
```
        Spearman's rank correlation rho

data:  data_role_bug$ST and data_role_bug$`bug issues/relea`
S = 3056.2, p-value = 0.03303
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.3838351
```

Fig 26: the relationship between column of "ST"and column of " bug issues/release"



```
        Spearman's rank correlation rho

data:  data_role_bug$IH and data_role_bug$`bug issues/relea`
S = 3428, p-value = 0.0909
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.3088746
```

Fig 27: the relationship between column of "IH"and column of " bug issues/release"



```
        Spearman's rank correlation rho

data:  data_role_bug$SP and data_role_bug$`bug issues/relea`
S = 2438.8, p-value = 0.003505
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.508308
```

Fig 28: the relationship between column of "SP"and column of " bug issues/release"

As figure 23 to figure 28 suggest, according to the calculations from code snippet 3 to code snippet 8 in section III, The bug-fixed issues in selected releases from k9-mail are more subject to some particular class role stereotypes. such as "Controller," "Coordinator," and "Service Provider. "Besides, the lower "P-values" and higher "ρ" values are displayed respectively in figure24, figure25, and figure28 to clarify this point. Especially the "Controller" architecture role type, the" p-value "equal to "0.0001239," which is far less than the " α " value of"0.05," and the "ρ" value equal to "0.6351254" indicates the robust correlation between mentioned entities above. Meaning: We shall reject the null hypothesis.

On the contrary, we found that except for those three mentioned class role types, the higher "p-values" and lower "ρ" coefficient values appearing in figure 23, figure 26, and figure 27 imply weak correlations between bug-fixed issues and the other class role types. In other words, statistically insignificant.

V. Discussion

In this section, We use the results from the previous sections to answer the RQs and verify the hypothesis. Then, discuss the validity threats.

A. Answers for RQs

RQ1 : *Are there any correlations between "closed issues"with the label "bug" and architectural role changes of class in software repositories?*

We discovered Non-Normal distribution for both attribute columns of "bug issues/release " and "role change/release" in table III. This pattern can be verified by the exceedingly low "P" values displayed in both figure 18 and figure 19, resulting from the "Shapiro-Wiki" test. However, if we observe them as pairs, the concordant pattern between them is pretty obvious. In many circumstances, sharp increases in bug-fixed issues from several releases in k9-mail are always associated with a significant number of class role changes at the same releases. e.g., 22 bug-fixed issues found in the release V.5.500, corresponding with as many as 13 times class roles change at the same release. Likewise, we found a similar concordant pattern in V 5.700 and V.5.709, etc. However, It is not very explicit for many other releases. e.g., there is a drastic rise for bug-fixed issues at the release V.5.114, a total of 17 of such issues, but we discovered no corresponding class role changes in the same release. Thus, this phenomenon may be due to some other confounding factors. We decide to perform the "Spearman" test by "R" to uncover the evidence to validate the hypothesis.

Figure 18 and Figure 19 demonstrated that the sort order for both datasets is concordant because the " S" values from Mann-Kendall tests are positive. As figure 21 from section IV revealed, we shall choose the alternative hypothesis(H1) then reject the null hypothesis(H0). The P-value of 0.0004151 falls below the significance level(α=0.05). Furthermore, the ρ (rho) coefficient, which represents the correlation between two attribute columns, equals "0.594957." which is greater than "0," Meaning a robust positive correlation between the two columns. So, the relationship is statistically significant.

Furthermore, this conclusion reconfirms the findings from a previous study by M. Di. Penta et al.[3]: the architecture change through refactoring operations is correlated with bug-fix in the system.

RQ1.1: *Are there any correlations between "closed issues" with the label"bug" and systemic change in software repositories?*

As figure 1 illustrated, the "bug-fixed commits/issues" column from Table IV is the quantifiable reflection of "systematic changes" upon the software system. Meanwhile, the "bug-fixed issues" enclosed "bug-fix commits." In addition, the "total counts" column from Table IV connotes the summation between amounts of bug-fixed issues and none bug-fixed issues in the same release. Meanwhile, they are synonymous with "the total number of 'closed issues' with a label 'bug' in k9-mail".

We have to observe these two attribute columns from table IV to explore their relationship. Moreover, we spot that two datasets seem to change concurrently towards the same direction. As figure 19 and figure 20 revealed, the positive "S" values resulting from the "MK test" are displayed in both figures. Meaning, A growing number of "closed issues" with the label"bug" over selected releases are associated with the significant number increase of "bug-fixed commits"over the same releases. For instance: In release V.5.007, 17 buggy issues[24] are correspondent with 11 buggy

[24] Buggy issues refer to" bug-fix issues"

commits[25]. So far, we have found the same pattern for many other releases; 14 buggy issues correspond to 9 buggy commits in V.5.109; 11 buggy issues correspond to 9 buggy commits in V.5.111; 15 buggy issues correspond to 14 buggy commits in V.5.114. 25 buggy issues correspond to 13 buggy commits in V.5.204; 21 buggy issues correspond to 17 buggy commits in V.5.300; 13 buggy issues correspond to 12 buggy commits in V.5.301; 24 buggy issues correspond to 22 buggy commits in V.5.500; 83 buggy issues correspond to 25 buggy commits in V.5.700, and 21 buggy issues correspond to 18 buggy commits in V.5.709.

We run the "Spearman" test for those two columns by "R" to further verify the hypothesis. Hence, the variables in figure 22 clearly explain all this. The exceedingly low P-value of "2.2e-16" and an unusually high "ρ" coefficient value of "0.9222829." Implying the strong positive correlation between systematic changes and "closed issues with label 'bug'" over selected releases in k9-mail.

In addition, this finding is consistent with the conclusion from previous work by B. J. Williams and J. C. Carver[2]: the increasing number of "defects" results in a systematic change. They are deriving from system quality decline.

RQ1.2: *Whether the amount of bug-fixed issues are subject to one particular type of class role change in software repositories? Which stereotype ?*

To answer RQ1.2, we have closely examined the correlations between the six pairs of attribute columns from table III. They respectively are "IT" and "bug-fix/release," "CD" and "bug-fix/release," "CT" and "bug-fix/release," "ST" and "bug-fix/release," "IH" and "bug-fix/release," and "SP" and "bug-fix/release." Before running any "Correlation test," we have to detect the monotonic trend of every input attribute column to determine whether or not the "Spearman test" suits the job. We perform the "Mann-Kendall test" by "R" for every attribute column relating to "role stereotype." The "S" value from Figures 12 to 17 shows either monotonic increase or monotonic decrease. However, It is hard to determine which class role stereotype is more prone to influence the number of bug-fixed issues over selected releases from k9-mail through naked eyes. Thus, We sum the figures from each column relating “class role type” over selected releases from table III by the "R" command like following:

```
sum(data_role_bug$IT)
```

The changing frequency of each "class role stereotype" is presented below:

- The classes have changed role stereotypes to "Interfacer" 18 times.
- The classes have changed the role stereotypes to "Coordinator"13 times.
- The classes have changed the role stereotype to "Controller" 10 times.
- The classes have changed the role stereotype to "Structurer" 11 times.
- The classes have changed the role stereotype to "Information Holder" 7 times.
- The classes have changed the role stereotype to "Service Provider" 20 times.

.

We get the sum from the "bug-fixed issues/release" column via “R” command below,, a total of 217 bug-fix issues.

```
sum(data_role_bug$`bug issues/relea`)
```

[25] Buggy commits refer to “bug-fix commits”

Those figures (fig 23-fig28) revealed compelling evidence that the "Controller" stereotype demonstrates a very low positive p-value of "0.0001239" and a higher "ρ" coefficient value of " 0.6351254," which is greater than the "ρ" value of "Coordinator," "0.5008807." and the "ρ" value of "Service Provider," which is "0.508308." Meaning, the amount of bug-fixed issues is more likely subject to the "Controller" stereotype across selected releases from k9-mail.

Furthermore, Wirfs-Brock pointed out the "Controller" responsible for controlling a complex task. Meanwhile, It makes most of the decisions.[5] It may increase both direct and indirect interdependency between the "Controller" class and other objects/classes of the system, Thereby increasing "complexity," resulting in "high coupling" between software modules. The higher the "coupling" among them, the higher the risk of incurring the bug-fixed in the repository. Besides, Truong Ho-Quan et al.[6] mentioned, " the design intention of these classes suggest that the decisions made by controllers affect a broader control flow of the system." That is another driving cause of inducing fixes in the system. Moreover, their work[6] suggests that the " Coordinator" shall collaborate with the " Service Provider" to perform cross-layer tasks. Therefore, each "class role" alone doesn't affect the system as much as the Controller. That's probably the reason why a growing number of bug-fixed issues in k-9mail is always associated with an occurrence of the "controller" class role stereotype.

B. Validity threats:

Construct Validity: We closely examine more than 450 closed issues with a specific label in k9-mail. Besides, "Inductive" is the process of from specific to the general.[14, p. 2] So, The chosen case can represent many repositories with a lot in common with "k9-mail." Moreover, we can have an in-depth analysis for a single case despite time constraints and limited resources. However, the data extracted from a single archival may lead to bias due to skepticism on data reliability from the single source. In addition, The conclusion results from the single method calculation, Namely, the "Spearman ranked test." It might lead to bias due to the lack of verifications from alternative methods. In general, the source triangulation and method triangulation produce more reliable results. However, To a large extent, the results from this study verify our hypothesis. Thus, we can use them to answer our research questions. (RQs)

Internal Validity: As we mentioned early on, the "Spearman ranked test" is the "distribution-free"[26] test. In other words, It suits any input datasets with/without recognizable distribution patterns. Therefore, we believe that the results rule out the possibilities the confounding factors may play a part.

External validity: The research is a single case embedded study. Due to the nature of the research per se, It is hard to draw a precise conclusion based on evidence gathered from a single source. It may not represent the whole picture in the industry. Even though the results have already verified our hypothesis, the degree of generalization of the results into the broader spectrum is relatively minimal.

VI. Conclusion

In conclusion, we used the empirical evidence collected from k9-mail to corroborate the hypothesis, Then answered the research questions(RQs).

After closely examining over 450 "closed issues" with the label "bug" over selected releases from k9-mail, we gathered four data types for this case study. Then, we performed the "Spearman ranked test" on seven pairs of attribute columns from table III and one

[26] "The distribution-free test": https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/parametric-and-non-parametric-data/

additional pair of attribute columns from table IV to verify the hypothesis of whether the correlations exist between those pairs. The low p-value and extremely high positive "ρ" coefficients displayed in figure 22 prove the strong positive correlation between " closed issues" with a specific label and systematic change in k9-mail. Further, the empirical evidence displayed in figure 21 indicates the "systematic changes" reflected by " bug-fixed issues" are strongly correlated with the "architectural role changes" reflected by "class roles/responsibilities changes" over selected releases in k9-mail. Moreover, the evidence reflected by the low p-value and high positive "ρ" value from figure 25 has revealed that the bug-fixed issues found over selected releases from k9-mail have been prone to one particular architecture class role. Namely, the "Controller" stereotype. This phenomenon implies the developers may want to either shift the responsibilities of classes or increase the "directing activities" to other classes over selected release. As we mentioned in section V, this will increase " Coupling" between various classes., Resulting in a drastic increase of bug-fixed commits/issues in many releases.

Limitations:

Since the nature of the single case study, we cannot generalize its results into common sense.

Future Study:

If the resources are available, it could be interesting to replicate the current study within a large sample domain for another study. For instance, to expand this study into 100 cases scenarios with the assistance of a better toolchain. Thus, we can explore the relationship in the broader spectrum to find solid evidence to support/refute the hypothesis.

Since no prior studies are targeting this research topic, the finding from this study provides theoretical guidance to future studies in the same domain. The correlation between "bug-fix issues" and "class role change" could be an unproven conjecture. It also could become an established theory if the assumption is re-confirmed within a large sample domain in the future study. If that is the case, we will rule out the possibility that the chosen case is an unusual exception.

VII. Reference:

[1]. I. Sommerville, *Software engineering*, 9th ed. Boston: Pearson, 2011.

[2]. B. J. Williams and J. C. Carver, “Characterizing software architecture changes: A systematic review,” *Information and Software Technology*, vol. 52, no. 1, pp. 31–51, Jul. 2009.

[3]. M. Di. Penta, G. Bavota, and F. Zampetti, “On the relationship between refactoring actions and bugs: A differentiated replication,” *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1–10, 2020.

[4]. D. Nguyen Ngoc and F. Fröding, “The Evolution of Role-Stereotypes and Related Design (Anti)Patterns,” *GUPEA*, 03-Dec-2020. [Online]. Available: https://gupea.ub.gu.se/handle/2077/67098[Accessed: 30-May-2021].

[5]. R. Wirfs-Brock, "Characterizing classes", *IEEE Software*, vol. 23, no. 2, pp. 9-11, 2006. Available: 10.1109/ms.2006.43.

[6]. A. Nurwidyantoro, T. Ho-Quang, and M. R. Chaudron, “Automated Classification of Class Role-Stereotypes via Machine Learning,” *Proceedings of the Evaluation and Assessment on Software Engineering*, 2019.

[7]. P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering", *Empirical Software Engineering*, vol. 14, no. 2, pp. 131-164, 2008. Available: 10.1007/s10664-008-9102-8.

[8]. P. Schober, C. Boer and L. Schwarte, "Correlation Coefficients", *Anesthesia & Analgesia*, vol. 126, no. 5, pp. 1766, 2018. Available: 10.1213/ane.0000000000002864.

[9]. T. F. Bissyande, D. Lo, L. Jiang, L. Reveillere, J. Klein, and Y. L. Traon, “Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub,” *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, 2013.

[10]. J. Cabot, J. L. Canovas Izquierdo, V. Cosentino, and B. Rolandi, “Exploring the use of labels to categorize issues in Open-Source Software projects,” *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2015.

[11]. R. Malhotra, “Data analysis and Statistical testing,” in *Empirical research in software engineering: concepts, analysis and applications,* , 1st ed., Boca Raton, FL: CRC press, 2016, p. 218.

[12].]Y. Perez-Riverol et al., "Ten Simple Rules for Taking Advantage of Git and GitHub", *PLOS Computational Biology*, vol. 12, no. 7, p. e1004947, 2016. Available: 10.1371/journal.pcbi.1004947.

[13]. F. L. MARIOTTO, P. P. ZANNI, and G. H. MORAES, “What is the use of a single-case study in management research?,” *Revista de Administração de Empresas*, vol. 54, no. 4, pp. 358–369, 2014.

[14]. E. Athanassopoulos and M. G. Voskoglou, “A Philosophical Treatise on the Connection of Scientific Reasoning with Fuzzy Logic,” *Mathematics*, vol. 8, no. 6, p. 2, Jun. 2020.

[15]. C. Welch, “The archaeology of business networks: the use of archival records in case study research,” *Journal of Strategic Marketing*, vol. 8, no. 2, pp. 2–3, 2000.

[16]. M. Q. Patton, *Qualitative research and evaluation methods*, 3rd ed. Thousand Oaks, CA: Sage, 2002. Pp.247-249.

VIII. Appendix:

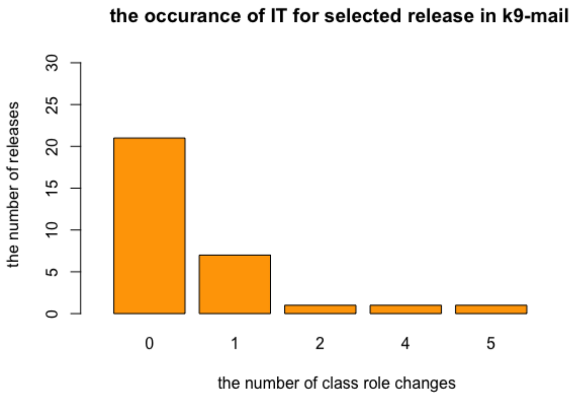


Fig 3: the distribution of "interfacer" for selected releases in k9-mail

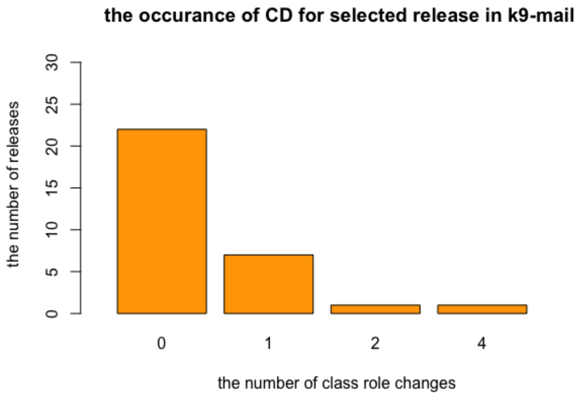


Fig 4:: the distribution of "Coordinator" for selected releases in k9-mail

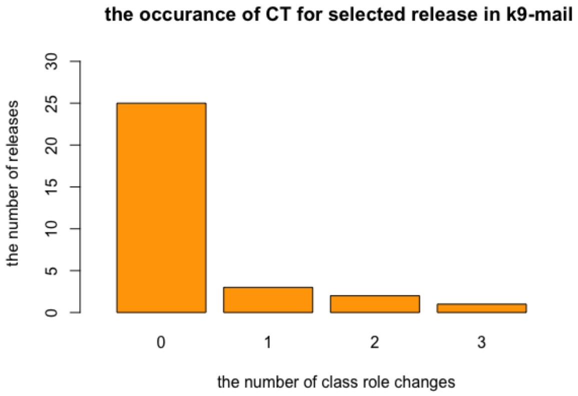


Fig 5: the distribution of "Controller" for selected releases in k9-mail

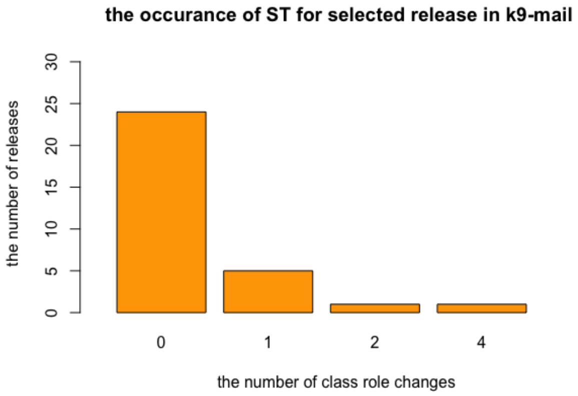


Fig 6: the distribution of "Structurer" for selected releases in k9-mail

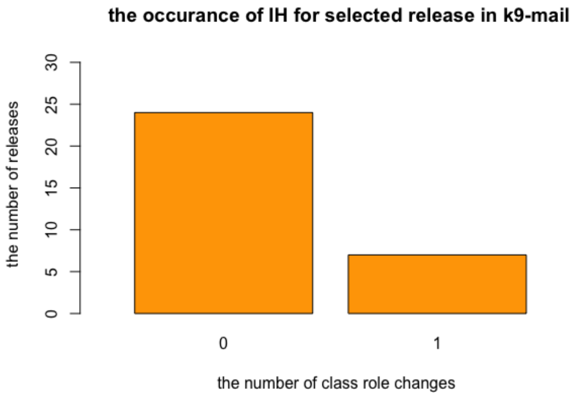


Fig 7: the distribution of" Information Holder" for selected releases in k9-mail

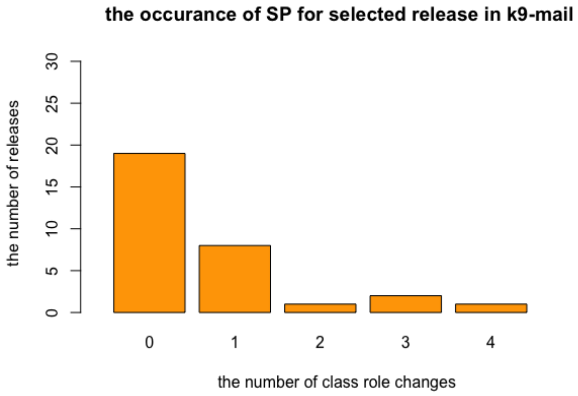


Fig 8: the distribution of "Service provider" for selected releases in k9-mail

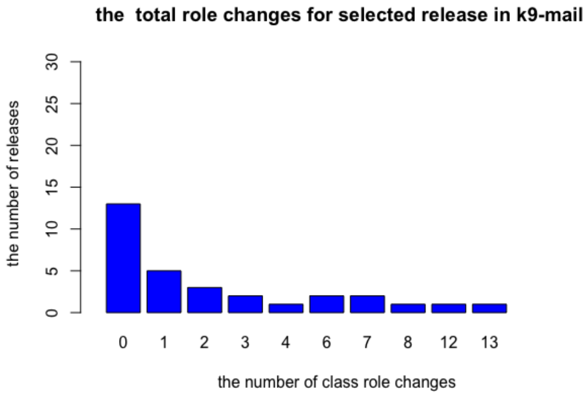


Fig 9: the distribution of "total class role changes" for selected releases in k9-mail

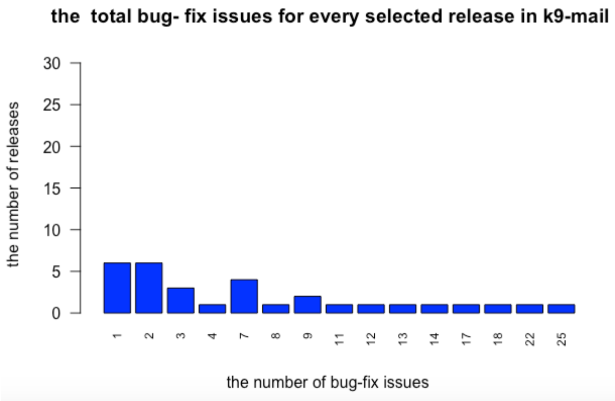


Fig 10: the distribution of" bug-fix issues for every selected release" in k9-mail

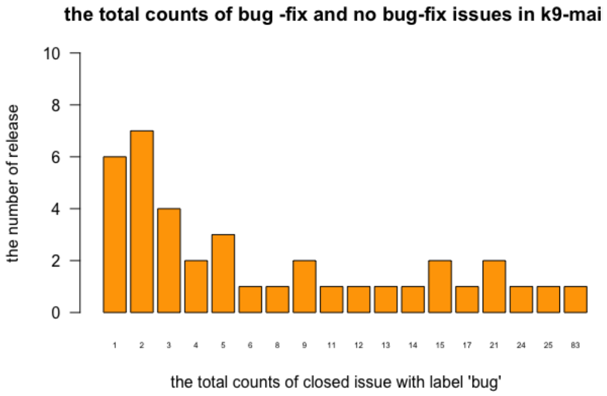
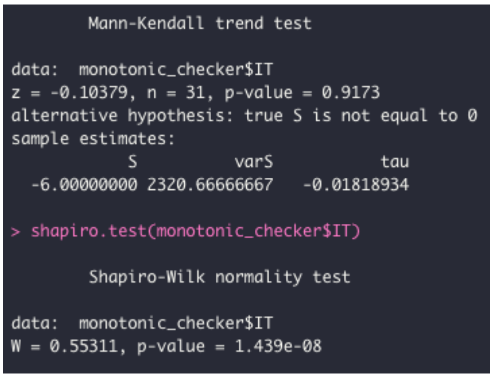
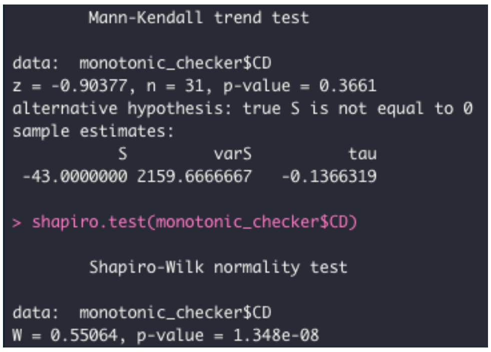


Fig11: the distribution of " total counts" for bug-fixed issues and none bug-fixed issues 'in selected releases from k9-mail



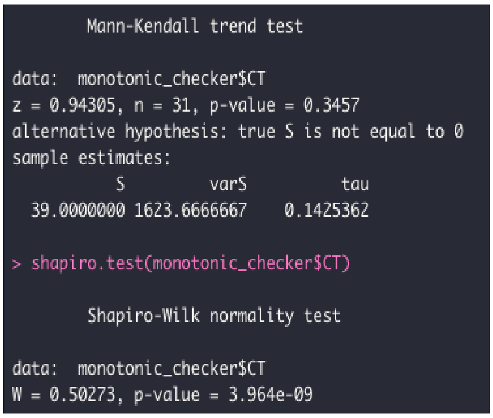
```
        Mann-Kendall trend test

data:  monotonic_checker$IT
z = -0.10379, n = 31, p-value = 0.9173
alternative hypothesis: true S is not equal to 0
sample estimates:
            S          varS           tau
  -6.00000000 2320.66666667   -0.01818934

> shapiro.test(monotonic_checker$IT)

        Shapiro-Wilk normality test

data:  monotonic_checker$IT
W = 0.55311, p-value = 1.439e-08
```

Fig 12: The results from both "Shapiro test" and "Mann-kendall test" for the column of " interfacer"



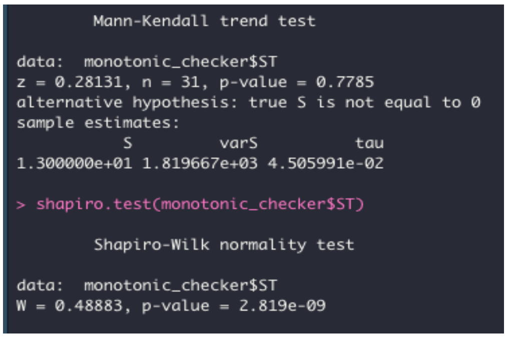
```
        Mann-Kendall trend test

data:  monotonic_checker$CD
z = -0.90377, n = 31, p-value = 0.3661
alternative hypothesis: true S is not equal to 0
sample estimates:
          S          varS           tau
 -43.0000000 2159.6666667   -0.1366319

> shapiro.test(monotonic_checker$CD)

        Shapiro-Wilk normality test

data:  monotonic_checker$CD
W = 0.55064, p-value = 1.348e-08
```

Fig 13: The results from both "Shapiro test and Mann-kendall test" for the column of " Coordinator"



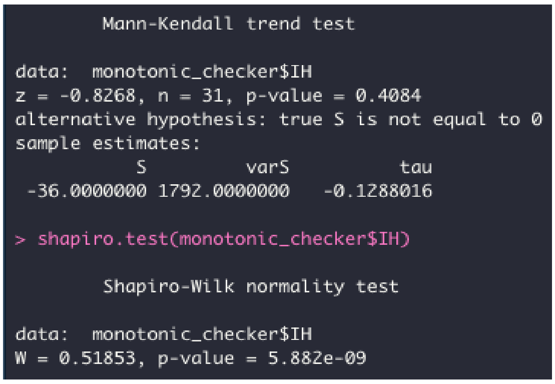
```
        Mann-Kendall trend test

data:  monotonic_checker$CT
z = 0.94305, n = 31, p-value = 0.3457
alternative hypothesis: true S is not equal to 0
sample estimates:
          S         varS          tau
  39.0000000 1623.6666667    0.1425362

> shapiro.test(monotonic_checker$CT)

        Shapiro-Wilk normality test

data:  monotonic_checker$CT
W = 0.50273, p-value = 3.964e-09
```

Fig 14: The results from both "Shapiro test and Mann-kendall test" for the column of " Controller"



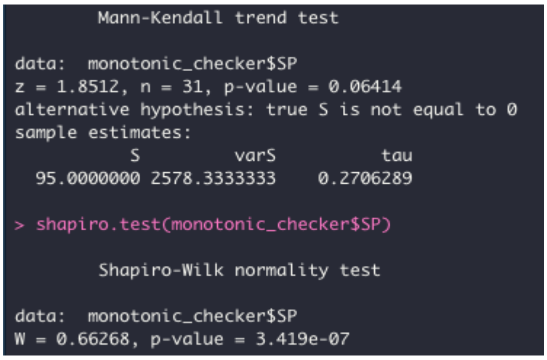
```
        Mann-Kendall trend test

data:  monotonic_checker$ST
z = 0.28131, n = 31, p-value = 0.7785
alternative hypothesis: true S is not equal to 0
sample estimates:
           S         varS          tau
1.300000e+01 1.819667e+03 4.505991e-02

> shapiro.test(monotonic_checker$ST)

        Shapiro-Wilk normality test

data:  monotonic_checker$ST
W = 0.48883, p-value = 2.819e-09
```

Fig 15: The results from both "Shapiro test and Mann-kendall test" for the column of " Structure"



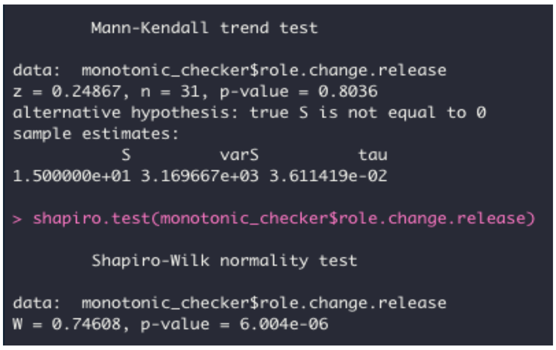
```
        Mann-Kendall trend test

data:  monotonic_checker$IH
z = -0.8268, n = 31, p-value = 0.4084
alternative hypothesis: true S is not equal to 0
sample estimates:
          S          varS           tau
 -36.0000000 1792.0000000   -0.1288016

> shapiro.test(monotonic_checker$IH)

        Shapiro-Wilk normality test

data:  monotonic_checker$IH
W = 0.51853, p-value = 5.882e-09
```

Fig 16: The results from both "Shapiro test and Mann-kendall test" for the column of " Information Holder"



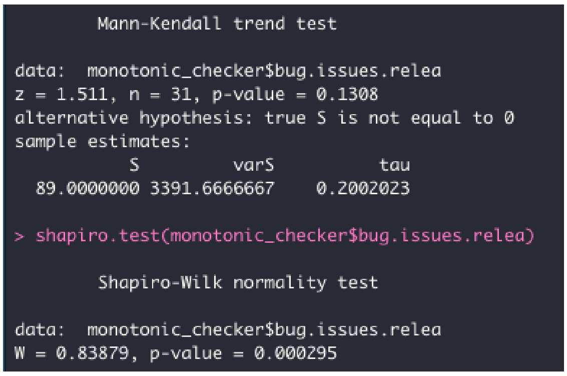
```
        Mann-Kendall trend test

data:  monotonic_checker$SP
z = 1.8512, n = 31, p-value = 0.06414
alternative hypothesis: true S is not equal to 0
sample estimates:
         S       varS        tau
  95.0000000 2578.3333333    0.2706289

> shapiro.test(monotonic_checker$SP)

        Shapiro-Wilk normality test

data:  monotonic_checker$SP
W = 0.66268, p-value = 3.419e-07
```

Fig 17: The results from both "Shapiro test and Mann-kendall test" for the column of " Service Provider"



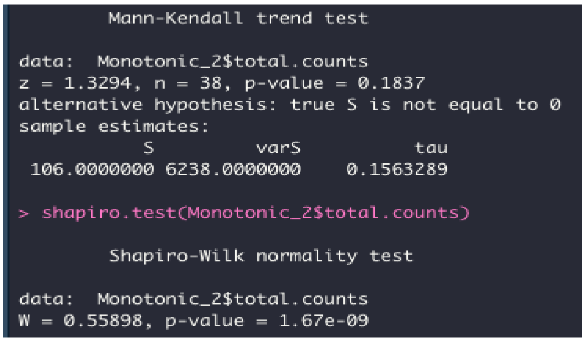
```
        Mann-Kendall trend test

data:  monotonic_checker$role.change.release
z = 0.24867, n = 31, p-value = 0.8036
alternative hypothesis: true S is not equal to 0
sample estimates:
           S          varS          tau
1.500000e+01 3.169667e+03 3.611419e-02

> shapiro.test(monotonic_checker$role.change.release)

        Shapiro-Wilk normality test

data:  monotonic_checker$role.change.release
W = 0.74608, p-value = 6.004e-06
```

Fig 18: The results from both "Shapiro test and Mann-kendall test" for the column of " role change/release"



```
        Mann-Kendall trend test

data:  monotonic_checker$bug.issues.relea
z = 1.511, n = 31, p-value = 0.1308
alternative hypothesis: true S is not equal to 0
sample estimates:
         S       varS        tau
  89.0000000 3391.6666667    0.2002023

> shapiro.test(monotonic_checker$bug.issues.relea)

        Shapiro-Wilk normality test

data:  monotonic_checker$bug.issues.relea
W = 0.83879, p-value = 0.000295
```

Fig 19: The results from both "Shapiro test and Mann-kendall test" for the column of " bug-fix issues/release"



```
        Mann-Kendall trend test

data:  Monotonic_2$total.counts
z = 1.3294, n = 38, p-value = 0.1837
alternative hypothesis: true S is not equal to 0
sample estimates:
          S        varS         tau
 106.0000000 6238.0000000    0.1563289

> shapiro.test(Monotonic_2$total.counts)

        Shapiro-Wilk normality test

data:  Monotonic_2$total.counts
W = 0.55898, p-value = 1.67e-09
```

Fig 20: The results from both "Shapiro test and Mann-kendall test" for the column of "total counts "of issues with label “bug”

Both The table “k9-change-names.csv” and the Table “k9-change-numers.csv “ don’t attached to this paper!

Table II-- Bug-fix issues over selected releases in K9-mail

Issue ID	Date of created	Date of closed	bug-fix	none bug-fix	commit ID	commit date	release	bug issues/relea
2433	23/03/2017	25/03/2017	Y		3a81ccf	04/02/2013	4.322	1
828	03/10/2015	03/10/2015	Y		9c7776d	17/12/2014	5.002	1
609	13/04/2015	29/08/2015	Y		065088f	13/09/2015	5.007	11
616	16/04/2015	06/07/2015	Y		d8aef84	06/07/2015	5.007	
618	21/04/2015	28/04/2015	Y		d538278	28/04/2015	5.007	
690	18/06/2015	07/09/2015	Y		a6b9384	7/9/2015	5.007	
744	11/08/2015	13/09/2015	Y		065088f	13/09/2015	5.007	
770	02/09/2015	03/10/2015	Y		e76c489	28/09/2015	5.007	
786	08/09/2015	11/10/2015	Y		de401db	02/10/2015	5.007	
933	04/12/2015	11/12/2015	Y		e6c52d3	11/10/2015	5.007	
765	28/08/2015	13/09/2015	Y		696579f	13/09/2015	5.007	
809	22/09/2015	03/10/2015	Y		e76c489	28/9/2015	5.007	
818	26/09/2015	27/09/2015	Y		d84ce23	27/09/2015	5.007	
864	24/10/2015	13/01/2016	Y		a7c9b80	09/01/2016	5.008	6
871	25/10/2015	13/01/2016	Y		4b52312	13/01/2016	5.008	
899	17/11/2015	12/01/2016	Y		f018902	12/01/2016	5.008	
926	02/12/2015	12/12/2015	Y		acc2e42	12/12/2015	5.008	
969	21/12/2015	07/01/2016	Y		e1ca89b	03/01/2016	5.008	
640	11/05/2015	08/01/2016	Y		a0a362	16/12/2015	5.008	
615	14/04/2015	02/04/2016	Y		15a44ce	02/04/2016	5.009	1
807	21/09/2015	13/02/2016	Y		06e1777	13/02/2016	5.108	7
811	24/09/2015	03/03/2016	Y		150fc82	03/03/2016	5.108	
1050	29/01/2016	16/02/2016	Y		2a8b855	16/02/2016	5.108	
1110	21/02/2016	02/03/2016	Y		fcbfc4d	2/3/2016	5.108	
1143	05/03/2016	11/03/2016	Y		eb31a0f	11/3/2016	5.108	
1150	05/03/2016	11/03/2016	Y		3491f99	11/03/2016	5.108	
1151	05/03/2016	11/03/2016	Y		41bfaf2	08/03/2016	5.108	
746	12/08/2015	13/08/2016	Y		ef04d07	13/04/2016	5.109	9
916	28/11/2015	09/04/2016	Y		186ed1b	23/03/2016	5.109	
1139	03/03/2016	24/03/2016	Y		d93a7de	24/03/2016	5.109	
1164	09/03/2016	29/03/2016	Y		245deef	24/03/2016	5.109	
1204	23/03/2016	23/03/2016	Y		91c60a4	23/03/2016	5.109	
1224	28/03/2016	12/04/2016	Y		46dd8c7	08/04/2016	5.109	
1227	29/03/2016	02/04/2016	Y		fd89879	01/04/2016	5.109	
1250	03/04/2016	09/04/2016	Y		51b310c	06/04/2016	5.109	

Table II-- Bug-fix issues over selected releases in K9-mail

1252	03/04/2016	19/05/2016	Y		7614c8f	15/05/2016	5.109	
1275	11/04/2016	28/04/2016	Y		b160e21	22/04/2016	5.110	2
1277	12/04/2016	13/04/2016	Y		2d67b49	12/04/2016	5.110	
1293	16/04/2016	18/04/2016	Y		050316e	01/07/2016	5.111	9
1297	17/04/2016	28/02/2019	Y		534d75d	21/04/2016	5.111	
1369	11/05/2016	21/05/2016	Y		43899da	20/05/2016	5.111	
732	04/08/2015	07/07/2016	Y		7ebf79c	01/07/2016	5.111	
1271	11/04/2016	01/07/2016	Y		240c5c8	30/06/2016	5.111	
819	28/09/2015	05/08/2016	Y		915f44a	05/08/2016	5.111	
1201	22/03/2016	30/08/2016	Y		921ee5c	25/07/2016	5.111	
1206	23/03/2016	29/08/2016	Y		96d210c	19/08/2016	5.111	
1251	03/04/2016	24/07/2016	Y		034b1ed	22/07/2016	5.111	
1394	19/05/2016	10/08/2016	Y		b40d64e	10/08/2016	5.112	3
1495	10/07/2016	02/08/2016	Y		b3f2974	01/08/2016	5.112	
1504	15/07/2016	25/07/2016	Y		8b1c13d	24/07/2016	5.112	
1581	27/08/2016	10/09/2016	Y		4d591a7	10/09/2016	5.114	14
1582	30/08/2016	30/08/2016	Y		5a17768	30/08/2016	5.114	
1604	07/09/2016	07/10/2016	Y		5c0a7f6	05/10/2016	5.114	
1607	09/09/2016	13/09/2016	Y		192ce7e	11/09/2016	5.114	
1243	31/03/2016	12/10/2016	Y		aaa904e	12/10/2016	5.114	
1609	11/09/2016	11/10/2016	Y		7a0bacf	11/10/2016	5.114	
1625	21/09/2016	07/10/2016	Y		0cd52bc	07/10/2016	5.114	
1629	23/09/2016	08/10/2016	Y		fc79b29	08/10/2016	5.114	
1662	04/10/2016	08/10/2016	Y		302b668	08/10/2016	5.114	
1665	05/10/2016	06/10/2016	Y		88eb0f6	08/10/2016	5.114	
1666	05/10/2016	08/10/2016	Y		88eb0f6	8/10/2016	5.114	
1685	07/10/2016	12/10/2016	Y		2087f04	11/10/2016	5.114	
1699	11/10/2016	11/10/2016	Y		b01f49b	11/10/2016	5.114	
1245	01/04/2016	27/01/2017	Y		302b668	08/10/2016	5.114	
1951	04/01/2017	28/02/2019	Y		cedaecb	08/11/2016	5.115	1
1826	01/12/2016	11/12/2016	Y		5fca3c8	11/12/2016	5.200	1
1644	29/09/2016	08/11/2016	Y		cedaecb	08/11/2016	5.201	4
1660	03/10/2016	19/10/2016	Y		06e1647	19/10/2016	5.201	
1874	28/12/2016	30/12/2016	Y		50d81f	30/12/2016	5.201	
1878	28/12/2016	30/12/2016	Y		b9147f1	30/12/2016	5.201	
1919	02/01/2017	03/01/2017	Y		a56f12f	03/01/2017	5.202	3
1930	02/01/2017	08/01/2017	Y		1af2f23	08/01/2017	5.202	

Table II-- Bug-fix issues over selected releases in K9-mail

1932	03/01/2017	10/01/2018	Y		fc6d518	10/02/2017	5.202	
1741	21/10/2016	31/10/2016	Y		d0b3caf	31/10/2016	5.203	8
1917	02/01/2017	10/01/2017	Y		217c614	10/01/2017	5.203	
1938	03/01/2017	05/01/2017	Y		a452ded	05/01/2017	5.203	
1950	04/01/2017	15/01/2017	Y		bb514f7	11/01/2017	5.203	
1960	04/01/2017	19/01/2017	Y		f5e837c	15/01/2017	5.203	
1981	05/01/2017	08/01/2017	Y		aef446f	08/01/2017	5.203	
1984	05/01/2017	16/02/2017	Y		3490a7f	09/01/2017	5.203	
2010	07/01/2017	17/01/2017	Y		3b83b18	17/01/2017	5.203	
1762	28/10/2016	06/11/2016	Y		d0b3caf	31/10/2016	5.204	13
1836	09/12/2016	09/12/2016	Y		87e13ef	9/12/2016	5.204	
711	14/06/2015	18/01/2017	Y		dbb5180	12/1/2017	5.204	
979	27/12/2015	22/02/2017	Y		c150baf	12/02/2017	5.204	
1240	31/03/2016	06/02/2017	Y		8c55e57	5/2/2017	5.204	
1500	12/07/2016	15/02/2017	Y		329ed78	15/02/2017	5.204	
1875	28/12/2016	21/01/2017	Y		e238ee5	21/01/2017	5.204	
810	22/09/2015	28/02/2019	Y		c60f97f	31/01/2017	5.204	
1998	06/01/2017	28/02/2019	Y		8c55e57	05/02/2017	5.204	
2103	21/01/2017	26/01/2017	Y		9e102a5	26/01/2017	5.204	
2121	23/01/2017	25/01/2017	Y		bf881cd	25/01/2017	5.204	
2134	25/01/2017	31/01/2017	Y		3bd84de	31/01/2017	5.204	
2143	27/01/2017	13/02/2017	Y		8ee9b2c	13/02/2017	5.204	
1653	01/10/2016	28/02/2019	Y		168f9a8	09/02/2017	5.205	2
1822	28/11/2016	05/02/2017	Y		8c55e57	05/02/2017	5.205	
1141	04/03/2016	28/02/2019	Y		88a86a1	3/3/2017	5.206	2
1476	27/06/2016	27/04/2017	Y		9d079bd	28/02/2017	5.206	
1223	28/03/2016	22/03/2017	Y		32212a4	22/3/2017	5.207	3
1857	20/12/2016	03/04/2017	Y		06b0f7d	02/04/2017	5.207	
1418	30/05/2016	26/05/2017	Y		6520f3a	26/03/2017	5.207	
1879	28/12/2016	30/12/2016	Y		3bee80a	30/12/2016	5.300	17
1889	30/12/2016	07/02/2017	Y		6738b49	26/01/2017	5.300	
1893	31/12/2016	31/12/2016	Y		df9009e	31/12/2016	5.300	
1899	31/12/2016	19/01/2017	Y		dc38b6d	19/01/2017	5.300	
1901	31/12/2016	15/01/2017	Y		f5e837c	15/01/2017	5.300	
1908	01/01/2017	10/01/2017	Y		92196c0	10/01/2017	5.300	
1914	02/01/2017	04/01/2017	Y		b516af2	04/01/2017	5.300	
1915	02/01/2017	05/01/2017	Y		ea699b3	05/01/2017	5.300	

Table II-- Bug-fix issues over selected releases in K9-mail

1959	04/01/2017	05/01/2017	Y		6beb990	05/01/2017	5.300	
1965	04/01/2017	08/01/2017	Y		985cd85	08/01/2017	5.300	
2015	08/01/2017	16/01/2018	Y		c816276	25/05/2017	5.300	
2044	12/01/2017	21/01/2017	Y		3e8ad4b	15/01/2017	5.300	
2057	15/01/2017	21/01/2018	Y		b5cf015	27/03/2017	5.300	
2083	18/01/2017	18/01/2017	Y		4b745ca	18/01/2017	5.300	
2148	28/01/2017	25/05/2017	Y		c816276	25/05/2017	5.300	
2503	17/04/2017	02/05/2017	Y		754837d	02/05/2017	5.300	
2605	27/06/2017	15/08/2018	Y		3700e20d4d	17/08/2017	5.300	
2699	23/08/2017	07/04/2020	Y		de2f772	06/09/2017	5.300	
697	24/06/2015	14/10/2017	Y		0b480d7	14/10/2017	5.301	12
2282	26/02/2017	14/03/2017	Y		b901b81	14/03/2017	5.301	
2337	04/03/2017	12/06/2017	Y		b5cf015	23/03/2017	5.301	
2475	02/04/2017	14/10/2017	Y		0b480d7	14/10/2017	5.301	
2602	26/06/2017	25/10/2017	Y		54d4a8e	09/09/2017	5.301	
2708	27/08/2017	29/08/2017	Y		8639664	29/08/2017	5.301	
2758	11/09/2017	16/09/2017	Y		e266547	16/09/2017	5.301	
2765	13/09/2017	01/03/2019	Y		8fabd3e7a0	14/09/2017	5.301	
2766	13/09/2017	01/03/2019	Y		6b8e452	14/09/2017	5.301	
2788	24/09/2017	01/03/2019	Y		d9789e9	14/10/2017	5.301	
3847	27/12/2018	07/10/2020	Y		b79673b	05/02/2017	5.301	
3848	27/12/2018	07/10/2020	Y		b79673b	05/02/2017	5.301	
701	29/06/2015	26/01/2018	Y		9d90c53	25/01/2018	5.500	22
1988	05/01/2017	17/12/2017	Y		a95e897	17/12/2017	5.500	
2572	07/06/2017	21/01/2018	Y		e9d90b1	05/06/2018	5.500	
2846	18/10/2017	01/11/2017	Y		0a6ef2b	01/11/2017	5.500	
2847	18/10/2017	28/10/2017	Y		24de0df	28/10/2017	5.500	
2856	24/10/2017	28/10/2017	Y		23b903e	28/10/2017	5.500	
2861	25/10/2017	28/10/2017	Y		9fdcf44	28/10/2017	5.500	
2891	01/11/2017	02/11/2017	Y		f366e50	02/11/2017	5.500	
2941	16/11/2017	27/11/2017	Y		dd9639c	27/11/2017	5.500	
2949	25/11/2017	29/01/2018	Y		11fae34	12/01/2018	5.500	
2962	05/12/2017	25/01/2018	Y		a36254d	25/01/2018	5.500	
2966	08/12/2017	01/03/2019	Y		d503190	29/12/2017	5.500	
2973	16/12/2017	08/01/2018	Y		d0c8cc3	08/01/2018	5.500	
2983	22/12/2017	04/01/2018	Y		023caaa	04/01/2018	5.500	
2999	29/12/2017	06/01/2018	Y		e9d90b1	06/01/2018	5.500	

Table II-- Bug-fix issues over selected releases in K9-mail

3004	30/12/2017	28/01/2018	Y		1962def	28/01/2018	5.500	
3006	30/12/2017	04/01/2018	Y		118b465	04/01/2018	5.500	
3011	31/12/12017	11/01/2018	Y		b5cffe8	11/01/2018	5.500	
3018	01/01/2018	04/01/2018	Y		11fae34	02/01/2018	5.500	
3032	03/01/2018	04/01/2018	Y		f5c9ae4	04/01/2018	5.500	
3052	06/01/2018	04/01/2018	Y		f69ac06	04/01/2018	5.500	
3065	08/01/2018	11/01/2018	Y		c95f7f7	11/01/2018	5.500	
1220	27/03/2016	28/02/2019	Y		2ec44b6	24/02/2018	5.501	7
2188	05/02/2017	07/02/2017	Y		4c8dd42	07/02/2017	5.501	
2222	10/02/2017	21/01/2018	Y		d6090c6	12/11/2017	5.501	
3121	22/01/2018	26/01/2018	Y		affc41c	25/01/2018	5.501	
3125	23/01/2018	25/01/2018	Y		26f6963	25/01/2018	5.501	
3129	24/01/2018	25/01/2018	Y		c24c3ae	25/01/2018	5.501	
3215	25/02/2018	25/02/2018	Y		1618b6f	25/02/2018	5.501	
632	29/04/2015	26/02/2018	Y		49257b0	27/2/2018	5.502	1
633	29/04/2015	28/04/2019	Y		310600d	30/03/2018	5.503	1
2164	01/02/2017	17/02/2018	Y		1645c38	17/02/2018	5.600	2
3289	28/03/2018	29/03/2018	Y		46a51f1	31/03/2018	5.600	
890	10/11/2015	27/08/2018	Y		e65daf5	14/2/2019	5.700	25
1619	19/09/2016	03/09/2018	Y		83b6ab0	3/9/2018	5.700	
2538	16/05/2017	24/08/2017	Y		e65daf5	14/02/2019	5.700	
2756	11/09/2017	07/03/2019	Y		33e7456	15/08/2018	5.700	
3138	26/01/2018	08/04/2018	Y		2aa4041	08/04/2018	5.700	
3255	13/03/2018	23/06/2018	Y		bf33cfd	23/06/2018	5.700	
3265	15/03/2018	16/04/2018	Y		ffccd9b	16/04/2018	5.700	
3616	14/09/2018	27/03/2019	Y		e65daf5	14/02/2019	5.700	
3786	02/12/2018	23/12/2018	Y		e3d193c	23/12/2018	5.700	
3787	02/12/2018	03/12/2018	Y		62411ac	03/12/2018	5.700	
3801	05/12/2018	07/10/2020	Y		f1963ae	25/01/2019	5.700	
3803	07/12/2018	06/01/2019	Y		2cb299d	06/01/2019	5.700	
3811	10/12/2018	12/12/2018	Y		ecfbbca	12/12/2018	5.700	
3832	18/12/2018	23/12/2018	Y		f0b12e5	23/12/2018	5.700	
3866	07/01/2019	08/03/2019	Y		c3bcf50	08/03/2019	5.700	
3880	16/01/2019	01/12/2019	Y		5871726	16/12/2018	5.700	
3998	01/04/2019	28/10/2019	Y		457f27e	28/10/2019	5.700	
4008	06/04/2019	06/10/2020	Y		f1963ae	25/01/2019	5.700	
4016	10/04/2019	01/05/2019	Y		af6550d	23/04/2019	5.700	

Table II-- Bug-fix issues over selected releases in K9-mail

4121	23/07/2019	13/10/2019	Y		7dadab7	13/10/2019	5.700	
4153	12/08/2019	31/08/2019	Y		a8f4d33	31/08/2019	5.700	
4160	15/08/2019	31/08/2019	Y		2543711	31/08/2019	5.700	
4201	23/09/2019	06/11/2019	Y		0168789	06/11/2019	5.700	
4248	12/11/2019	20/11/2019	Y		fe76cc9	20/11/2019	5.700	
4250	14/11/2019	16/11/2019	Y		dcb9130	16/11/2019	5.700	
3861	06/01/2019	09/01/2019	Y		88c1232	09/01/2019	5.701	2
3862	06/01/2019	09/01/2019	Y		45bf82b	09/01/2019	5.701	
723	25/07/2015	12/12/2019	Y		bb845e0	12/12/2019	5.702	7
3515	20/07/2018	22/07/2018	Y		5e9dfa3	22/07/2018	5.702	
3652	08/10/2018	28/11/2018	Y		67df429	28/11/2018	5.702	
4296	27/11/2019	14/12/2019	Y		4d91d8e	02/12/2019	5.702	
4301	28/11/2019	02/12/2019	Y		f443835	02/12/2019	5.702	
4304	28/11/2019	12/12/2019	Y		bf3f1a6	12/12/2019	5.702	
4341	04/12/2019	10/12/2019	Y		15a0bed	10/12/2019	5.702	
3111	18/01/2018	17/12/2019	Y		5a0aa15	17/12/2019	5.703	7
3254	12/03/2018	07/02/2020	Y		5a0aa15	07/12/2019	5.703	
3303	04/04/2018	29/06/2018	Y		c1a5a60	29/06/2018	5.703	
3685	29/10/2018	27/12/2019	Y		5a0aa15	17/12/2019	5.703	
4333	03/12/2019	22/12/2019	Y		b5df319	22/12/2019	5.703	
4359	12/12/2019	17/12/2019	Y		615cad7	17/12/2019	5.703	
4379	15/12/2019	19/12/2019	Y		617624c	19/12/2019	5.703	
1074	07/02/2016	02/04/2020	Y		68213ac	2/4/2020	5.709	18
2023	09/01/2017	21/01/2018	Y		5a0aa15	17/12/2019	5.709	
2552	26/05/2017	21/01/2018	y		5a0aa15	17/12/2019	5.709	
3266	16/03/2018	08/01/2020	Y		ad39ac2	08/01/2020	5.709	
3957	09/03/2019	15/03/2020	Y		e98d350	15/03/2020	5.709	
4340	04/12/2019	24/01/2020	Y		66ac635	04/03/2020	5.709	
4342	04/12/2019	03/03/2020	Y		c041a2e	03/03/2020	5.709	
4393	19/12/2019	03/03/2020	Y		d2c6770	03/03/2020	5.709	
4435	10/01/2020	18/04/2020	Y		45a7942	18/04/2020	5.709	
4436	10/01/2020	14/01/2020	Y		12ddaec	14/01/2020	5.709	
4452	15/01/2020	06/02/2020	Y		aa4f1fd	06/02/2020	5.709	
4453	15/01/2020	13/04/2020	Y		1ec1a37	13/04/2020	5.709	
4472	23/01/2020	26/01/2020	Y		2afacbc	26/01/2020	5.709	
4498	02/02/2020	16/02/2020	Y		66b4990	16/02/2020	5.709	
4519	08/02/2020	08/02/2020	Y		8f4a287	08/02/2020	5.709	

Table II-- Bug-fix issues over selected releases in K9-mail

4539	15/02/2020	04/03/2020	Y		7ff55ed	04/03/2020	5.709	
4610	15/03/2020	15/03/2020	Y		e461f73	15/03/2020	5.709	
4620	21/03/2020	04/04/2020	Y		3f60e41	04/04/2020	5.709	
2705	25/08/2017	24/04/2020	Y		f9bbeec	24/04/2020	5.710	3
4678	20/04/2020	24/04/2020	Y		532b94b	24/04/2020	5.710	
4685	23/04/2020	24/04/2020	Y		626f8e1	24/04/2020	5.710	
3691	02/11/2018	06/05/2020	Y		664e444	06/05/2020	5.711	4
4100	07/07/2019	28/04/2020	Y		db34c3e	28/04/2020	5.711	
4374	14/12/2019	11/10/2020	Y		496dac7	03/05/2020	5.711	
4708	28/04/2020	28/04/2020	Y		0dabe18	28/04/2020	5.711	
4738	08/05/2020	08/05/2020	Y		633fee4	08/05/2020	5.712	1
4622	22/03/2020	19/06/2020	Y		48a76d5	19/06/2020	5.717	1
2136	25/01/2017	03/07/2020	Y		0c40a77	03/07/2020	5.718	1
3653	09/10/2018	23/09/2020	Y		67df429	23/09/2020	5.719	1
3357	24/04/2018	07/10/2020	Y		b76d112	07/10/2020	5.721	1
3767	30/11/2018	08/10/2020	Y		2c95b7d	09/10/2020	5.722	2
4360	12/12/2019	13/10/2020	Y		57bde56	13/10/2020	5.722	
3971	17/03/2019	06/01/2021	Y		1259d37	06/01/2021	5.726	1
4125	24/07/2019	16/02/2021	Y		ad83acb	16/02/2021	5.730	1
4412	27/12/2019	11/05/2021	Y		45a7942	11/05/2021	5.735	1
913	22/11/2015	28/02/2019		Y				
583	19/03/2015	17/01/2016		Y				
597	29/03/2015	16/09/2016		Y				
598	29/03/2015	02/05/2015		Y				
605	05/05/2015	05/10/2020		Y				
614	14/04/2015	13/04/2016		Y				
639	09/05/2015	02/12/2015		Y				
642	13/05/2015	24/05/2015		Y				
659	26/05/2015	21/06/2015		Y				
660	17/05/2015	16/06/2015		Y				
663	27/05/2015	13/10/2016		Y				
673	07/06/2015	10/06/2015		Y				
710	11/07/2015	28/02/2017		Y				
716	17/07/2015	28/02/2019		Y				
721	24/07/2015	28/02/2019		Y				
724	27/07/2015	17/01/2016		Y				
730	03/08/2015	22/03/2016		Y				

Table II-- Bug-fix issues over selected releases in K9-mail

749	14/08/2015	18/07/2016		Y				
766	29/08/2015	05/10/2020		Y				
771	03/09/2015	27/09/2015		Y				
783	07/09/2015	05/10/2020		Y				
813	25/09/2015	10/10/2015		Y				
814	25/09/2015	30/03/2016		Y				
815	26/09/2015	15/01/2016		Y				
839	10/10/2015	17/01/2016		Y				
851	19/10/2015	17/01/2016		Y				
854	20/10/2015	23/10/2015		Y				
858	23/10/2015	17/01/2016		Y				
876	31/10/2015	28/02/2016		Y				
886	05/11/2015	20/11/2015		Y				
889	09/11/2015	17/12/2015		Y				
901	19/11/2015	17/01/2016		Y				
941	09/12/2015	17/01/2016		Y				
946	11/12/2015	17/01/2016		Y				
961	17/12/2015	09/02/2017		Y				
966	20/12/2015	13/04/2016		Y				
970	21/12/2015	08/03/2019		Y				
971	21/12/2015	28/02/2019		Y				
990	01/01/2016	02/01/2016		Y				
1008	08/01/2016	09/02/2017		Y				
1024	17/01/2016	17/01/2016		Y				
1039	26/01/2016	17/11/2018		Y				
1047	28/01/2016	05/10/2020		Y				
1063	02/02/2016	02/11/2017		Y				
1065	03/02/2016	12/04/2016		Y				
1079	09/02/2016	12/03/2016		Y				
1100	17/02/2016	14/04/2016		Y				
1130	28/02/2016	09/09/2020		Y				
1140	04/03/2016	06/10/2016		Y				
1152	05/03/2016	05/10/2020		Y				
1155	06/03/2016	28/02/2019		Y				
1161	08/03/2016	11/03/2016		Y				
1176	14/03/2016	14/03/2016		Y				
1236	31/03/2016	27/08/2018		Y				

Table II-- Bug-fix issues over selected releases in K9-mail

1244	01/04/2016	28/02/2019		Y				
1254	04/04/2016	04/04/2016		Y				
1268	08/04/2016	08/04/2016		Y				
1272	11/04/2016	28/02/2019		Y				
1283	14/04/2016	28/02/2019		Y				
1326	24/04/2016	05/10/2020		Y				
1368	10/05/2016	08/02/2017		Y				
1401	24/05/2016	09/02/2017		Y				
1408	26/05/2016	17/06/2016		Y				
1422	01/06/2016	05/10/2020		Y				
1431	05/06/2016	28/06/2016		Y				
1434	05/06/2016	15/10/2017		Y				
1437	05/06/2016	05/10/2020		Y				
1452	09/06/2016	28/02/2019		Y				
1454	10/06/2016	05/10/2020		Y				
1455	11/06/2016	28/02/2019		Y				
1461	14/06/2016	17/10/2017		Y				
1468	20/06/2016	05/10/2020		Y				
1473	25/06/2016	19/06/2017		Y				
1494	08/07/2016	05/10/2020		Y				
1505	16/07/2016	01/08/2016		Y				
1507	17/07/2016	02/03/2017		Y				
1541	03/08/2016	09/02/2017		Y				
1543	04/08/2016	28/02/2019		Y				
1547	05/08/2016	08/07/2020		Y				
1561	13/08/2016	09/02/2017		Y				
1566	16/08/2016	28/02/2019		Y				
1575	24/08/2016	30/08/2016		Y				
1599	05/09/2016	28/02/2019		Y				
1602	07/09/2016	05/10/2020		Y				
1623	20/09/2016	28/02/2019		Y				
1627	21/09/2016	28/02/2019		Y				
1651	30/09/2016	28/02/2019		Y				
1655	01/10/2016	19/01/2018		Y				
1658	03/10/2016	20/10/2016		Y				
1659	03/10/2016	26/08/2018		Y				
1663	05/10/2016	28/02/2019		Y				

Table II-- Bug-fix issues over selected releases in K9-mail

1667	05/10/2016	28/02/2019		Y				
1671	05/10/2016	28/02/2019		Y				
1673	06/10/2016	28/02/2019		Y				
1690	07/10/2016	05/10/2020		Y				
1697	10/10/2016	04/01/2017		Y				
1721	17/10/2016	28/02/2019		Y				
1733	19/10/2016	20/10/2016		Y				
1748	24/10/2016	28/02/2019		Y				
1765	30/10/2016	05/10/2020		Y				
1772	05/11/2016	05/10/2020		Y				
1809	22/11/2016	01/04/2020		Y				
1831	05/12/2016	28/02/2019		Y				
1864	22/12/2016	28/02/2019		Y				
1865	22/12/2016	28/02/2019		Y				
1881	28/12/2016	29/12/2016		Y				
1910	02/01/2017	07/02/2017		Y				
1933	03/01/2017	03/01/2017		Y				
1934	03/01/2017	03/01/2017		Y				
1940	03/01/2017	07/02/2017		Y				
1991	06/01/2017	28/02/2019		Y				
1992	06/01/2017	07/01/2017		Y				
1997	06/01/2017	06/02/2017		Y				
2016	08/01/2017	05/10/2020		Y				
2033	10/01/2017	05/10/2020		Y				
2034	10/01/2017	28/02/2019		Y				
2039	11/01/2017	21/01/2017		Y				
2060	15/01/2017	28/02/2019		Y				
2090	18/01/2017	06/02/2017		Y				
2117	22/01/2017	06/02/2017		Y				
2147	28/01/2017	28/02/2019		Y				
2310	02/03/2017	28/02/2019		Y				
2331	03/03/2017	08/03/2017		Y				
2336	04/03/2017	05/10/2020		Y				
2357	07/03/2017	01/03/2019		Y				
2414	21/03/2017	01/03/2019		Y				
2479	04/04/2017	25/03/2019		Y				
2507	20/04/2017	26/06/2017		Y				

Table II-- Bug-fix issues over selected releases in K9-mail

2543	23/05/2017	06/03/2019		Y				
2568	05/06/2017	01/03/2019		Y				
2569	05/06/2017	28/10/2017		Y				
2604	27/06/2017	05/10/2020		Y				
2616	04/07/2017	05/10/2020		Y				
2627	15/07/2017	01/03/2019		Y				
2647	28/07/2017	30/07/2017		Y				
2648	28/07/2017	01/03/2019		Y				
2717	30/04/2017	31/08/2017		Y				
2784	20/09/2017	01/03/2019		Y				
2814	04/10/2017	23/10/2020		Y				
2815	04/10/2017	01/03/2019		Y				
2829	09/10/2017	05/10/2020		Y				
2858	24/10/2017	25/10/2017		Y				
2968	09/12/2017	05/10/2020		Y				
2990	26/10/2017	01/03/2019		Y				
3009	31/12/2017	19/01/2018		Y				
3013	31/12/2017	01/03/2019		Y				
3047	05/01/2018	01/03/2019		Y				
3230	02/03/2018	06/10/2020		Y				
3239	06/03/2018	01/03/2019		Y				
3252	11/03/2018	06/10/2020		Y				
3262	15/03/2018	01/03/2019		Y				
3281	25/03/2018	06/10/2020		Y				
3293	29/03/2018	29/03/2018		Y				
3306	05/04/2018	06/10/2020		Y				
3308	06/04/2018	06/10/2020		Y				
3351	20/04/2018	06/10/2020		Y				
3355	22/04/2018	06/10/2020		Y				
3356	23/04/2018	06/10/2020		Y				
3393	16/05/2018	19/12/2019		Y				
3401	19/05/2018	06/10/2020		Y				
3404	22/05/2018	06/10/2020		Y				
3408	23/05/2018	06/10/2020		Y				
3411	25/05/2018	06/10/2020		Y				
3420	31/05/2018	06/10/2020		Y				
3427	04/06/2018	06/10/2020		Y				

Table II-- Bug-fix issues over selected releases in K9-mail

3429	06/06/2018	06/10/2020		Y				
3431	06/06/2018	08/03/2019		Y				
3434	08/06/2018	06/10/2020		Y				
3450	17/06/2018	06/10/2020		Y				
3451	17/06/2018	07/07/2020		Y				
3493	09/07/2018	08/03/2019		Y				
3496	10/07/2018	06/10/2020		Y				
3517	21/07/2018	06/10/2020		Y				
3532	27/07/2018	06/10/2020		Y				
3540	02/08/2018	08/03/2019		Y				
3568	24/08/2018	08/03/2019		Y				
3577	30/08/2018	06/10/2020		Y				
3605	09/09/2018	06/10/2020		Y				
3611	12/09/2018	08/03/2019		Y				
3614	13/09/2018	07/10/2020		Y				
3626	24/09/2018	07/10/2020		Y				
3631	30/09/2018	07/10/2020		Y				
3634	01/10/2018	07/10/2020		Y				
3642	03/10/2018	07/10/2020		Y				
3657	11/10/2018	08/03/2019		Y				
3659	12/10/2018	07/10/2020		Y				
3663	18/10/2018	07/10/2020		Y				
3669	20/10/2018	07/10/2020		Y				
3681	24/10/2018	07/10/2020		Y				
3683	27/10/2018	07/10/2020		Y				
3689	31/10/2018	07/10/2020		Y				
3690	01/11/2018	07/10/2020		Y				
3703	11/11/2018	07/07/2020		Y				
3708	12/11/2018	06/10/2020		Y				
3712	14/11/2018	07/10/2020		Y				
3713	14/11/2018	07/10/2020		Y				
3731	20/11/2018	08/03/2019		Y				
3781	02/12/2018	07/10/2020		Y				
3816	12/12/2018	07/10/2020		Y				
3900	04/02/2019	07/10/2020		Y				
3903	06/02/2019	06/10/2020		Y				
3938	03/03/2019	24/04/2020		Y				

Table II-- Bug-fix issues over selected releases in K9-mail

3944	05/03/2019	13/03/2019		Y				
3953	08/03/2019	09/03/2019		Y				
3974	20/03/2019	21/03/2019		Y				
4017	10/04/2019	07/10/2020		Y				
4022	16/04/2019	20/04/2020		Y				
4039	06/05/2019	07/10/2020		Y				
4048	13/05/2019	07/10/2020		Y				
4050	15/05/2019	07/10/2020		Y				
4079	30/05/2019	17/12/2019		Y				
4082	13/06/2019	12/10/2019		Y				
4167	19/08/2019	09/10/2020		Y				
4169	22/08/2019	09/10/2020		Y				
4175	30/08/2019	16/10/2019		Y				
4202	23/09/2019	11/05/2021		Y				
4298	28/11/2019	28/11/2019		Y				
4334	04/12/2019	09/12/2019		Y				
4335	04/12/2019	11/10/2020		Y				
4420	02/01/2020	25/01/2020		Y				
4423	04/01/2020	20/02/2020		Y				
4437	10/01/2020	24/01/2020		Y				
4502	04/02/2020	22/10/2020		Y				
4512	07/02/2020	09/02/2020		Y				
4538	15/02/2020	04/03/2020		Y				
4554	20/02/2020	04/03/2020		Y				
4592	06/03/2020	21/03/2020		Y				
4599	10/03/2020	22/10/2020		Y				
4631	27/03/2020	06/04/2020		Y				
4639	05/04/2020	05/05/2020		Y				
4642	06/04/2020	22/10/2020		Y				
4697	26/04/2020	08/05/2021		Y				
4705	27/04/2020	22/10/2020		Y				
4737	08/05/2020	06/10/2020		Y				

Table III. Bug-fix issues and Class role stereotype over selected releases in K9-mail

release	IT	CD	CT	ST	IH	SP	role change/release	bug issues/relea
4.322	1	4	0	0	1	1	7	1
5.002	1	0	0	0	0	0	1	1
5.007	1	1	1	1	0	0	4	11
5.008	1	0	0	0	0	0	1	2
5.009	0	0	0	0	0	0	0	1
5.108	0	0	0	0	1	2	3	7
5.109	0	1	0	1	0	0	2	9
5.110	0	0	0	0	0	0	0	2
5.111	0	1	1	2	1	1	6	9
5.112	0	0	0	0	0	0	0	3
5.114	0	0	0	0	0	0	0	14
5.115	0	0	0	0	0	0	0	1
5.201	1	0	0	0	1	0	2	4
5.202	0	0	0	0	0	0	0	3
5.203	0	0	0	0	0	0	0	8
5.204	0	1	0	1	1	3	7	13
5.205	0	0	0	0	0	0	0	2
5.206	0	0	0	0	0	0	0	2
5.207	0	0	0	0	0	0	0	3
5.300	1	2	2	0	0	1	6	17
5.301	0	1	0	0	0	0	1	12
5.500	5	1	2	0	1	4	13	22
5.501	0	0	0	0	0	1	1	7
5.502	0	0	0	0	0	0	0	1
5.503	0	0	0	0	0	0	0	1
5.600	1	0	0	1	0	1	3	2
5.700	4	1	1	4	1	1	12	25
5.701	0	0	0	0	0	0	0	2
5.702	0	0	0	1	0	1	2	7
5.703	0	0	0	0	0	1	1	7
5.709	2	0	3	0	0	3	8	18

Table IV- The “total count” for both bug-fix and none bug-fix issues over selected releases in k9-mail

Release/period	bug-fix issues/commits	none bug-fix issues	total counts
5.002	1	0	1
5.106	0	1	1
5.006	0	1	1
5.107	0	1	1
5.007	11	6	17
5.008	2	13	15
5.108	7	5	12
5.009	1	1	2
5.010	1	1	2
5.109	9	5	14
5.110	2	2	4
5.111	9	2	11
5.112	3	2	5
5.113	1	1	2
5.114	14	1	15
5.115	1	2	3
5.201	4	1	5
5.202	3	3	6
5.203	8	1	9
5.204	13	12	25
5.206	2	3	5
5.207	3	0	3
5.300	17	4	21
5.301	12	1	13
5.302	0	3	3
5.303	0	1	1
5.500	22	2	24
5.503	1	1	2
5.600	2	2	4
5.700	25	58	83
5.701	2	1	3
5.702	7	1	8

Table IV- The “total count” for both bug-fix and none bug-fix issues over selected releases in k9-mail

5.703	7	2	9
5.705	0	2	2
5.706	0	2	2
5.707	0	2	2
5.709	18	3	21