DO DEVELOPERS DISCUSS QUALITY ATTRIBUTES?

Arman Yousef Zadeh Shooshtari, V00914798

Computer Science Department, University of Victoria, Canada.

armanyousefzade95@gmail.com

Table of Contents

1 Introduction 2
2 Quality Attributes 3
2.1 Functional Suitability 3
2.2 Performance efficiency 4
2.3 Compatibility 4
2.4 Usability 4
2.5 Reliability 5
2.6 Security 5
2.7 Maintainability 6
2.8 Portability 6
2.9 Other 6
3 Data Set 7
4 Methodology 7
4.1 Building the machine learning model 7
4.2 Answering the research questions 9
5 Results 9
5.1 RQ1: To what extent do developers discuss quality attributes in open-source projects? 9
5.2 RQ2: Which quality attributes are discussed in open source projects? 9
5.3 RQ3: To what extent is it possible to automatically classify a discussion as design-related or not given the quality attribute and sub-attribute it discusses. 12
6 Threats to validity 13
7 Related work 13
8 Conclusion and future work 14
9 Acknowledgments 15
10 References 15

1 Introduction

Open software development environments enable developers to submit code to any project. As a result, a lot of people with various expertise can add unique value to a project. At the same time, openness poses the tremendous problem of assessing these contributions to ensure their quality and to maintain technical integrity and quality attributes [1]. When contributions are deemed inappropriate or threaten technical integrity or quality attributes, a negotiation often takes place between contributors and project members [2]. In response to code contributions, a study of the content and form of comments in lengthy discussions would extend our understanding of collaboration dynamics in an open environment. We can inform policies and tools that allow software developers to better manage open projects by better understanding the nature of comments in long discussions about contributions to open software projects [1].

Developers are usually struggling with the balance between writing the code necessary to provide desired behavior and respecting the system's quality attributes when working on a project. This task is made even more difficult because information on quality attributes is seldom explicitly documented, at least in the case of open source systems. Instead, it is scattered through various types of artifacts, including discussions on issues tracker systems and pull requests. As a result, in particular, newcomers are struggling to determine and respect the current project design and quality attributes [3, 4]. If we could determine which quality attributes were being discussed, such information could then be represented and used to ease many software development tasks. For example, if we were able to automatically determine the quality attribute being examined during a pull request discussion, we could invite core project members with adequate expertise to participate. In turn, such invitations could reduce the time and effort required to review a pull request. Also, we can build tools to help developers understand why a piece of code has been designed in a specific way, by extracting and identifying quality attributes from discussions related to previous pull requests.

To understand whether quality attributes are discussed and shared in issues commits' comments, and pulled requests from open source projects, I conducted an empirical study on 5 of GitHub's top popular projects to provide quantitative evidence on how developers conduct discussions on quality attributes. I am trying to answer three research questions:

RQ1: To what extent do developers discuss quality attributes in open-source projects?

RQ2: Which quality attributes are discussed in open source projects?

RQ3: To what extent is it possible to automatically classify a discussion as design-related or not given the quality attribute and sub-attribute it discusses.

To answer these questions, I first manually tagged 1000 discussion with three labels. The first label shows whether the discussion discusses any quality attribute or not, the second one shows which quality attribute was discussed in the discussion and the third one shows which quality sub-attribute discussed in the discussion. After that, I implemented naïve based text classifier and trained and evaluate the classifier by using cross-validation. I utilized a logistic regression model to answer the third research question. In the following sections, all the steps will be described in details.

2 Quality Attributes

The quality model is at the heart of a system of quality assessment. When assessing the properties of a software product, the quality model determines which quality attributes are being considered. The quality of a system is the degree to which the system meets its various stakeholders ' stated and implied needs and therefore provides value. The needs of those stakeholders (functionality, performance, security, maintainability, etc.) are precisely what the quality model represents, which categorizes the quality of the product into attributes and sub-attributes [5].

In this paper, we use the product quality model defined in ISO/IEC 25010[5]. It includes the 8 quality attributes and 31 quality sub-attributes shown in the figure below:

Fig1: Quality attributes and sub-attributes considered by ISO/IEC FCD 25010[5]

Now, I define each of the quality attributes and their corresponding sub-attributes. For each of the quality attributes, you can see a discussion which is related to that quality attribute in the listings. The definitions are based on ISO/IEC 25010[5]. For the labeling the discussions I utilized the following quality attributes and sub-attributes.

2.1 Functional Suitability

This attribute represents the extent to which, when used under specified conditions, a product or system provides functions that satisfy stated and implied needs. The following sub-attributes are included in this attribute:

- **Functional completeness.** The degree to which the set of functions covers all the specified tasks and user objectives.
- **Functional correctness.** The degree to which a product or system with the required degree of accuracy delivers the correct results.
- **Functional appropriateness.** The degree to which the functions facilitate the fulfillment of specific tasks and goals.

Listing 1: Discussion related to functional suitability

```
Just to complete the picture it works well for day level interval just not minute
level and probably hour as well
```

2.2 Performance efficiency

This attribute shows the performance relative to the number of resources used in specified conditions. This attribute consists of the following sub-attributes:

- **Time behavior.** The degree to which a product or system's response and processing times and throughput rates meet requirements when carrying out its functions.
- **Resource Utilization.** The degree to which a product or system's response and processing times and throughput rates meet requirements when carrying out its functions.
- **Capacity.** The degree to which the maximum limits of a product or system parameter meet requirements.

Listing 2: Discussion related to performance efficiency

```
I would like to see benchmarks against the normal MySQL driver as part of this effort
For performance memory usage and CPU usage
```

2.3 Compatibility

The degree to which a product, system or component may exchange information and/or perform its required functions with other products, systems or components while sharing the same hardware or software environment. The following sub-attributes make up this attribute:

- **Co-existence.** The degree to which a product can effectively perform its required functions while sharing a common environment and resources with other products without adverse effects on any other product.
- **Interoperability.** To what extent two or more systems, products or components can exchange information and use the exchanged information.

Listing 3: Discussion related to compatibility

```
erh Sorry I meant that I have only relaxed it for trim aka slice which by my
definition states that it will always be evaluated after any other modifier So it is
backwards compatible
```

2.4 Usability

The degree to which specified users can use a product or system to achieve specified objectives with efficiency, effectiveness, and satisfaction within a specified context of use. The following sub-attribute form parts of this attribute:

- **Appropriateness recognizability.** To what degree users can recognize the suitability of a product or system for their needs.
- **Learnability.** To what extent a product or system may be used by specified users to achieve specified objectives of learning to use the product or system in a specified context of use with effectiveness, efficiency, risk - free and satisfaction.
- **Operability.** The degree to which a product or system has attributes that make it easy to operate and control.

- **User error protection.** To what degree a system protects users from making mistakes.
- **User interface aesthetics.** The degree to which a user interface allows the user to interact pleasingly and satisfyingly.
- **Accessibility.** The degree to which people with the widest range of features and capabilities can use a product or system to achieve a specified purpose in a specified use context.

Listing 4: Discussion related to usability

```
True but I do not want to waste CPU formatting something that might not be used and
then the user can not really reformat easily
```

2.5 Reliability

The degree to which for a specified period of time a system, product or component performs specified functions under specified conditions. This attribute contains the following sub-attributes.

- **Maturity.** The degree to which the reliability needs of a system, product or component under normal operation are met.
- **Availability**. The degree of operation and accessibility of a system, product or component when required for use.
- **Fault tolerance.** The degree to which, despite hardware or software faults, a system, product or component operates as intended.
- **Recoverability.** The degree to which a product or system can recover the data directly affected and restore the desired state of the system in the event of an interruption or failure.

Listing 5: Discussion related to reliability

```
hmm I have issue in my server when server crashes or players net or client game crashes
```

2.6 Security

To what extent a product or system protects information and data in such a way that individuals or other products or systems have the degree of access to data appropriate to their types and authorization levels. This attribute comprises the following sub-attributes:

- **Confidentiality.** To what extent a product or system ensures that data is only accessible to those who are authorized to access it.
- **Integrity.** To what extent a system, product or component prevents unauthorized access to or alteration of computer programs or data.
- **Non-repudiation.** To what extent actions or events can be proven to have occurred so that events or actions cannot be later repudiated.
- **Accountability.** To what extent an entity's actions can be traced to the entity uniquely.
- **Authenticity.** The degree to which a subject or resource's identity can be proven to be the claimed one.

Listing 6: Discussion related to security

```
The events need to be private and should not be exposed They are for internal use only
Users can use awaitActivation methods from Activation
```

2.7 Maintainability

This attribute reflects the degree of efficiency and effectiveness with which a product or system can be modified in order to improve, correct or adapt it to environmental and requirements changes. This attribute consists of sub-attributes as follows:

- **Modularity.** To what extent a system or computer program consists of discrete components so that a change to one component has minimal impact on other components.
- **Reusability.** To what extent an asset can be used in more than one system or in the construction of other assets.
- **Analysability.** The degree of efficiency and effectiveness with which it is possible to evaluate the impact of an intended change to one or more of its parts on a product or system, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.
- **Modifiability.** To what extent a product or system can be modified effectively and efficiently without introducing defects or degrading the existing quality of the product.
- **Testability.** The degree of efficiency and effectiveness with which test criteria for a system, product or component can be established and tests can be performed to determine whether those criteria have been met.

Listing 7: Discussion related to maintainability

```
Note that dump cpp has the same code see SERVER 1833If we change it in export cpp we
should make the same change in dump cpp
```

2.8 Portability

The level of efficiency with which a system, product or component can be transferred from one to another hardware, software or other operating or usage environment. This attribute is made up of the following sub-attributes:

- **Adaptability.** To what extent a product or system can be adapted to different or evolving hardware, software or other operational or usage environments effectively and efficiently.
- **Installability.** The degree of efficiency and effectiveness with which it is possible to successfully install and/or uninstall a product or system in a specified environment.
- **Replaceability.** To what degree a product in the same environment can replace another specified software product for the same purpose.

Listing 8: Discussion related to portability

```
Also I don t think this works with multiple machines or discover
```

2.9 Other

This category of the quality attributes represents two other attributes that are not included in the quality model defined in ISO/IEC 25010, but I consider them in labeling the discussions because they are important in software quality.

- **Scalability.** The system's ability to handle load increases without decreasing performance or the ability to increase the load quickly.
- **Supportability.** The system's ability to provide useful information to identify and resolve issues.

Listing 9: Discussion related to supportability

```
Added a more verbose message to the illegal argument exception in case the date format
cannot be parsed the format is now put into the message
```

3 Data Set

In the data collection phase, I needed 1000 discussions to manually labels them with quality attributes described earlier. I also intended to find an answer for the third research question and find out whether it is possible to automatically classify a discussion as design-related or not design-related given the quality attribute and sub-attribute it discusses. As a result, I labeled the discussions which Brunet et al. [6] had already tagged as design related or not design-related. I labeled them as related to quality attributes or not. I also tagged them with the quality attributes and sub-attributes which are discussed in them, but Brunet et al. [6] had labeled them in a different way that showed whether they were design related or not. I used my labels to answer the first and second research question and utilized Brunet et al. [6]'s labels along with my labels to answer the third research question. Brunet et al. [6] for selection the discussions had discarded 13 projects with fewer than 50 discussions of 90 projects present in the GHTorrent data set [7]. In order to work with a reasonable amount of data, they selected 77 projects with over 50 discussions. This decision was made by them because the more discussions the project has, the more likely it is that the quality attributes and design will be discussed. Since I was interested in discussions about quality attributes, I made the same decision and utilized the same discussions and projects. I have also considered projects and their forks as one project to simplify the analysis.

4 Methodology

4.1 Building the machine learning model

I implemented the multinomial naïve Bayes text classifier in Java. Algorithm 1 shows the pseudo code of implemented multinomial naïve Bayes text classifier. After implementing that to ensure the accuracy of the classifier, I used 10 fold cross-validation on the same data used in Brunet et al. [6]. The accuracy of my classifier was similar to their classifier when trained and tested on the same data. So, I became sure that my implemented multinomial naïve Bayes text classifier performed correctly and it was ready to train and test on my data.

After implementing the text classifier, I conducted the following steps to build my quality attributes classifier model:

Step 1. As I mentioned earlier, for the labeling the discussions, I chose the same discussion which Brunet et al. [6] selected. First, 5 projects were randomly selected from the 77 projects. The projects were: BitCoin, Akka, OpenFrameworks, Mono, and Twitter--Finagle. Then, 200 discussions were randomly chosen from each of the 5 projects, totaling I had 1,000 discussions for labeling.

Step 2. I manually classified 1,000 discussions. I tagged the discussions with three labels. First as related to quality attributes or not, second with the quality attribute they discuss, and third with the quality sub-attribute they discuss. For tagging, I considered the quality attributes and sub-attributes that defined in

section 2. The discussions were also tagged as design discussions or not by Brunet et al. [6]. So, each discussion has 4 tags. I made the discussions and their tags available online. 247 (24.7 %) of these discussions refer to some aspects of design, while 753 (75.3 %) refer to other software development concerns. Also, 605(60.5%) of the discussions refer to some quality attributes, while 395(39.5%) refer to other topics. More detailed statistical analysis of the discussions and tags will be given in the following sections.

Step 3. I used 10-fold cross validation methodology to train and evaluate the multinomial naïve Bayes text classifier for each of the 4 tags. To do that, I first divided discussions into 10 sets with the same size (100 discussions). Then, as training data, I used nine of the sets, and one as test data. I repeated the process of cross-validation 10 times, using each set as test data exactly once. I calculated the average of the accuracy values of 10 executions in order to estimate the accuracy of the classifiers. I evaluated the text classifiers by using this method. In classifying a discussion as related to quality attribute or not related (the First classifier), the accuracy was 70.8%, so it was much more accurate than random classifier which has 50% accuracy. The second classifier which automatically labels the discussions with the related quality attribute had 48.1% accuracy. Since we have 10 different quality attributes, the accuracy of the random classifier, in this case, is 10%, so 48.1% is considered a high accuracy for this classifier. The third classifier which automatically tagged the discussions with the related quality sub-attribute had 42.5% accuracy. Since we have 34 different quality sub-attributes, the accuracy of the random classifier, in this case, is about 3%, so 42.5% is relatively considered a very high accuracy for this classifier. The forth classifier which automatically classifies the discussions as design related or not, had 85.4% accuracy which is much better than random classifier with 50% accuracy.

```
TRAINMULTINOMIALNB(ℂ, 𝔻)
 1  V ← EXTRACTVOCABULARY(𝔻)
 2  N ← COUNTDOCS(𝔻)
 3  for each c ∈ ℂ
 4  do N_c ← COUNTDOCSINCLASS(𝔻, c)
 5     prior[c] ← N_c/N
 6     text_c ← CONCATENATETEXTOFALLDOCSINCLASS(𝔻, c)
 7     for each t ∈ V
 8     do T_ct ← COUNTTOKENSOFTERM(text_c, t)
 9     for each t ∈ V
10     do condprob[t][c] ← (T_ct+1) / (Σ_t'(T_ct'+1))
11  return V, prior, condprob

APPLYMULTINOMIALNB(ℂ, V, prior, condprob, d)
 1  W ← EXTRACTTOKENSFROMDOC(V, d)
 2  for each c ∈ ℂ
 3  do score[c] ← log prior[c]
 4     for each t ∈ W
 5     do score[c] += log condprob[t][c]
 6  return arg max_c∈ℂ score[c]
```

Algorithm 1: Multinomial naïve Bayes text classifier pseudo code

4.2 Answering the research questions

In order to answer the first and second research questions, I did data mining and statistical analysis on the 1000 discussions and their tags. As mentioned earlier, the first tag shows whether the discussion is related to some quality attribute or not. The second tag refers to the quality attribute discussed in the discussion and the third tag determines the quality sub-attribute discussed in the discussion. The fourth tag classifies the discussion as design related or not related. So, the statistical analysis on the first tags will help us find the answer to the RQ1 and statistical analysis on the second and the third tags helps us answer RQ2. You can find my proposed answers to RQ1 and RQ2 and the detailed results of statistical analysis in the following section. In order to answer RQ3, I utilized a logistic regression model. I built the logistic regression model in Python in order to classify the fourth tag given the second and third tags. The results of applying logistic regression and answer to RQ3 will be described in the following section.

5 Results

5.1 RQ1: To what extent do developers discuss quality attributes in open-source projects?

As you see in figure 3, Out of the 1000 discussions that I manually labeled, 605(60.5%) of the discussions refer to some quality attributes, while 395(39.5%) refer to other topics. As a result, most of the discussions in open-source projects which we analyzed were related to some quality attribute.

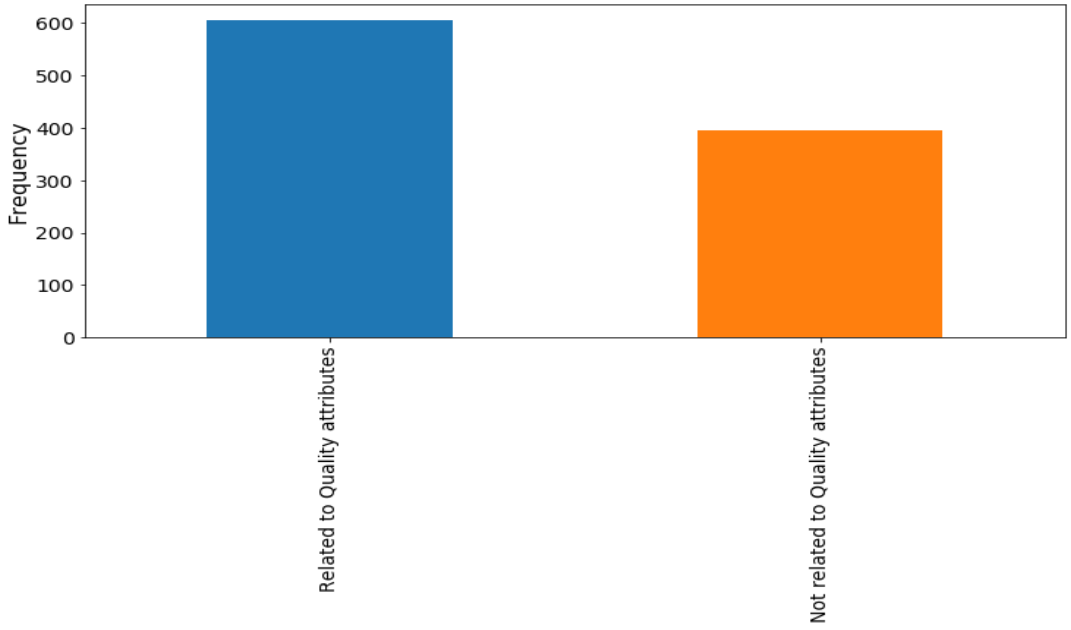

Fig2: frequencies of related and not related to quality attributes discussions

5.2 RQ2: Which quality attributes are discussed in open source projects?

As mentioned in the previous section, statistical analysis on second and third tags help us answer this question. As you see in table 1 and figure 4, out of the 1000 labeled discussions, most of the discussions are related to maintainability, functional suitability, and security. These results were expected because

for most of the projects these quality attributes are included in the utility trees and they are considered as the critical attributes.

Quality Attribute	Frequency
Maintainability	277
Functional Suitability	189
Security	62
Performance	24
Reliability	17
Usability	17
Compatibility	9
Other	7
Portability	3

Table 1: frequencies of discussions related to each quality attribute

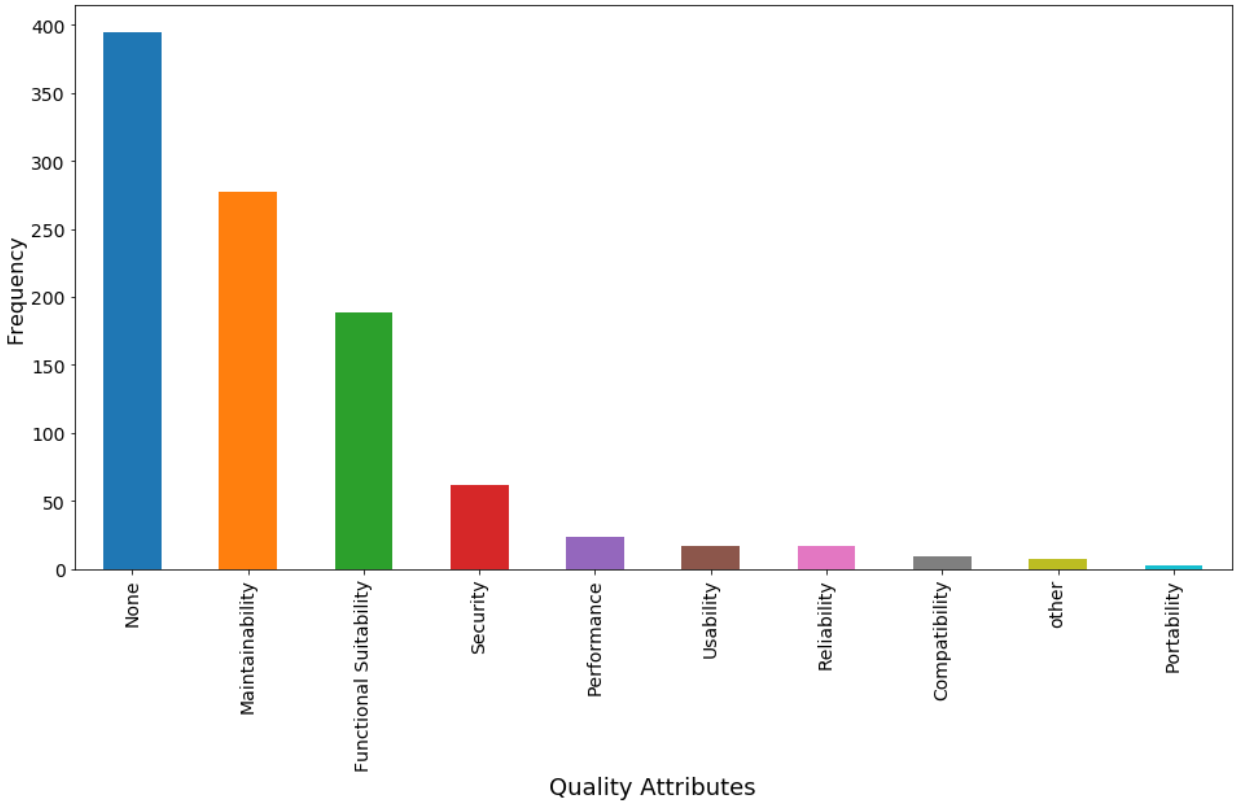

Fig3: frequencies of discussions related to each quality attribute

For fully answering RQ2, we should also consider the statistical analysis related to quality sub-attributes. As you can see in table 2 and figure 5, functional correctness, analyzability, confidentiality, testability, reusability, and modularity are the quality sub-attributes which are discussed in most of the discussions. This shows that these sub-attributes are important for most of the projects owners and the developers try to respect these sub-attributes in their contributions and pull requests. Since these attributes are more important for contributors and project owners, they are discussed a lot in the projects

This fact that functional correctness and analyzability are the most frequent sub-attribute in the discussions was expected because the most important factor for each pull request is that it works correctly. Also, for the developers, the analyzability of the code is very critical because this sub-attribute help them diagnosis the code for the cause of failures and identify the impacts of changes on code and respecting the sub-attribute saves a lot of time and effort of developers and project owners.

Quality sub-attributes	Frequency
Functional correctness	165
Analyzability	131
Confidentiality	59
Testability	52
Reusability	36
Modifiability	34
Modularity	24
Time behavior	16
Functional completeness	12
Functional appropriateness	12
Operability	11
Interoperability	9
Resource utilization	8
Supportability	7
Fault tolerance	7
Availability	6
Adaptability	3
Maturity	3
User interface aesthetics	3
Integrity	2
Learnability	1
Recoverability	1
Accountability	1
Accessibility	1
Appropriateness recognizability	1

Table 2: frequencies of discussions related to each quality sub-attribute

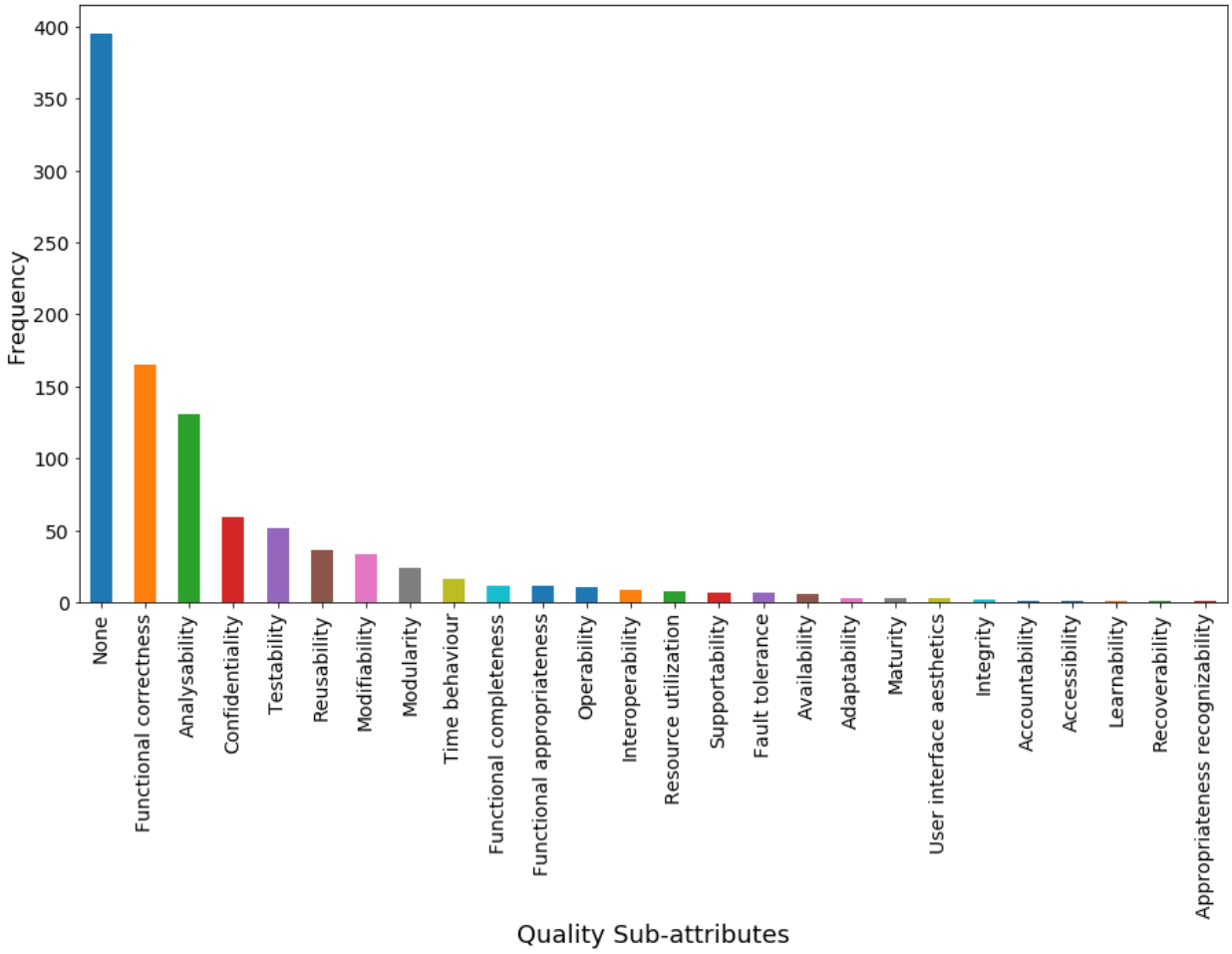

Fig 4: frequencies of discussions related to each quality sub-attribute

5.3 RQ3: To what extent is it possible to automatically classify a discussion as design-related or not given the quality attribute and sub-attribute it discusses.

As mentioned earlier, 1000 discussions were previously tagged as design discussions or not by Brunet et al. [6]. 247 (24.7 %) of these discussions refer to some aspects of design, while 753 (75.3 %) refer to other software development concerns. In order to answer this research question, I built a logistic regression model, as you see in figure 6 it automatically classifies a discussion as design-related or not related given the attribute and sub-attribute are discussed in that discussion. If no attribute is discussed in the discussion, the attribute and sub-attribute will be none. After training and testing, the logistic regression model had 83% accuracy which is considered a high accuracy. As a result, Given the quality attribute and sub-attribute discussed in a discussion, it is possible to classify a discussion as design related or not with 83% accuracy. This finding also shows us that some of the quality attributes and sub-attributes are more related to design than other ones. We should take into consideration that if a discussion is related to quality attributes does not imply that it is also related to discussions. We had several examples in our data set that were related to some quality attribute but they were not related to design. So, as we stated earlier, all of the quality attributes are not related to design in the same extent.

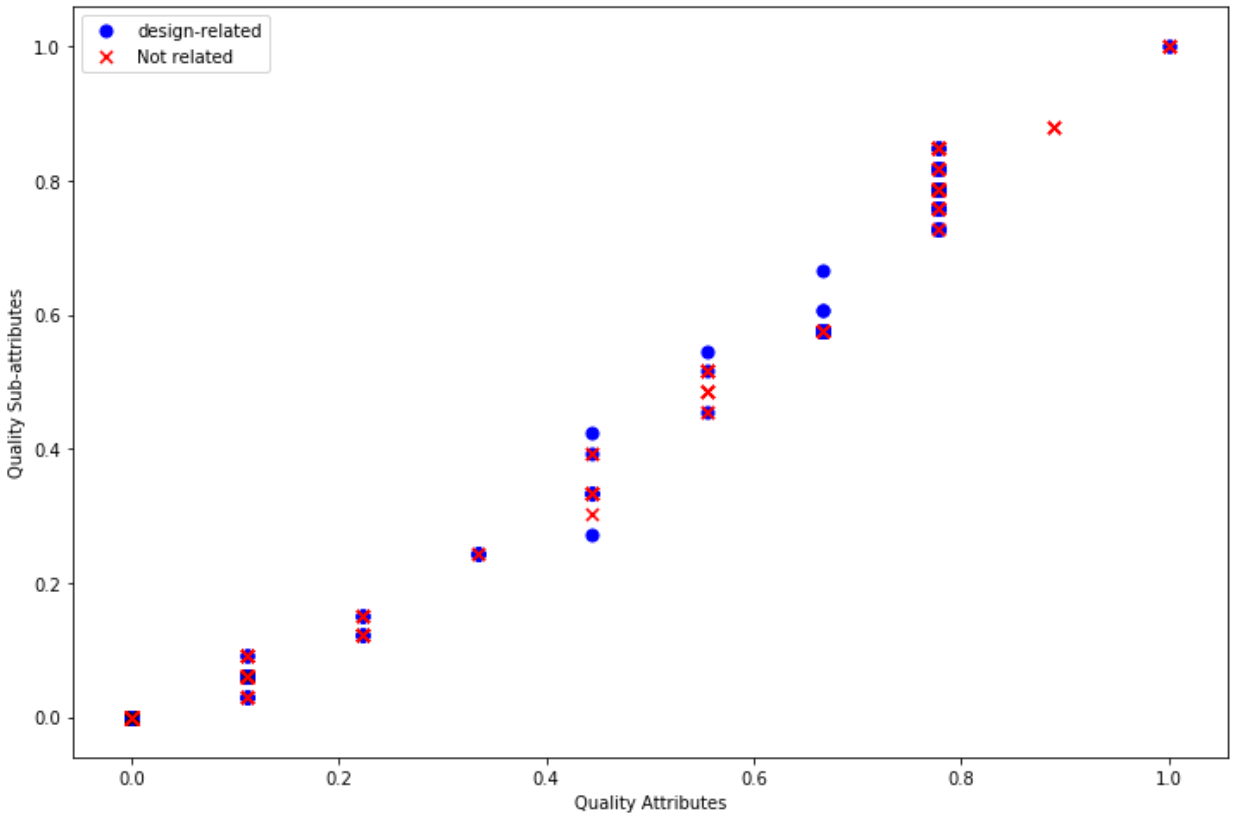

Fig 5: Logistic regression automatically tags a discussion as design-related or not related given the quality attribute and sub-attribute discussed in that. Each attribute and sub-attribute is scaled to a number in [0,1].

6 Threats to validity

Two threats may affect the validity of the text classifier results. First, with just using 1000 discussions, I trained the classifier. Second, the discussions were tagged manually by only one researcher. Ideally, classifying more discussions by a wider range of researchers and practitioners reduce the threats to the validity of the results. There is also another threat that may affect the validity of answers to research questions. I answered the research questions just based on only 1000 discussions that were randomly selected from 5 projects. Maybe these discussions and project are not representative of all the open source projects and discussions. Discussions should be selected from a wider range of projects to decrease this kind of threats. Because I just selected 5 projects, maybe the answers to the research questions are not valid for all kinds of open source projects.

7 Related work

This is not the first work that quantitatively increases knowledge of quality attribute discussions and design discussions in open-source projects and their distribution among discussions. The most similar works to our work have been done by Viviani et al. [8] and Brunet et al. [6]. In developer written discussions, Viviani et al. [8] introduced a new idea of identifying design points at the paragraph level. They have also identified the types of design topics discussed by developers and described how this type of information can help to provide development tools. The variety of the design topics and the number of discussions analyzed in their work are much fewer than this work. Brunet et al. [6] found out that to what

extent developers discuss design in open-source projects and which developers discuss design. However, they did not organize design discussions in categories which we did in our work.

Ebert et al. [9] conducted a case study to investigate the communicative intent of developers' questions during the code review process. They discovered that developers are using questions extensively to convey suggestions. In code review comments, they found that developers also express their cognitive and affective states, such as attitudes of doubts and criticisms or emotions such as anger and surprise. Their work is different from our work because they did not consider quality attributes and sub-attributes in their analysis. Arya et al. [10] looked at the types of information contained in discussions on OSS issues. They have also discovered 16 categories of information using a qualitative content analysis approach that can potentially support OSS participants to retrieve and discover useful and otherwise hidden elements from discussion threads. This work is different from theirs because their categories of information are not related to quality attributes and sub-attributes. However, their approach to building the text classifier and evaluating that is similar to my approach.

8 Conclusion and future work

In open source projects, contributions are usually evaluated based on the extent to which they respect quality attributes. So, a considerable number of discussions in open source projects discuss quality attributes. I presented quantitative results showing that developers address quality attributes through discussions in commits, issues and pull requests. First, I manually labeled 1000 discussions from 5 projects with three tags. The first tag shows a discussion is related to quality attribute or not, the second tag determines the quality attribute which the discussion is related to and the third tag shows the quality sub-attribute related to the discussion. Then, for each of the tags, I built automated text classifiers that use machine learning to label discussions. I evaluated such classifiers using 10-fold cross-validation, These classifiers can be improved in the future works and used to automatically label a large number of discussions. The main observations about the labeled discussions are: i) More than 60% of the discussions are related to quality attributes; ii) 53% of discussions are related to maintainability, functional suitability, and security which shows these quality attributes are very important for developers in the selected 5 projects; iii) 50% of the labeled discussions are about functional correctness, analyzability, confidentiality, testability, reusability, modifiability, and modularity which shows the high importance of these quality sub-attributes for the developers and project owners; and iv) with 83% accuracy, it is possible to automatically classify a discussion as design related or not given the quality attribute and sub-attribute related to the discussion.

In order to answer the research questions, I did data mining and statistical analysis of 1000 manually labeled discussions. But we should take into consideration that to find a fully reliable and more accurate answer to these questions, we need to tag much more discussions. If we tag more discussions, we can train our model on more data, as a result, the automated text classifier will have better accuracy and then we can use the classifier to automatically classify a huge amount of discussions. So, we will have a larger labeled discussions set that is representative of all the kinds of open source projects. By doing data analysis on such a large set of discussions, future works are able to find more accurate results that show us that to what extent developers discuss quality attributes in open-source projects and find out the types of the quality attributes and sub-attributes which are discussed in open source projects. As a result, in this work, because of the limited number of the labeled discussions and low accuracy of the text classifier, I did not use the text classifier to automatically label the discussions. However, building the text classifier

in this work is the first step, and future works should improve the accuracy of that by manually labeling more discussions. When they build a text classifier that is accurate enough, they can use that to automatically label a huge set of discussions. So they will be able to do data analysis on a larger data set and gain more accurate answers for the research questions. Also, in this work, just one researcher manually labeled the discussions. Future works can have a broader range of researchers and practitioners manually classifying more discussions.

9 Acknowledgments

I want to give my deepest appreciations to Professor Neil Ernst and Omar Elazhary, who provided us with the great opportunity to meet this great field of computer science.

10 References

[1] J. Tsay, L. Dabbish, and J. Herbsleb. Let's talk about it: Evaluating contributions through discussion in GitHub. In FSE, pages 144{154. ACM, 2014.

[2] Mockus, A., Fielding, R.T. and Herbsleb, J.D. 2002. Two case studies of open source software development: Apache and Mozilla. ACM Trans. Softw. Eng. Methodol. 11, 3 (Jul. 2002), 309–346.

[3] Susan Elliott Sim and Richard C. Holt. 1998. The Ramp-up Problem in Software Projects: A Case Study of How Software Immigrants Naturalize. In Proceedings of the 20th International Conference on Software Engineering (ICSE '98). IEEE Computer Society, Washington, DC, USA, 361–370. http://dl.acm.org/citation.cfm?id=302163.302199

[4] I. Steinmacher, I. Scaliante Wiese, T. Conte, M. A. Gerosa, and D. Redmiles. 2014. The Hard Life of Open Source Software Project Newcomers. In Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2014). ACM, New York, NY, USA, 72–78. https://doi.org/ 10.1145/2593702.2593704

[5] http://iso25000.com/index.php/en/22-english/iso-iec-25010?limit=3&limitstart=0

[6] Jo~ao Brunet, Gail C. Murphy, Ricardo Terra, Jorge Figueiredo, and Dalton Serey. Do developers discuss design? In Proceedings of the 11th Working Conference on Mining Software Repositories, MSR 2014, pages 340{343, New York, NY, USA, 2014. ACM.

[7] Georgios Gousios. The GHTorrent dataset and tool suite. In Proc. of Working Conference on Mining Soft. Repositories, MSR'13, pages 233–236, 2013.

[8] G. Viviani, C. Janik-Jones, M. Famelis, X. Xia, and G. C. Murphy, "What design topics do developers discuss?" in ICPC, 2018.

[9] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "Communicative intention in code review questions," in ICSME, 2018.

[10] Arya D, Wang W, Guo JL, Cheng J. Analysis and Detection of Information Types of Open Source Software Issue Discussions. arXiv preprint arXiv:1902.07093. 2019 Feb 19.