# Investigating Information about Software Requirements in Projects That Use Continuous Integration or Not: An Exploratory Study

Rafael Nascimento[a], Luana Souza[b], Pablo Targino[c], Gustavo Sizílio[d], Uirá Kulesza[e] and Márcia Lucena[f]
*Department of Informatics and Applied Mathematics, Federal University of Rio Grande do Norte, Natal, Brazil*

Keywords: Continuous Integration, Github, Project Open Source, Requirements Engineering.

Abstract: Continuous Integration (CI) is a development practice that involves the automation of compilation and testing procedures, increasing the frequency of code integration and the delivery of new features and providing improvements in software quality. Open Source Software (OSS) projects are increasingly associated with the use of CI practices. However, the literature has not yet explored how and if this practice can influence the presence and the types of artifacts and information related to requirements. Thus, this study aimed to investigate the presence, types of artifacts, and information related to requirements found in projects on GitHub, in particular projects that use CI. An exploratory methodology was used to identify and classify the requirements artifacts where the result shows that projects that adopt the CI have, in general, a more amount of requirements artifacts, mainly in artifacts of the GitHub platform such as issues, pull requests, and labels.

## 1 INTRODUCTION

CI is a development practice for automation and frequent code integration (Hilton et al., 2016), where compiling and testing procedures are automated, leading to a more frequent delivery of new features and products (Shahin et al., 2017). The benefits of CI in software development are code errors identified and corrected earlier, thus improving software quality (Zhao et al., 2017). Over the years, Open Source Software (OSS) projects have had greater adherence to this practice (Hilton et al., 2016). But, OSS developers perform requirements engineering activities informally (Kuriakose and Parson, 2015), using artifacts such as issue tracker systems, forums, and blogs to perform communication about requirements (Salo, 2015; Xiao et al, 2018).

When it comes to the quality of the final product with a lower incidence of errors, the impact of CI in software development has already been investigated by several authors (Bernardo and Kulesza, 2018; Hilton et al, 2017; Labuschagne et al, 2017; Zhao et al, 2017). However, the literature still needs to investigate whether the use of CI in GitHub projects contributes to developers storing artifacts and information related to requirements. Considering that projects that adopt the CI practice deliver new features more often (Shahin et al., 2017), it is expected that GitHub projects will have information related to requirements and be accessible in the repositories. Therefore, this work investigates the presence of information and artifacts related to software requirements. It is worth mentioning that it is not this work's goal to analyze the requirements artifacts' quality, but only to identify them and understand your relationships with projects.

In this study, a dataset composed of 164 projects found in (Bernardo and Kulesza, 2017; Nery and Kulesza, 2018) was used. It was divided into two groups: 82 projects that use CI and 82 projects that do not use it (NoCI). The purpose of this selection into two groups is to check if there are similarities and

[a] https://orcid.org/0000-0001-8620-0983
[b] https://orcid.org/0000-0002-9594-8616
[c] https://orcid.org/0000-0002-4646-0526
[d] https://orcid.org/0000-0003-0349-7588
[e] https://orcid.org/0000-0002-5467-6458
[f] https://orcid.org/0000-0002-9394-6641



Nascimento, R., Souza, L., Targino, P., Sizílio, G., Kulesza, U. and Lucena, M.
Investigating Information about Software Requirements in Projects That Use Continuous Integration or Not: An Exploratory Study.
DOI: 10.5220/0010447903030312
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 2*, pages 303-312
ISBN: 978-989-758-509-8
Copyright © 2021 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

differences in the quantity and types of requirements artifacts. In this sense, several indicators for analysis present in the literature were used, such as issues, labels, pull requests (PR), and UML artifacts. The research questions (RQs) addressed were:

- RQ1: What types of artifacts with information related to requirements prevail in CI and NoCI projects?
- RQ2: What is the volume of information related to requirements found in native artifacts in the Github of CI and NoCI projects?
- RQ3: Is there a difference in the volume of requirements information for CI and NoCI projects?

The research is classified as exploratory since it provides an overview of the subject addressed, bringing together characteristics and new dimensions to be explored (Raupp, 2006). Among the results obtained, it was found that there is a similarity in the types and a difference in the quantity of requirements artifacts found in both groups of projects. This work's main contribution is to attest that, just like it happens in NoCI projects (but in a smaller quantity), CI projects present information related to software requirements in their repositories, in an way informal language. Such artifacts are mostly directed to final users in the form of websites and tutorials, while issues and PRs artifacts used by collaborators to communicate information related to requirements on the GitHub platform.

This work is organized as follows: Section 2 presents the research methodology used. In Section 3, the results obtained by this study are presented and discussed. Section 4 presents the threats to the study's validity and the means to mitigate its effects. The works related to this study are presented in Section 5. Finally, the final considerations and future works are shown.

# 2 RESEARCH METHODOLOGY

## 2.1 Projects Investigated

For the CI projects, Bernardo and Kulesza (2017) took into account the 3,000 most popular GitHub projects that were written in programming languages Java, Python, Ruby, PHP, and JavaScript. That have been filtered to guarantee the quality of the proposed dataset. The first filter consists of the definition of the projects that used CI, which was obtained by separating only the projects that contained a build-in Travis-CI, to ensure that the projects in this group have used the CI practice. After, the authors ensured that the projects had a substantial amount of PRs, were all active, and did not consist of sample or toy projects. The collection process resulted in 87 projects; however, a final step was applied by Nery and Kulesza (2018) where considered only 82 projects in which they found an automated test code.

Concerning NoCI projects, Nery and Kulesza (2018) used a proposal similar to Bernardo and Kulesza (2017). The authors also start from the 3,000 most popular projects on GitHub that were written in programming languages Java, Python, Ruby, PHP, and JavaScript. However, the authors made sure to separate projects that never adopted CI in their life cycle. It is a difficult task to perform with an automatic analysis since the projects may not present CI configuration files and still use some internal server or apply the practice in a way that is not reflected in the published code. Therefore, to projects that were not found CI service configuration files, the authors contacted project contributors via e-mail and other communication channels to ensure that the project never adopted CI. In this dataset, the same filters were used to guarantee the quality, i.e., only active and relevant projects. In the process, the authors provided 82 CI projects and 82 NoCI projects.

So, our study addresses these 164 projects, divided into two groups: 82 projects that use CI through the Travis CI tool and 82 NoCI projects.

## 2.2 Data Collection and Analysis Procedures

In possession of the 164 projects selected, a manual and an automatic search was carried out looking for artifacts that could be related to information about requirements (Salo, 2015; Robles et al, 2017; Portugal and Prado Leite, 2016; Portugal et al, 2016; Ho-Quang et al, 2017), such as:

- Native GitHub artifacts (readme files; Wiki page - used to describe the project's information; issues - used by users to submit project tasks; PRs - used to solve issues; and labels - used to classify issues and PRs);
- Files that are not native to GitHub (UML - use-case, activities, sequence, states, classes, and domain diagrams; feature model, goals model, entity-relationship diagram [ERD], software requirements specification (SRS), personas, mind maps, user stories, websites, tutorials, and functional and/or acceptance test scripts).

The manual search process was used to identify artifacts that were not native to the GitHub platform

and the native artifact "labels." After identifying the labels artifacts related to requirements, they were used in an automated search to identify issues and PRs also related to requirements. Both research modes were carried out with the English language, in the period of October 2019.

### 2.2.1 Search in Native Artifacts

In this work, the terms employed in (Glinz, 2011; IEEE Standards Coordinating Committee, 1990) were used. The terms "feature(s)," "requirement(s)," "functionality(ies)," and "functional" were searched for in readme files, followed by information related to features, such as a list of features. For the labels artifacts, were used the same terms plus the standard GitHub terms: enhancement and improvement. Also, terms that refer directly/indirectly to non-functional requirements were also used, based in the literature (Glinz, 2017; IEEE Standards Coordinating Committee, 1990): security, performance, UX, and UI. It is important to note that several different types of terms can describe non-functional requirements, and that is why we do not limit the number of types of terms to be found in the projects, so we can have a broad view of the types of terms used. Finally, labels with terms for functional and non-functional requirements were used to filter and quantify issues and PRs related to the requirements.

### 2.2.2 Search in Non-native Artifacts

The same terms used to search for functional and non-functional requirements' native artifacts were also used when searching for non-native artifacts, both in the name of the files and in their content. Regarding to UML artifacts, entity-relationship diagram, feature model, goals model, mind maps, user stories and personas, a search was made for files with “.uml”, “.xml”, “.xmi”, ".jpg", ".jpeg", ".png", ".bmp", ".gif" and ".svg" extensions. For SRS files, tutorials and websites, a search was made for files with ".doc(x)", ".pdf", ".odt", ".ppt(x)" and ".html" extensions; and it was checked if they had descriptions of features.

In the manual search for functional and non-functional test artifacts, folders and scripts of codes named with the terms "functional(ity)" and "acceptance" were considered, based on the following research papers (Glinz, 2011; IEEE Standards Coordinating Committee, 1990). Since test artifacts can also reveal relevant information related to requirements, they have been analyzed for this purpose. The manual search for websites verified whether the projects had a valid link to their websites. For the artifacts with readme files, Wiki page, website, and tutorials, only the most recent versions were considered. For the SRS, UML, ERD, feature model, goals model, personas, mind maps, user stories, issues, labels, and PRs artifacts, several were counted and analyzed per project. Regarding the functional and non-functional test artifacts, it was considered whether the projects have test artifacts or not to understand if they verify and validate the requirements.

After its acquisition, the data were classified to enable their interpretation and further analysis. The following data were classified: the number of versions and collaborators in each project; the number of non-native artifacts identified and of your projects; the number of native artifacts identified and of your projects. Based on the number of native artifacts, analyses were performed to understand how many are relevant to information related to requirements.

# 3 RESULTS AND DISCUSSIONS

## 3.1 RQ1: What Types of Artifacts with Information Related to Requirements Prevail in CI and NoCI Projects?

To answer the first research question, we counted which projects have artifacts describing requirements considering the different types of existing artifacts (readme file, UML, Wiki page, websites etc.). Figure 1 presents an overview of the results. Where, we can see that for the artifacts tutorials, websites, test scripts and UML, there was a more significant number of CI projects presenting system requirements compared to NoCI projects. It is possible to note that the most common way to present requirements in both projects that do and do not use CI were tutorials and websites. The tutorial artifact is the most common form of requirements documentation, being found in 49 CI’s and 41 NoCI’s projects. The website artifact, in turn, was used in 46 CI’s and 39 NoCI’s projects.

In 36 of the 82 NoCI projects analyzed, the readme artifacts describe the system's requirements, as opposed to only 21 CI projects. Regarding the Wiki page artifact, it was found that only 14 NoCI projects and 8 CI projects. Regarding the UML artifacts, they were discovered in only four CI projects and were: class, activity, and sequence diagrams. Concerning the test script artifacts only 9 NoCI’s and 25 CI projects. The test artifacts found are scripts to execute functional, non-functional, and acceptance test specifications for validation within these projects.

Finally, only one test plan artifact was found in a NoCI project.

### 3.1.1 Most Used Artifacts

As shown in Figure 1, the number of projects that describe requirements through UML artifacts, test plans, and Wiki pages is minimal compared to the total number of NoCI and CI projects. Most NoCI and CI projects use tutorials and websites to describe the system's functionalities. It may indicate that developers prefer to conduct the documentation or requirements information in a way that is more oriented to the system's end-user and not to the other stakeholders involved in the development process.

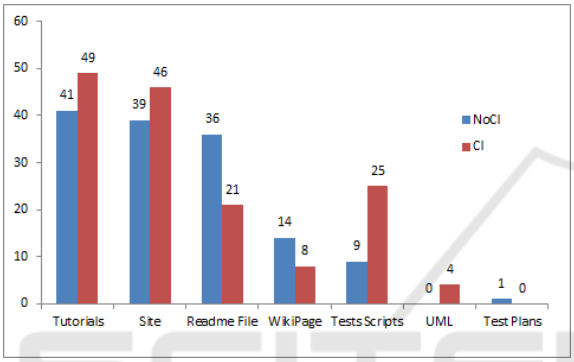


Figure 1: Number of projects by type of artifact.

Regarding the readme files, about half of the NoCI projects use this artifact to describe requirements. However, only 21 (25.60% of total) CI projects use it, which is considered a small number compared to the total number of CI projects.

In particular, the number of CI projects with test scripts related to requirements is also small compared to the total number of CI projects, corresponding to about 25 (30.49% of total). No test plan artifacts were found int CI projects, which are generally used to declare test setup and procedures (containing information related to requirements). Only one example was found in a single NoCI project.

### 3.1.2 Labels Related to Requirements

Another type of artifact within GitHub repositories that is widely used to describe requirements are the issues or PRs labels. Figure 2 shows the number of projects that have labels that are related to functional and/or non-functional requirements. We found about 100 label names used for requirements, but this chart only shows the label names found in at least two projects per group. As can be seen, the most used label in the projects was "enhancement," which was used by 47 NoCI projects and 44 CI projects. The "feature" label was the second most used, found in 18 NoCI projects and 23 CI projects. Then came the "feature request" label, used by 15 NoCI projects and 22 CI projects. We also found labels related to non-functional requirements, such as the "performance" label used in 4 NoCI's and 25 CI's projects, and the "security" label (4 NoCI's and 16 CI's projects).

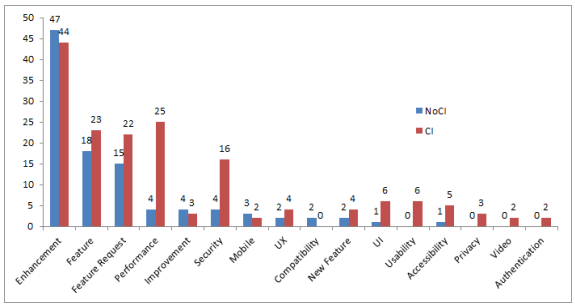


Figure 2: Number of projects by type requirement labels.

In general, only about 16 labels are present in at least more than one project. The other labels are conditioned to only one project, whether it is a NoCI or CI project. It is important to note that the labels were used to filter issues and PRs that may contain information related to requirements. The fact that an issue or pull request uses a label whose name is related to requirements does not guarantee that it has information about requirements. In general, it indicates that the developers' code commits are associated with an issue or pull request that represents that specific requirement. Besides, issues and PRs can be tagged with more than one label, including more than one label related to requirements.

## 3.2 RQ2: What Is the Volume of Information Related to Requirements Found in Native Artifacts of Github in CI and NoCI Projects?

To answer the second research question, we present data on issues, PRs, and labels for CI and NoCI projects, comparing which ones are related to requirements and which are not. Table 3 shows the statistics for NoCI projects. It can be seen that there is a significant proportional difference between issues, PRs, and labels containing explicit requirements information (letter R highlighted in the columns) and those that do not, which can be verified in all three types of data, that is, issues, PRs, and labels. Also, there are cases of projects without any type of data related to requirements.

Table 1: Issues, PRs, and labels for CI projects.

| | Issues | Issues R | Pull | Pull R | Labels | Labels R |
|---|---|---|---|---|---|---|
| Min | 6 | 0 | 0 | 0 | 6 | 0 |
| Max | 31578 | 12881 | 35883 | 8065 | 416 | 100 |
| Avg | 4684,86 | 619,634 | 4874,122 | 342,768 | 50,659 | 4,061 |
| Median | 2750 | 204 | 2750 | 17 | 30,500 | 2 |
| Sum | 384179 | 50810 | 399678 | 28107 | 4154 | 333 |
| Std. Deviation | 5888,749 | 1617,117 | 6956,890 | 1195,132 | 64,663 | 11,130 |
| 1º Quartile | 1369,750 | 65 | 1127,750 | 2,250 | 16,250 | 1 |
| 2º Quartile | 2750 | 204 | 2750 | 17 | 30,500 | 2 |
| 3º Quartile | 5339,250 | 593,500 | 4950,500 | 188,500 | 55,500 | 3 |

Table 2 presents data on issues, PRs, and labels for CI projects, both from a global perspective and related to requirements. It can be seen, that, there is a significant difference between the data, which can be verified in all three types of data found, that is, issues, PRs, and labels. Besides, there are examples of projects without any kind of data related to requirements.

Table 2: Issues, PRs, and labels for NoCI projects.

| | Issues | Issues R | Pull | Pull R | Labels | Labels R |
|---|---|---|---|---|---|---|
| Min | 7 | 0 | 7 | 0 | 0 | 0 |
| Max | 10596 | 1023 | 9650 | 1023 | 77 | 12 |
| Avg | 961,49 | 84,329 | 414,732 | 24,463 | 11,610 | 1,512 |
| Median | 326 | 16,500 | 121,500 | 0 | 7,500 | 1 |
| Sum | 78847 | 6915 | 34008 | 2006 | 952 | 124 |
| Std. Deviation | 1924,684 | 195,396 | 1143,486 | 121,639 | 13,209 | 1,701 |
| 1º Quartile | 134 | 1,250 | 64,250 | 0 | 6 | 1 |
| 2º Quartile | 326 | 16,500 | 121,500 | 0 | 7,500 | 1 |
| 3º Quartile | 797 | 56,750 | 322,500 | 1,750 | 12 | 2 |

### 3.2.1 Requirements Artifacts in NoCI Projects

In Table 3, it is noted that there are projects that do not have requirements issues and those that have a maximum number of 1,023 issues. Note that up to the third quartile of the NoCI projects, there are about 56 requirements issues. Only 8.77% of issues on the sum of issues for all projetcs are dedicated to requirements (Figure 3a and Table 3). This information may indicate that the flow of communication of information about requirements between the collaborators is small or that the system is already at a maturity level where there are not many changes in requirements, but only priority in the communication regarding the system's maintenance.

To PRs related to requirements, the minimum and the maximum number are the same as the number of issues related to requirements. Only from the third quartile do projects with at least 1.75 requirements PRs arise. Also, the percentage of total PRs is around 0.5% (Figure 3b and Table 3).

There is a proportion of 30% of requirements PRs for the total number of issues. Which can indicate that

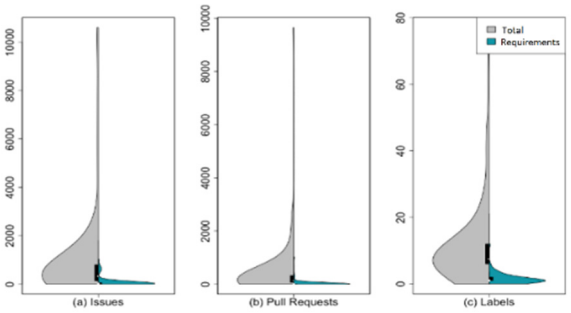


Figure 3: Issues, PRs, and requirements labels in NoCI projects.

about 70% of the PRs submitted in the projects are not about requirements and that only about 25% of the projects have the PRs submitted to solve requirements issues (Figure 3a and 3b).

Regarding the number of requirements labels (Figure 3c and Table 3), the number remains the same for the minimum, while the maximum number of requirements labels in the projects is 12. However, half of the projects only use one label, and up to the third quartile uses only two labels. It allows us to conclude that most projects do not use a wide variety of requirements label. This information corresponds with Figure 2, which illustrates that most NoCI projects use labels such as "enhancement," "feature," and "feature request" and that, in general, they are not specifying the types of requirements. Besides, only 13.02% of the labels in the projects refer to the requirements. This statement is made based only on the name of the labels. However, this method is described as a threat to validation if the label is misused or used generically.

### 3.2.2 Requirements Artifacts in CI Projects

In Table 2, there are projects do not have requirements issues, while some have a maximum number of 12,881 requirements issues. Also, the median number (204) and the third quartile (593.5) are low compared to the projects' maximum number of issues. It means that about 25% of the projects have a significant number of requirements issues. Besides, only has a percentage of 13.22% of the requirements issues in CI projects (Figure 4a and Table 2).

Regarding PRs, there are projects do not have requirements PRs, and the maximum number reaches 8,065. There is a proportional rate of 55.32% of PRs out of the total number of requirements issues (Figure 4a and 4b). That is, about half of the requirements issues have PRs submitted. However, despite the median number being 17 PRs per project, and the third quartile having about 188,500 PRs. Only about

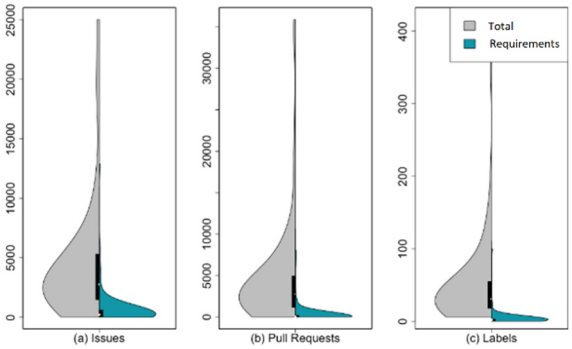


Figure 4: Issues, PRs, and labels related to requirements in CI projects.

25% of the projects have a higher concentration of PRs. In general, only 7.03% of all CI projects' PRs are related to requirements (Figure 4b and Table 2).

Regarding the labels, the situation is similar in NoCI projects. There are projects do not have requirements labels, while there are projects with around 100 requirements labels. However, up to the third quartile of the total number of projects, only three labels are applied to requirements (only 25% of the projects use more than three labels to classify requirements) (Figure 5c and Table 2).

In Figure 2, can be seen that there is a variety of label names used by the projects, with emphasis on names that relate to non-functional requirements such as "performance" and "security." Still, compared to the total of labels for all projects, only 8.01% of the labels in projects are used to classify requirements.

## 3.3 RQ3: Is There a Difference in the Volume of Requirements Information for CI and NoCI Projects?

### 3.3.1 Number of Collaborators and Releases

Table 3 presents information about releases and contributors to the NoCI and CI projects. The maximum number of contributors is 327 for NoCI and 4,520 for CI projects. The minimum number of contributors is 2 for NoCI projects and 1 for CI projects. The median number of contributors is 51 for NoCI projects and 295 for CI projects. As can be seen, the maximum number of releases is 784 for NoCI projetcs and 2,284 for CI projects. The median number of releases ranges between 33.5 for NoCI projects and 112.5 for CI projects.

As noted in Table 3, the number of collaborators and releases in CI projects is much higher than in NoCI projects. Ståhl and Bosch (2014) argue that

Table 3: Project releases and contributors.

| | Releases NoCI | Releases CI | Contributors NoCI | Contributors CI |
|---|---|---|---|---|
| Min | 0 | 0 | 2 | 1 |
| Max | 784 | 2284 | 327 | 4520 |
| Avg | 91,390 | 175,439 | 68,500 | 460,378 |
| Median | 33,500 | 112,500 | 51 | 295 |
| Sum | 7494 | 14386 | 5617 | 37751 |
| Std. Deviation | 145,820 | 273,701 | 64,315 | 682,542 |
| 1º Quartile | 13 | 55,250 | 27 | 138,500 |
| 2º Quartile | 33,500 | 112,500 | 51 | 295 |
| 3º Quartile | 85,500 | 194,250 | 82,500 | 502 |

projects using CI have more frequent release deliveries due to constant collaboration. It may be an indication that CI projects have attracted more attention from the collaborators due to the automated support for testing and verifying the quality of the code (Duvall et al, 2007; Valinescu et al, 2015). This justifies CI projects to have a greater amount of information about requirements to use in the testing and verifying procedures.

Table 4 shows the number of issues, labels e PRs related to requirements for NoCI and CI projects. The minimum number of labels is zero for NoCI and CI projects. The maximum number of labels is 12 for NoCI projects and 100 for CI projects. The minimum number of issues is zero for NoCI and CI projects. The medians have a value of 1 for NoCI projects and 2 for CI projects. The maximum number of issues is 1,023 for NoCI and 12,881 for CI projects. The medians are 16.5 for NoCI projects and 204 for CI projects. The minimum number of PRs is zero for NoCI and CI projects. The maximum number of PRs is 1,023 for NoCI and 8,065 for CI projects. The medians are zero for NoCI and 17 for CI projects.

Table 4: Comparison between NoCI and CI projects.

| | Issues NoCI | Issues CI | Pull NoCI | Pull CI | Labels NoCI | Labels CI |
|---|---|---|---|---|---|---|
| Min | 0 | 0 | 0 | 0 | 0 | 0 |
| Max | 1023 | 12881 | 1023 | 8065 | 12 | 100 |
| Avg | 84,329 | 619,634 | 24,463 | 342,768 | 1,512 | 4,061 |
| Median | 16,500 | 204 | 0 | 17 | 1 | 2 |
| Sum | 6915 | 50810 | 2006 | 28107 | 124 | 333 |
| Std. Deviation | 195,396 | 1617,117 | 121,639 | 1195,132 | 1,701 | 11,130 |
| 1º Quartile | 1,250 | 65 | 0 | 2,250 | 1 | 1 |
| 2º Quartile | 16,500 | 204 | 0 | 17 | 1 | 2 |
| 3º Quartile | 56,750 | 593,500 | 1,750 | 188,500 | 2 | 3 |

### 3.3.2 Relationship between NoCI and CI Projects

In general, it can be observed that in both groups, there are projects that do not have issues and/or PRs related to requirements. However, the maximum number of issues and PRs in CI projects are more significant than in NoCI projects - a difference of

11,858 (92.06%) for issues and 7,042 (87.32%) for PRs between CI and NoCI projects. The percentage in relation to the total number of issues and PRs are also significant - about 43,895 (86.39%) issues and 26,101 (92.86%) for PRs between CI and NoCI projects (Figures 7a and 7b). This information shows that the number of issues and PRs with requirements information in CI projects is much higher than in NoCI projects.

There are projects in both groups that do not have requirements labels. There are projects in the NoCI group with a maximum number of 12 and CI projects with a maximum of 100 (a difference of 88 labels, about 80%). In a comparison made with the total number of labels in all projects, the difference is 209 labels (a percentage of 62.76%). However, the median of NoCI projects is equal to one label, as opposed to two labels for CI projects. From up to third quartile of the projects, NoCI projects use up to two labels to classify requirements, while CI projects use up to three labels (Figure 7c).

Tables 2 and 3 present data that indicate the existence of a difference in the number of issues, PRs, and labels related to requirements between NoCI and CI projects. To compare the two samples and better understand how our metrics are associated with each of the approaches (i.e., CI and NoCI), our study applied statistical tests to attest to the difference between the data presented. First, we calculated the percentage corresponding to the requirements for each of the adopted metrics. For example, if a project has 200 PRs, of which 50 were related to requirements, this project would have a proportion of 0.25 (or 25%).

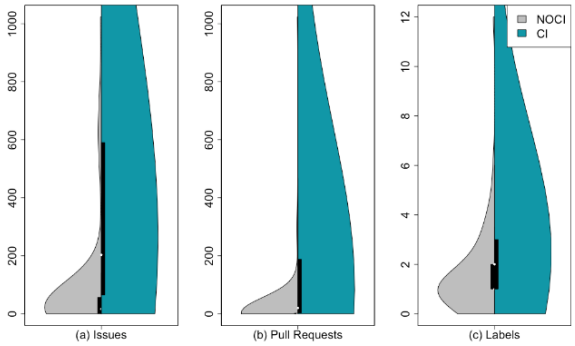


Figure 7: Requirements data in NoCI and CI projects.

Following this example, we calculated the proportions for issues, PRs, and labels. The higher the proportion, the more related to requirements the variable was. Then, we compared the projects' factors from the CI sample with those from the NoCI sample for each of the variables. For this purpose, two statistical tests were applied: (i) Mann-Whitney-Wilcoxon tests (MWW or Wilcoxon rank-sum test) (Wilks, 2011), a non-parametric method used to compare samples and certify that the values are statistically different, that is, a p-value $< 0.05$ indicates that the samples came from different populations; and (ii) Cliff's delta, a metric computed to measure the magnitude of such difference between distributions (Macbeth et al, 2011). To interpret Cliff's delta, we used the thresholds indicated by Romano et al. (2006), i.e., delta $< 0.147$ (negligible), delta $< 0.33$ (small), delta $< 0.474$ (medium), and delta $> = 0.474$ (large).

The results of our statistical tests show that the PRs of the CI sample are statistically more associated with requirements, with the Wilcoxon p-value = 1.633e-06 and Cliff's delta 0.4297999 (medium). For the issues factor, the results also show that CI has more issues related to requirements, with the Wilcoxon p-value = 0.005635 and Cliff's delta of 0.2522383 (small). Finally, concerning the labels factor, we observe the opposite. With the Wilcoxon p-value = 0.000342 and Cliff's delta -0.335443 (medium), we observed that CI projects are associated with fewer labels related to requirements.

### 3.3.3 Discussion of Results

Through the interpretation of the data collected and analyzed, we can assume that both groups of CI and NoCI projects use informal artifacts to describe requirements, which, in general, are tutorials, readme files, websites, issues, PRs, and labels, as noted in recent works (Salo, 2015; Portugal and Prado Leite, 2016; Portugal et al, 2016). Mainly, issues and PRs Where the developers use in their frequent communication as collaborators in a project.

Regarding the labels, about 55% of the CI and NoCI projects prefer using labels with generic names to present requirements information. Besides, about 35% of CI projects use words for non-functional requirements such as "performance" and "security."

It was also observed and validated through statistical tests that projects that adopt the practice of CI tend to have a higher proportion of issues and PRs related to requirements than projects that do not adopt the practice of CI.

This may indicate that CI projects tend to have more information about requirements, due to the need for better code quality and the need for more frequent deliveries, requiring more frequent communication. In addition, these projects tend to attract more contributors and end up using mainly informal artifacts to communicate information about

requirements. And where a few types of non-functional requirements are communicated with specific and somewhat diverse keywords.

# 4 THREATS TO VALIDITY

## 4.1 Construction Validation

The construction validation can be threatened by the selection mechanisms used in works (Bernardo and Kulesza, 2017; Nery and Kulesza, 2018) to select the projects and the types of artifacts used in this study. However, the authors state that the projects collected were carefully selected as CI and NoCI projects, and have been used in previous work experiments.

Regarding the mechanisms for selecting the artifacts, we can consider threats both the types of extensions of artifacts and how they were acquired. The types of extensions used are limited and, therefore, artifacts extensions of tool like Astah, ArgoUML, Modelio, among others, were suppressed. UML artifacts and other types of artifacts with other types of extensions for documents and/or images may also have been suppressed. However, native and non-native types of artifacts used in other research were considered objects of studies with information related to requirements (Salo, 2015; Ho-Quang, 2017).

Regarding the search for native artifacts, in the manual search for readme files, Wiki pages, and labels artifacts, many may have been discarded for not meeting the terminology used or the name given to the artifacts. The manual search may also generate an incorrect count of labels, which may have caused, in its turn, an inaccurate count in the number of issues and PRs for each project. Besides, it is essential to note that issues and PRs related to other types of labels whose names do not refer to requirements but that could contain information about requirements even though they used other labels (issues and PRs that were wrongly tagged, for example) were ignored. On the other hand, among the data obtained on issues and PRs related to requirements, there may be data that do not include information on requirements because they were tagged with the wrong labels. Unfortunately, this threat cannot be reduced.

Regarding test scripts, it may also have discarded test scripts that were not organized in folders named "test(s)" and/or "spec(s)" and/or that used other keywords for the name of the scripts. To mitigate this threat, extensions of different tools were included in the searches carried out, and different synonyms related to software tests were used.

## 4.2 External Validation

The threats to external validation are related to the generalization of the results of the study. Our study analyzed about 164 popular GitHub projects. They were collected to represent samples of projects that use or do not use the CI practice. Despite the extent and size of the collected dataset, it is not possible to generalize the results beyond the defined context, which requires future analyzes and studies for this purpose. Finally, the types of artifacts used in our selection may not generally represent all types of requirements artifacts used by developers in GitHub projects. This threat has been mitigated through a detailed manual and automatic analysis of the artifacts that make up the projects and using results reported by other studies (Salo, 2014; Salo, 2015) that investigate requirements in OSS projects.

## 4.3 Internal Validation

Threats to the construction validity may have had consequences for the data's internal validation since relevant NoCI and/or CI projects may have been discarded. This threat may not have been avoided since the projects used to come from other researches and the application of selection mechanisms. Also, the keywords used in selecting types of artifacts may have caused the suppression of other kinds of artifacts that could lead to a different path in the answers to the research questions. However, this threat has been minimized by use of keywords consolidated in the literature. Both the language used in the search and all selected projects are in the English language.

# 5 RELATED WORKS

Robles et al. (2017) investigated whether UML artifacts are used in GitHub projects. They analyzed about 12 million projects and found 93,000 UML artifacts in about 24,000 projects. Our work focused on the search for different types of requirements artifacts, not just UML artifacts. Also, our study focused on the context of comparing CI and NoCI GitHub projects. Thus, the similarity between the works is only their purpose of finding UML artifacts in GitHub projects, with our study focusing on requirements artifacts.

Ho-Quang et al. (2017) investigate the practices and perceptions of using UML in OSS projects to understand its motivations and benefits. A survey was carried out with 485 respondents, only for projects that use UML. As a result, it was noted that

collaboration is the most important factor since it benefits new project collaborators to understand the requirements, design, and implementation of the system. Using UML to improve communication and planning effort in implementation. In our case, developers of CI and NoCI OSS projects use native artifacts GibHub to communicate requirements.

Salo et al. (2014, 2015) investigate guidelines for managing agile requirements on GitHub projects. They propose good practices for using the GitHub platform functionalities to contain information related to requirements such as issues, PRs, labels, and milestones. They conclude that with little effort, integrating the proposed guidelines with GitHub is feasible for managing requirements in agile environments. The study conducted and presented in this article found several pieces of evidence of the use of the guidelines presented by Salo et al. (2014, 2015) on GitHub projects, such as creating, updating, and maintaining issues to represent different types of requirements combined with Wiki documentation.

## 6 CONCLUSIONS

This work presented an exploratory study on how information related to requirements is being stored in OSS GitHub projects. It also explored the similarities and differences between CI and NoCI projects in their development, using previous research datasets. Mechanisms for selecting and searching for artifacts related to requirements have been developed based on other research in the literature. Manual and automated searches were performed to retrieve, analyze, and interpret this data.

In general, the study concluded that GitHub projects that have a more number of collaborators will consequently have a greater amount of information related to requirements, mainly in informal artifacts such as issues and PRs. It happens because with more collaborators and releases, consequently, they will have a greater flow of information, a behaviour that can be observed in CI projects.

The study noted that GitHub projects, regardless of the group, use informal artifacts such as tutorials, websites, readme files, and, mainly, issues and pull request artifacts that serve as communication and collaboration mechanisms between developers on the platform to describe information related to requirements. However, the results indicate that CI projects have more information related to requirements, mainly stored in issues and PRs, than projects that do not use CI.

Regarding the classification of information related to requirements, in both groups, the use of labels with keywords such as "enhancement," "feature," and/or "feature request" is predominant. However, only 25% of CI projects use a large number of labels to classify information about the requirements.

The following future works of this research are being planned: (1) conducting new analyzes that allows evaluating a greater proportion of information related to requirements according to the characteristics of the projects, such as languages used, types of software, age of the projects, most recognized projects; (2) conducting qualitative analyzes with participants of the investigated projects through surveys, seeking to confirm and expand the results related to requirements specification in OSS projects.

## ACKNOWLEDGEMENTS

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — Brasil (CAPES) — Finance Code 001.

## REFERENCES

Bernardo, J. H., da Costa, D. A., & Kulesza, U. (2018, May). Studying the impact of adopting continuous integration on the delivery time of PRs. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)* (pp. 131-141). IEEE.

Nery, G. S., da Costa, D. A., & Kulesza, U. (2019). An Empirical Study of the Relationship between Continuous Integration and Test Code Evolution. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 426-436). IEEE.

Salo, R. (2014). *A guideline for requirements management in GitHub with lean approach* (Master's thesis).

Salo, R., Poranen, T., & Zhang, Z. (2015, October). Requirements management in GitHub with a lean approach. In *SPLST* (pp. 164-178).

Robles, G., Ho-Quang, T., Hebig, R., Chaudron, M. R., & Fernandez, M. A. (2017, May). An extensive dataset of UML models in GitHub. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)* (pp. 519-522). IEEE.

Kuriakose, J., & Parsons, J. (2015, August). How do open source software (OSS) developers practice and perceive requirements engineering? An empirical study. In *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)* (pp. 49-56). IEEE.

Vlas, R., Robinson, W., & Vlas, C. (2017). Evolutionary software requirements factors and their effect on open source project attractiveness.

Portugal, R. L. Q., & do Prado Leite, J. C. S. (2016, September). Extracting requirements patterns from software repositories. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)* (pp. 304-307). IEEE.

Portugal, R. L. Q., Roque, H., and do Prado Leite, J. C. S., 2016. A Corpus Builder: Retrieving Raw Data from GitHub for Knowledge Reuse In Requirements Elicitation. In *3rd. Annual International Symposium on Information Management and Big Data*, 48.

Ho-Quang, T., Hebig, R., Robles, G., Chaudron, M. R., and Fernandez, M. A., 2017. Practices and perceptions of UML use in open source projects. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP),* 203-212. IEEE.

Ferrari, A., Spagnolo, G. O., and Gnesi, S., 2017. PURE: A dataset of public requirements documents. In *2017 IEEE 25th International Requirements Engineering Conference (RE),* 502-505, IEEE.

Kuriakose, J., 2017. *Understanding and improving requirements discovery in open source software development: an initial exploration.* Doctoral dissertation, Memorial University of New Foundland.

Saeed, S., Fatima, U., and Iqbal, F., 2018. A review of Requirement Elicitation techniques in OSSD. *International Journal of Computer Science and Network Security*, 86-92.

Iyer, D. G., 2018. *Propagation of requirements engineering knowledge in open source development: causes and effects–A social network perspective,* Doctoral dissertation, Case Western Reserve University.

Glinz, M., 2011. A glossary of requirements engineering terminology. *Standard Glossary of the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version*, *1*.

IEEE Standards Coordinating Committee, 1990. IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). Los Alamitos. *CA: IEEE Computer Society*, *169*.

Ståhl, D., and Bosch, J., 2014. Modeling continuous integration practice differences in industry software development. In *Journal of Systems and Software*, 87, 48-59.

Duvall, P. M., Matyas, S., and Glover, A., 2007. *Continuous integration: improving software quality and reducing risk*. Pearson Education.

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., and Filkov, V., 2015. Quality and productivity outcomes relating to continuous integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering,* 805-816. ACM.

Stolberg, S. (2009, August). Enabling agile testing through continuous integration. In *2009 agile conference* (pp. 369-374). IEEE.

Hilton, M., Nelson, N., Tunnell, T., Marinov, D., & Dig, D. (2017, August). Trade-offs in continuous integration: assurance, security, and flexibility. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (pp. 197-207).

Labuschagne, A., Inozemtseva, L., & Holmes, R. (2017, August). Measuring the cost of regression testing in practice: a study of Java projects using continuous integration. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (pp. 821-830).

Zhao, Y., Serebrenik, A., Zhou, Y., Filkov, V., & Vasilescu, B. (2017, October). The impact of continuous integration on other software development practices: a large-scale empirical study. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 60-71). IEEE.

Raupp, F. M., & Beuren, I. M. (2006). Metodologia da Pesquisa Aplicável às Ciências. *Como elaborar trabalhos monográficos em contabilidade: teoria e prática. São Paulo: Atlas*, 76-97.

Hilton, M., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016, September). Usage, costs, and benefits of continuous integration in open-source projects. In *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 426-437). IEEE.

Shahin, M., Babar, M. A., and Zhu, L., 2017. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909-3943.

Wilks, D. S. (2011). *Statistical methods in the atmospheric sciences* (Vol. 100). Academic press.

Macbeth, G., Razumiejczyk, E., & Ledesma, R. D. (2011). Cliff's Delta Calculator: A non-parametric effect size program for two groups of observations. *Universitas Psychologica*, *10*(2), 545-555.

Romano, J., Kromrey, J. D., Coraggio, J., & Skowronek, J. (2006, February). Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys. In *annual meeting of the Florida Association of Institutional Research* (pp. 1-33).

Xiao, X., Lindberg, A., Hansen, S., and Lyytinen, K. (2018). "Computing" Requirements for Open Source Software: A Distributed Cognitive Approach. *Journal of the Association for Information Systems*, *19*(12), 2.