Do Developers Discuss Design?

João Brunet
Federal University of Campina Grande, Brazil
jarthur@dsc.ufcg.edu.br

Gail C. Murphy
University of British Columbia, Canada
murphy@cs.ubc.ca

Ricardo Terra
Federal University of Lavras, Brazil
terra@dcc.ufla.br

Jorge Figueiredo
Federal University of Campina Grande, Brazil
abrantes@dsc.ufcg.edu.br

Dalton Serey
Federal University of Campina Grande, Brazil
dalton@dsc.ufcg.edu.br

ABSTRACT

Design is often raised in the literature as important to attaining various properties and characteristics in a software system. At least for open-source projects, it can be hard to find evidence of ongoing design work in the technical artifacts produced as part of the development. Although developers usually do not produce specific design documents, they do communicate about design in different ways. In this paper, we provide quantitative evidence that developers address design through discussions in commits, issues, and pull requests. To achieve this, we built a discussions' classifier and automatically labeled 102,122 discussions from 77 projects. Based on this data, we make four observations about the projects: i) on average, 25% of the discussions in a project are about design; ii) on average, 26% of developers contribute to at least one design discussion; iii) only 1% of the developers contribute to more than 15% of the discussions in a project; and iv) these few developers who contribute to a broad range of design discussions are also the top committers in a project.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms

Design, Documentation

Keywords

Design Discussions, Machine Learning, Empirical Study

1. INTRODUCTION

Design can be defined as an artifact and also as an activity in software development [1]. As an artifact, design is a representation of how a portion of the code should be organized. As an activity, design is the process of discussing the structure of the code to organize abstractions and their relationships. In this context, practitioners and researchers regard design decisions as a fundamental concern when developing software [2, 3]. They frequently credit good design as easy to maintain and evolve, and as a sign of internal quality. For this reason, ideally, design decisions should be communicated effectively among the development team.

Open source developers share the majority of the information in a project in written form. Despite a plethora of mailing list archives, issues, commit information, and other resources associated with an open-source project, it is not usual to find a design document in project's archives. For example, we inspected *docs* folder, *wiki* pages, and main web sites of the top 90 popular projects in GitHub[4] and could not find any design documentation in 61 (68%) of them. Even considering those projects that have some documentation about their design, we could only find explicit technical artifacts (e.g. UML diagrams) in 7 (9%) projects.

Although no specific artifacts related to design can be detected in open-source projects, the other media used for communication, such as issues, commits' comments, and pull requests may include design concerns and discussions. To understand if design information is discussed and shared in these other forms in open source projects, we conducted an empirical study on 77 of the top popular projects in GitHub to provide quantitative evidence on how developers drive design discussions. Because developers usually approach structural aspects of design [5], such as communication constraints among classes, we focus our study on such aspects. In this context, we seek to investigate two questions:

- RQ1: To what extent do developers discuss design in open-source projects?
- RQ2: Which developers discuss design?

To answer these questions, we developed the first contribution of this work – the development and evaluation of a prototype based on machine learning technique to automatically identify design discussions. Then, using this prototype, we provide quantitative evidence that, on average, 25% of the discussions in a project mention some design aspect and 26% of developers contribute to design discussions. In addition, we found that very few developers contribute to a broader range of design discussions in a project. We found

MSR'14, May 31 – June 1, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2863-0/14/05...$15.00
http://dx.doi.org/10.1145/2597073.2597115

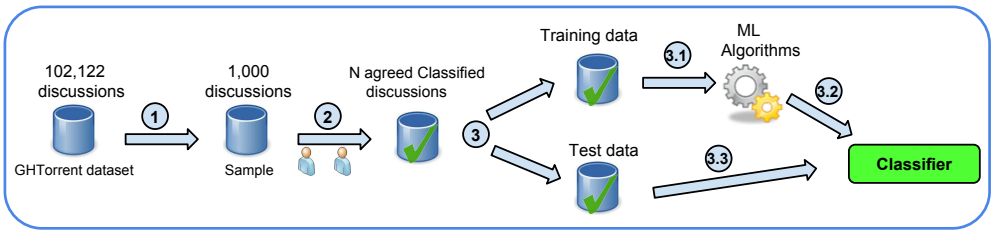


Figure 1: Methodology applied to build the design discussion classifier.

a strong correlation (74%) between commits and design discussions contributions, suggesting that developers who contribute with more commits tend to discuss more about the design of the system. These two contributions may be useful for several purposes. For instance, one could use this information about which developers are involved in design discussions to drive structural refactorings to this small group of developers responsible for design. As another example, researchers can use our tool support to automatically uncover design rules.

We organize this paper as follows. Section 2 describes our experimental design, including definitions about design discussions, dataset, and the procedures and measures employed. Section 3 shows results for our classifier and early results on analyzing design discussions. Section 4 discusses some important points related to our results as well as the relevance of this work. Section 5 briefly discusses related work, while Section 6 presents our final remarks and discusses future work.

2. STUDY DESIGN

In our study, we consider a discussion to be a set of comments on pull requests, commits, or issues. Because we were interested in discussions, we analyzed those pull requests, commits, and issues with more than one comment. Also, we consider a discussion to be about design if it contains at least one comment referring to some design concern. As we said before, we focus our study on structural characteristics of a software design. As an example of such structural characteristic, developers usually discuss about avoiding coupling among unrelated classes or applying a specific design pattern to solve a design issue.

2.1 Data Set

Of the 90 projects present in the GHTorrent data set [4], we discarded 13 projects with less than 50 discussions. We chose the 77 projects with more than 50 discussions to work with a reasonable amount of data. The more discussions present in the projects, the more likely they have design discussions. Due to the fact that we were interested in the degree of design discussions, we made such decision. In addition, to simplify our analysis, we treated projects and their forks as one single project. In summary, our data set includes 77 projects and 102,122 discussions.

2.2 Methodology

2.2.1 *Building the Classifier*

Figure 1 shows the steps we conducted to build our design discussions classifier:

Step 1. We randomly selected 5 of the 77 projects. They are: *BitCoin*, *Akka*, *OpenFrameworks*, *Mono*, and *Twitter-Finagle*. Then, we randomly selected 200 discussions from each of these 5 projects, totaling 1,000 discussions.

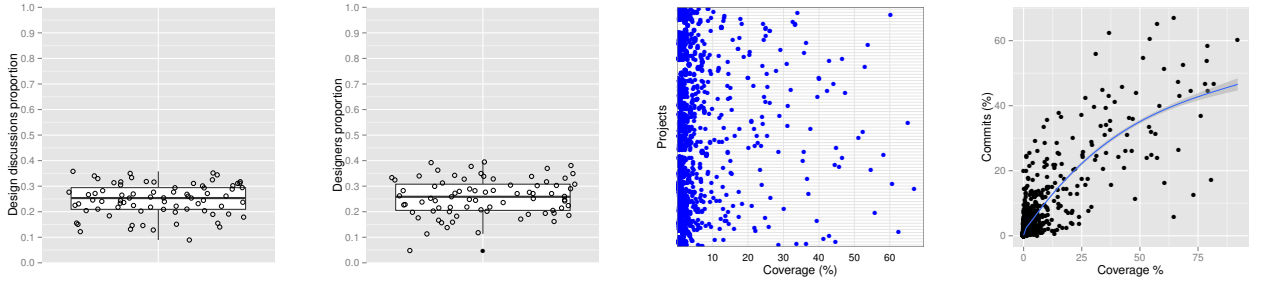
Step 2. We (two of the authors of this paper) classified the same set of 1,000 discussions separately. We tagged the discussions as design discussions or not. To avoid bias, before the classification, we did not specified or discussed any specific rules to classify the discussions. However, we stated that we would focus on structural design aspects. After the manual classification, we selected for training only the 967 discussions in which both classifications matched. 226 (23%) of these discussions refer to some design aspect, while 741 (77%) refer to other concerns related to software development.

Step 3. We used 10-fold cross validation methodology to train (steps 3.1 and 3.2) and evaluate (step 3.3) our classifier. That is, we randomly partitioned discussions into 10 equal size sets (96 discussions). Then, we use nine of these sets as training data and one of them as test data. We repeated the cross-validation process 10 times, using each one of the sets exactly once as test data. We use the mean of the 10 executions to produce an estimation of our classifier's accuracy. Using this method, we evaluated Naive Bayes and Decision Tree classifiers. We removed words from an English stoplist of common short words. As feature selectors to these classifiers, we used a combination of word frequency and bigrams. Besides the standard usage of word frequency, we also used bigrams because researchers have shown that these methods can significantly improve the results of text classification [6, 7]. For instance, in our context, the bigram "exposes API" is more representative than the word "exposes" isolated or combined to other non-related word.

2.2.2 *Answering Research Questions*

After we have built confidence in our classifier, we rely on it to label all 102,122 discussions in the data set. Then, we analyzed the design discussions to answer our research questions. For the first question, we simply measure, for each project, the proportion of design discussions over all discussions. For the second question, we investigated design discussions and commits to determine:

- the ratio between the number of developers that contribute to design discussions and the number of committers in a project;
- the proportion of all design discussions in a project to which a developer has contributed, which we name *Coverage*. For instance, if a project has 10 design discussions and a developer contributes to 5 of these discussions, the developer has 0.5 of coverage.



(a) Proportion of design discussions per project.

(b) Proportion of design discussions contributors.

(c) Developers' design discussions coverage.

(d) Developers' coverage and commits

Figure 2: Empirical Results

3. RESULTS

After executing the 10-fold cross validation, the results show that Decision Tree outperforms the Naive Bayes method. The former achieved $94 \pm 1\%$ accuracy[1], while the latter achieved $86 \pm 3\%$. For this reason, we decided to use the Decision Tree classifier to automatically label the remaining discussions.

RQ1: To what extent do developers discuss design? Of the 102,122 discussions, our classifier labeled 25,123 (25%) as design discussions. As examples, our classifier labeled as design concerns the following comments: "*I'd be surprised if this is the way to create RoutedActorRefs*" and "*We have the dependency issue that ActorSystem need to know about all extensions*". Comments such as "*See code style guide. We use underscore style for variable names.*" were not labeled as design.

Figure 2(a) shows the proportions of design discussions per project. Following the overall proportion, $25 \pm 6\%$ of discussions within a project refer to some design aspect. As we can see by the flattened boxplot, this is a common pattern among projects. Also, this result reinforces the confidence in our classifier, once it is similar to our training data, which shows a proportion of 23% of design discussions.

RQ2: Which developers discuss design? In total, we analyzed data regarding 22,789 developers from the 77 studied projects. 8207 (36%) of these developers contribute to at least one design discussion, while 14,582 (64%) do not. Our first step to answer this question was to investigate the proportion of developers that contribute to design discussions in a project. Figure 2(b) shows these results considering each project. A mean of $26 \pm 7\%$ of developers per project contribute to at least one comment regarding a design aspect. We inspected the projects with proportion above 30% (e.g., *Bitcoin*, *Django*, *Rails*, *Symfony*). These projects have a large number of committers and they are well known and established open source communities, which may explain the fact that more developers contribute to their design.

In a second step, to further investigate developers' contribution, we measure the coverage of each developer. Figure 2(c) shows the coverage of developer per project. Each point in the graph represents a developer of a project in y axis. As we can see, the majority of developers contribute to less than 10% of design discussions. In fact, 99% of developers contribute to less than 15% of all design discussions in their respective projects. This results lead us to conclude that very few developers contribute to a broader range of design discussions, while most of the developers contribute to few design discussions.

Several factors might lead to the scenario in which very few developers contribute to a broad range of design discussions. This scenario suggests that these developers play a central role in their projects. We took a step forward to investigate one of the factors that might be correlated to developer's ability to discuss design in a broad range. To do so, we measured the relationship between the proportion of developers' commits and their respective coverage. Figure 2(d) plots the coverage against the percentage of commits of all developers studied. The line represents the best fit for the data with 95% of confidence interval. We used Spearman's method and found a strong correlation (74%) between these two variables. As we can see, the developers with high level of Coverage are also the developers responsible for a high percentage of commits in their respective projects.

4. DISCUSSION

Walking architecture. Our results show that a very small number of developers have high levels of design discussions coverage. This result is aligned to a previous work that name this small set of developers as "walking architecture" [8]. The term refers to central developers who evaluate changes to code that affects design while at the same time update knowledge about design decisions. We argue that further work should invest in driving design issues to central developers. Through our classifier, researchers may use information about design discussions to build mechanisms to improve communication among developers. When developers discuss design often, they update their knowledge about the system and may achieve *Conceptual Integrity* — the uniformity of the understanding that the development team has about the software [9].

Developers' role. The high correlation between commits and design discussion coverage reveals that there is no clear separation between designers and developers role in these projects. Developers that discuss design in a broad range are the ones who most contribute to the code of the studied systems. The possible simple explanation for this scenario is the cumulative knowledge of code and design that these developers gain overtime. As time goes by, naturally these developers are responsible for the discussions, once

[1]The standard metric to evaluate classifiers, which stands for the percentage of instances correctly labeled.

they have a deeper knowledge about the system than the other committers.

Developers' contribution. We were somewhat surprised by the proportion of developers that do not contribute to design discussions (64%). Due to the fact that we analyzed open-source projects, we were expecting contributions from more developers. Specially because these projects are all hosted in GitHub, which provides lightweight mechanisms to promote comments and discussions during software development.

Classifier's Performance. One possible drawback of using Decision Tree classifier is the performance issue. While the Naive Bayes 10-fold cross validation took only 9 seconds to finish, the Decision Tree validation took approximately 4 hours. This happens in the tree construction step of the algorithm, which takes a meaningful amount of time to build the branches and rules of the tree, since the combination of words and bigrams generates several tree nodes. However, we only need to execute this process once, which pays-off its cost over time. For this reason, we decided to use Decision Tree classifier to label the remaining discussions of our study.

Threats. Two threats might influence the results of our classifier. First, we trained our classifier with approximately 1% of all discussions analyzed. Second, only two researchers manually classified the discussions. Ideally, we would like to have a broader range of researchers and practitioners classifying more discussions. However, we believe that we achieved a reasonable and reliable amount of training data. In addition, because we focus our classification on structural properties of design, our classifier may have missed discussions about other aspects related to design, such as dynamic and deployment concerns.

5. RELATED WORK

To the best of our knowledge, this is the first work that quantitatively raises knowledge about design discussions in open-source projects and their distribution among developers. However, other researchers have investigated how developers deal with design and architecture concerns. Lange and Chaudron [10] interviewed 80 architects and observed that 66% of them employ UML diagrams to perform design activities. Cherubini et al. identified that developers usually externalize design decisions in temporary drawings that are lost over time [11]. Unphon and Dittrich conducted 15 interviews to qualitatively understand how developers drive architecture and design concerns in software companies [8]. Two of their results are closely related to ours. First, they observed the "walking architecture" phenomenon, whose existence seems to be empirically supported by the data we analyzed. Second, they observed that design/architecture documentation might not be used during software development due to the usage of other media. In our work, our data support that, for open-source projects, such media can be discussions in issues, pull requests, and commits. This last result is in conformance with Guzzi et al.'s analysis, which found that developers' mailing list is not the main player in OSS project communication, as it also includes other channels such as the issue repository [12].

6. SUMMARY AND FUTURE WORK

Developers need to maintain, verify, and discuss design during software development. In this paper, we presented quantitative results indicating that developers address design through discussions in commits, issues and pull requests. We first built an automated classifier that employs machine learning to label discussions as design or not. We evaluated such classifier using 10-fold cross validation, achieving $94 \pm 1\%$ of accuracy. Then, using our classifier, we automatically labeled 102,122 discussions. The main observations about these discussions are: i) 25% of discussions in a project are about design; ii) 26% of developers contribute at least to one design discussion; iii) few developers contribute to a broad range of design discussions. In fact, 99% of developers contribute to less than 15% of design discussions; and iv) the very few developers who contribute to a broad range of design discussions are also the top committers in a project (correlation 74%).

In this work, we did not organize design discussions in categories. As a main future work, we intend to achieve this. As we could observe, the subject of design discussions varies. For instance, some discussions are related to constraints involving classes and interfaces usage, while others to the suitability of design patterns to solve particular design issues. After this categorization, we intend to identify which design aspects attracts more attention from developers. This will require us to analyze the distribution of developers per design discussion and identify the topic of such discussions. In a nutshell, we believe that such outcomes might assist in the prioritization of design issues, for example.

7. REFERENCES

[1] David Budgen. *Software design*. Pearson Education, 2003.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[3] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development,*. Pearson Education India, 2012.

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

[5] Christine Hofmeister, Robert L Nord, and Dilip Soni. Describing software architecture with uml. In *Soft. Architecture*, pages 145–159. Springer, 1999.

[6] Johannes Fürnkranz. A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence*, pages 1–10, 1998.

[7] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. *Ann Arbor MI*, pages 161–175, 1994.

[8] Hataichanok Unphon and Yvonne Dittrich. Software architecture awareness in long-term software product evolution. *Journal of Systems and Soft.*, pages 2211–2226, 2010.

[9] Frederick P Brooks. *The mythical man-month*. Addison-Wesley Reading, 1975.

[10] C.F.J. Lange, M. R V Chaudron, and J. Muskens. In practice: Uml software architecture and design description. *IEEE Soft.*, pages 40–46, 2006.

[11] Mauro Cherubini, Gina Venolia, Rob DeLine, and Andrew J Ko. Let's go to the whiteboard: how and why software developers use drawings. In *Proc. of SIGCHI conference on Human factors in computing systems*, pages 557–566, 2007.

[12] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie van Deursen. Communication in open source software development mailing lists. In *Proc. of International Workshop on Mining Software Repositories*, pages 277–286, 2013.