What Design Topics do Developers Discuss?

Giovanni Viviani
University of British Columbia
vivianig@cs.ubc.ca

Calahan Janik-Jones
University of Toronto
cal.janik.jones@mail.utoronto.ca

Michalis Famelis
Université de Montréal
famelis@iro.umontreal.ca

Xin Xia
Monash University
xin.xia@monash.edu

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

ABSTRACT

When contributing code to a software system, developers are often confronted with the hard task of understanding and adhering to the system's design. This task is often made more difficult by the lack of explicit design information. Often, recorded design information occurs only embedded in discussions between developers. If this design information could be identified automatically and put into a form useful to developers, many development tasks could be eased, such as directing questions that arise during code review, tracking design changes that might affect desired system qualities, and helping developers understand why the code is as it is. In this paper, we take an initial step towards this goal, considering how design information appears in pull request discussions and manually categorizing 275 paragraphs from those discussions that contain design information to learn about what kinds of design topics are discussed.

ACM Reference Format:
Giovanni Viviani, Calahan Janik-Jones, Michalis Famelis, Xin Xia, and Gail C. Murphy. 2018. What Design Topics do Developers Discuss?. In *Proceedings of ICPC '18: 26th IEEE/ACM International Confernece on Program Comprehension (ICPC '18).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3196321.3196357

1 INTRODUCTION

When working on a project, developers constantly struggle with the balance between writing the code required to provide desired behaviour and respecting the design of a system. This task is made even harder because, at least in the case of open source systems, design information is rarely documented explicitly. Instead, it is scattered across different kinds of artifacts, including discussions taking place on issue tracker systems and on pull requests. For instance, Brunet et al. found that a large number of discussions are about design [2]. The work of Tsay et al. reinforced this finding, showing how these discussion can often take a leading role on how the design of a system evolves [16].

Newcomers particularly struggle with determining and adhering to the current design of the project [3, 12]. If it was possible to identify where design was being discussed, and then to represent and make use of such information, many software development tasks could be eased. For instance, if during a pull request discussion, we could determine automatically the design topics being examined, we could invite core project members with appropriate design expertise to participate. Such invitations might in turn decrease the time and effort that goes into reviewing a pull request. As another example, we envision tools to help developers understand why a piece of code is designed in a particular way, by identifying and extracting design information from discussions about related past code changes.

Listing 1: Paragraph that contains design information.

```
1 leaky abstraction in the sense that your abstraction is
  saying too much about the implementation -- you're
  declaring to the world that you had to make compromises
  on your API to get other outcomes (performance), there
  has to be a tradeoff between pure perf and the best
  internal implementation and the API we expose to users
  and I'm here representing the API and this is that
  tradeoff discussion
```

Listing 2: Paragraph not related to design.

```
1 Ok. I was pretty keen on getting 1.7.2 within a week or
  so with a fix to a shared build. Guessing 2.0 might make
  that easier since we'd branch off to master/1.x/2.x?
```

Written developer discussions captured in issue trackers and pull requests can contain a wide range of information, not all of which touches on system design. The discussion snippets in listings 1 and 2 provide a sense of the range of information in such discussions. On the one hand, the snippet in listing 1 discusses the structure of a method, and thus considers the implications of the method's design. On the other hand, the snippet in listing 2 (part of the same pull request discussion) considers build and release information; we therefore do not consider the information in it as part of the system design.

As a step towards the identification of design information in discussions, we propose and investigate the concept of a *design point* (DP), defined as *a piece of a discussion relating to a decision about a software system's design that a software development team needs to make*. We investigated the presence of such design points in pull request discussions (Section 2). These discussions consist of a sequence of comments made by different individuals where each comment can consist of a number of paragraphs and code snippets. We found that paragraphs are a useful level of granularity at which to localize design points. We annotated the presence of design points in 10,790 paragraphs from 3 large open source projects, discovering 2,378 paragraphs that contain design points.[1]

[1]The dataset is available at https://www.cs.ubc.ca/~vivianig/icpc18dataset

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICPC '18, May 27–28, 2018, Gothenburg, Sweden
© 2018 Copyright held by the owner/author(s). Publication rights licensed to Association for Computing Machinery.
ACM ISBN 978-1-4503-5714-2/18/05…$15.00
https://doi.org/10.1145/3196321.3196357

We then wondered whether there are certain kinds of design information that appear consistently in these discussions. To investigate this question, we applied an open coding approach on 275 paragraphs selected from the 2,378 paragraphs identified as containing design information from the dataset to try to identify design topics (Section 3). The two coders agreed on 8 design topics ranging from *Implementation issues* to *Performances discussions*. This short paper makes three contributions: (1) it introduces the concept of a *design point* to capture where design information appears in written developer discussions, (2) it provides a dataset of 2,378 paragraphs from 10,790 pull request discussions where design points have been identified, and (3) it reports on the categories of design topics found in 275 paragraphs with design points coded by two individuals.

The two facts that design information occurs within paragraphs of a pull request discussion and that there are consistently recurring design topics lend credibility to the vision of automatically identifying design information in written developer discussions and building tools to usefully build upon this information. In Section 4, we explore some of the next steps to make the identification and use of design points reality.

2 DESIGN POINTS

What constitutes design for a software system is not well defined in the literature. For example, the term *design* is used to refer to how to structure aspects of a system, whether at a coarse scale architecturally [7] or at a finer grained scale, such as with a design pattern [5]. Its use is diverse and ranges from the creation of standardized documents [1] to the choice of a particular sorting algorithm [4]. To capture the breadth of design information, we choice a broad definition for design points: a design point (DP) is *a piece of a discussion relating to a decision about a software system's design that a software development team needs to make*. This broad definition is useful as we explore how design occurs in discussions so as not to prematurely limit the range of design information considered.

We chose to investigate design points in pull request discussions because pull requests have been shown to include design information [2]. To understand what characteristics might indicate the presence of DPs in discussion, we applied an iterative annotation process to pull request discussions from three open source systems. We chose pull requests with lengthy discussions to maximize the likelihood that design would be discussed, as opposed to a pull request that might be quickly accepted by a development team with little discussion. We chose to focus on the `Node.js`, `Rust` and `Rails` projects because they are large open source projects of different kinds, underlying technologies, and levels of maturity, thus allowing us a diverse perspective on design.

In an initial phase, two of the authors independently annotated the same pull request discussion looking for paragraphs containing DPs. Each one created an annotation guide to document annotations used, steps taken and assumptions made. This was followed by a consolidation phase, where the two independently created annotation guides were merged. During the consolidation phase, we opted to restrict the search of DPs to paragraphs. On the one hand, working with entire comments is too coarse, since different paragraphs within a comment sometimes discuss different topics. On the other hand, working with single sentences is too fine, as a DP can easily span over multiple sentences.

Table 1: Statistics for the annotated pull requests

Pull Request	#Paragraphs	#DPs	% Paragraphs with DPs
Node.js	3,963	985	24%
Rails	3,201	722	22%
Rust	3,626	770	20%
Total	**10,790**	**2,475**	**22%**

Following the consolidation phase, two of the authors annotated four pull requests. We computed the Cohen's Kappa Coefficient across the annotations, resulting in a value of 0.52. Given this "moderate agreement" level [8, 9] in the annotations, we then proceeded to apply the annotation process to more pull requests.

In all, three of the authors annotated a total of 34 pull requests: 14 from `Node.js`, 10 from `Rails` and 10 from `Rust`. Of these pull requests, 18 were successfully merged into the project; 16 were rejected by project developers. The annotated pull request data set consists of 10,790 paragraphs of which 2,378 contained at least one DP, for a total of 2,475 DPs. We summarize the collected data in Table 1. For each project, we list the number of paragraphs (#Paragraphs), the number of DPs (#DPs), and the percentage of paragraphs containing at least one DP (% Paragraphs with DPs).

Our results match the finding of Brunet et al. that, in a larger study of 102,122 discussions, found out that roughly 25% of discussions in a project are about design [2].

3 DESIGN TOPICS

Knowing that a substantial number of paragraphs in pull requests across three projects include DPs, we then wondered what topics relevant to design were being discussed.

Initially, from the 2,378 paragraphs identified as containing a design point in the previous step, we randomly sampled 50 paragraphs. We then applied open coding [13] to determine the design topics being discussed. Two of the authors coded the paragraphs separately, followed by a meeting where the two authors discussed the differences in their coding. 50 additional paragraphs were then randomly sampled and coded. We repeated this process three times, for a total of 150 paragraphs. Between each iteration, the two coders discussed the difference in topics, until a common set of of topics was determined and no new design topic appeared. Finally, the two coders, sampled a last set of 75 paragraphs, which they then coded and for which we measured the inter-rater agreement using Cohen's Kappa Coefficient. The inter-rater agreement was 0.64, a value considered to be "substantial agreement" [8, 9].

Table 2 summarizes topics that were considered during the process. The left side of the table (after the double line) list the main topics that two coders considered over the entire set of 275 paragraphs. The number of times each coder found each topic is indicated. For brevity, we do not include topics that appeared less than 5 times. Although there are similar numbers for some topics, many topics have a fairly large difference. These differences can be explained by the fact that Coder #2 tended to differentiate more between design topics, thus spreading the observations to a larger set of topics. In

other words, Coder #2 tended to have more topics that were used only once or twice, and therefore not shown in the Table.

The right side of Table 2 shows the final set of design topics found in the dataset, each with a short description of its meaning. Specifically, we list the results of the coding of the last 75 paragraphs, which took place after the two coders had finalized the common set of topics with significant inter-rater agreement. We show only the number of occurrences for which both coders agreed. Therefore, the total number of occurrences is 53, rather than 75. Given the size of the sample, some of the topics are present only in small numbers, we observe that in the larger set, on the left side, those topics are represented in a more significant number of cases. We thus consider them as relevant. Some of the topics of the left side had been grouped in single categories, as shown in the Table: for example, *Robustness* also includes *Safety*.

The determination of design topics presented multiple challenges. First of all, developers rarely explicitly state the topic of a paragraph. Thus, the coders often had to rely on their own intuition to understand what topic the paragraph was about. The decision of topic was made harder by the fact that paragraphs may not always make sense without the context of paragraphs around them.

The coding process also required a large amount of time. Our investigation was made possible thanks to the reduced sample size, but the amount of work that would be required to obtain enough data to use with an automated tool would make a manual categorization infeasible. We discuss in Section 4 the possibility to automatize this process.

4 BUILDING ON DESIGN POINTS

Developers discuss design in written asynchronous communication channels, such as issues and pull requests. In this paper, we have identified that a paragraph is a useful granularity of a discussion in which to identify design. We refer to such a paragraph as having a design point. We sampled a number of design points from pull request discussions to consider whether there was any consistency in the design topics discussed by developers. All of this work relied on manual annotations. To make our vision of creating tools and techniques that can make use of design points, the determination of whether a paragraph is a design point and what kind of design is discussed at a design point requires automation. We describe the kinds of automation we envision (Sections 4.1 and 4.2) and sketch, as one example, how such automated detection could be applied to help with recommending contributors to a discussion (Section 4.2). We finally also hypothesize how useful could be investigating more the relation between design points (Section 4.3).

4.1 Design Point Detection

Previous work has considered how to identify automatically which discussions are about design. Brunet et al.[2] and Shakiba et al.[11] used supervised learning techniques to determine whether entire discussions were about design or not.

We believe that design points provide a way to lower the granularity and directly identify which paragraphs contain design information. We are currently working on developing an automatic approach to detect the presence of design points in paragraphs of discussions among developers. In comparison to earlier approaches, this method would enable the more accurate identification of the location of design information in discussions. The more accurate and granular the detection method, the more likely it is that meaningful design information can be extracted, represented and used by other tools.

We are working towards the goal of automated determination by expanding our annotations to add additional labels to the design points we have localized in the paragraphs analyzed. For example, we are labelling if the author of that specific paragraph had previously been invited to the discussion, since it may be more likely that he/she is going to make to meaningful comment if invited. Based on such additional annotations, we will investigate whether a supervised learning approach can identify which paragraphs contain a design point.

4.2 Detection of Design Topics

To our knowledge, no existing work attempts to identify design topics in discussions at the paragraph level.

We believe design-related keywords can be extracted from paragraphs in discussions automatically, based on techniques such as TextRank[10]. Those keywords can become associated with manual annotations about design topics, and a supervised learning approach can then be used to determine automatically the topic of a paragraph with a design point.

If the topic of design could be identified automatically, this information could be used to build tools to help developers. Consider, for instance, the fact that discussions for pull requests can become lengthy. For example, pull request #4765 of the `Node.js` project has a total of 223 comments. In such discussions, it is not uncommon for there to be many back-and-forth comments between developers about a design topic, such as performance. Resolving the design issue in these cases can require the solicitation of another project member who is more expert on the topic to weigh in. If the design topic can be determined automatically as suggested above, a recommender tool could be provided to automatically introduce the expert project member. Such a recommender would go beyond existing work that recommends reviewers only at the beginning of a pull request based on the source code modifications [6, 14, 15, 17]. The recommender we propose would go beyond this existing work to include information found in discussions about a code review; information that we believe has been largely untapped.

4.3 Design Information Extraction

Even more meaningful tools to aid developers might be built if the design information being discussed could be represented more specifically. For example, instead of just determining the topic of the design information, the *actual* design information could be extracted. Imagine for listing 1 that in addition to determining the design point is about API and performance, information from other parts of the discussion (not shown in the listing) about the particular tradeoffs with the method design could be determined. Often, such information, requires synthesizing multiple arguments and positions raised by several developers across multiple design points. By using the structure of the discussion, a synthesis of more precise design information could be determined and used to help describe to developers what tradeoffs were made in the determination of the code as written.

5 SUMMARY

To help developers cope with meeting simultaneous goals of adding and modifying a system to meet desired behaviour and respecting the system design, new tools are needed that enable a developer to easily access design information. In this paper, we have taken some initial steps towards the long-term goal of providing such tools. We have introduced the new idea of identifying design points at the level of paragraphs in developer written discussions, arguing how this result goes beyond existing work that limits such identification to the discussions in their entirety.

Moreover, we have identified the kinds of design topics discussed by developers, which others have not yet considered. We have described how this kind of information can aid in the provision of development tools and sketched a future in which more precise design information might be extracted. We present these early results to start a dialogue in the community about how to best define a design point and how best to represent and use information that appears in a design point.

ACKNOWLEDGMENTS

The authors wish to thank Georgios Gousios for helping us create the corpus of GitHub discussions. The authors also acknowledge an NSERC Discovery Grant that was used to fund this research.

REFERENCES

[1] 2009. IEEE Standard for Information Technology–Systems Design–Software Design Descriptions. *IEEE STD 1016-2009* (July 2009), 1–35. https://doi.org/10.1109/IEEESTD.2009.5167255

[2] J. Brunet, G. C. Murphy, R. Terra, J. Figueiredo, and D. Serey. 2014. Do developers discuss design?. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. 340–343.

[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] Giorgio G. and Fabio R. 2001. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing* 19, 9 (2001), 699 – 707. https://doi.org/10.1016/S0262-8856(01)00045-2

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[6] H. Kagdi, M. Hammad, and J. I. Maletic. 2008. Who can help me with this source code change?. In *2008 IEEE International Conference on Software Maintenance*. 157–166. https://doi.org/10.1109/ICSM.2008.4658064

[7] P. Kruchten. 1995. Architectural Blueprints — The "4+1" View Model of Software Architecture. 12 (11 1995), 42–50.

[8] RJ Landis and GG Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[9] RJ Landis and GG Koch. 1981. The measurement of interrater agreement. *Statistics methods for rates and proportions* 2 (1981), 212–236.

[10] R. Mihalcea and P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP 2004*, Dekang Lin and Dekai Wu (Eds.). Association for Computational Linguistics, Barcelona, Spain, 404–411.

[11] A. Shakiba, R. Green, and R. Dyer. 2016. FourD: Do Developers Discuss Design? Revisited. In *Proceedings of the 2Nd International Workshop on Software Analytics (SWAN 2016)*. ACM, New York, NY, USA, 43–46. https://doi.org/10.1145/2989238.2989244

[12] 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

[13] A. Strauss and J. M. Corbin. 1990. *Basics of qualitative research: Grounded theory procedures and techniques.* Sage Publications, Inc.

[14] P. Thongtanunam, R. G. Kula, A. E. Camargo C., N. Yoshida, and H. Iida. 2014. Improving Code Review Effectiveness Through Reviewer Recommendations. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2014)*. ACM, New York, NY, USA, 119–122. https://doi.org/10.1145/2593702.2593705

[15] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K. i. Matsumoto. 2015. Who should review my code? A file location-based code-reviewer recommendation approach for Modern Code Review. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 141–150. https://doi.org/10.1109/SANER.2015.7081824

[16] J. Tsay, L. Dabbish, and J. Herbsleb. 2014. Let's talk about it: evaluating contributions through discussion in GitHub. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*. 144–154.

[17] M. B. Zanjani, H. Kagdi, and C. Bird. 2016. Automatically Recommending Peer Reviewers in Modern Code Review. *IEEE Transactions on Software Engineering* 42, 6 (June 2016), 530–543. https://doi.org/10.1109/TSE.2015.2500238

Table 2: Results of the coding process. Left side of the table contains information on all the 275 paragraphs coded, the right side includes only the 75 paragraphs with the finalized codebook

Category	Occurrences		Category	Occurrences	Description
	Coder #1	Coder#2		Agreed on	
code	105	105	code	11	Implementation issues
maintainability	58	38	maintainability	14	Future plans, OS support, code standards...
planning	10	0			
plan	6	4			
dependencies	7	0			
compatibility	7	0			
usability	0	3			
testing	7	3	testing	1	Tests and testability
robustness	32	36	robustness	13	Robustness, safety, security
safety	0	6			
performance	11	11	performance	2	Performance, runtime optimization
configuration	20	14	configuration	4	Configuration files, flags and options
documentation	23	26	documentation	1	Documentation in-code and off-code
clarification	22	29	clarification	7	Generic question