Pontifícia Universidade Católica
do Rio de Janeiro

Caio Barbosa Vieira da Silva

Exploring the Social Aspects of Design Decay

PUC-Rio - Certificação Digital Nº 1912701/CA

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Alessandro Fabricio Garcia

Rio de Janeiro
April 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Caio Barbosa Vieira da Silva

Exploring the Social Aspects of Design Decay

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee.

Prof. Alessandro Fabricio Garcia
Advisor
Departamento de Informática – PUC-Rio

Prof. Alberto Barbosa Raposo
Departamento de Informática – PUC-Rio

Prof. Marcos Kalinowski
Departamento de Informática – PUC-Rio

Rio de Janeiro, April 8th, 2021

PUC-Rio - Certificação Digital Nº 1912701/CA

All rights reserved.

Caio Barbosa Vieira da Silva

I am an MSc student in Computer Science at Pontifical Catholic University of Rio de Janeiro (PUC-Rio, Brazil), and an Android developer as hobby. I have a bachelor's degree in Computer Science from the Federal University of Alagoas (UFAL). During my graduation, I was always part of research projects. I participated on collaborations with relevant European Universities, such as University of Coimbra, University of Florence, which I visited for three months as part of the collaboration, and University of College London. My current research is focuses on understanding if social aspects that surround the collaborative code development have a relation with design decay. As result of my research, I submitted and got accepted papers in relevant international vehicles, such as The International Conference on Software Maintenance and Evolution (ICSME), International Conference on Software Analysis, Evolution and Reengineering (SANER), and The International Conference on Mining Software Repositories (MSR).

PUC-Rio - Certificação Digital Nº 1912701/CA

Bibliographic data

Barbosa Vieira da Silva, Caio

Exploring the Social Aspects of Design Decay / Caio Barbosa Vieira da Silva; advisor: Alessandro Fabricio Garcia. – Rio de janeiro: PUC-Rio, Departamento de Informática, 2021.

v., 89 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Social Aspects – Teses. 2. Design Decay – Teses. 3. Data Mining – Teses. 4. Aspectos Sociais;. 5. Decaimento de Design;. 6. Mineração de Dados. I. Garcia, Alessandro Fabricio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

First, I would like to thank my family, without their support, I probably would not be here. Second, I would like to thank my girlfriend, that is my best friend and supporter. Third, I would like to thank my research colleagues, that helped me very much in times of need. To my graduation advisor, Prof. Dr. Baldoino Fonseca, that teached me the way into the research world. Finally, but not less important, I would like to thank my advisor, Prof. Dr. Alessandro Garcia, for his trust in my work, advices, lessons and for giving me the opportunity of being his student. I am also grateful to Capes, CNPq, FAPERJ and PUC-Rio for the financial support that made my research possible in the first place. Finally, my sincere thanks to the administrative staff of the DI at PUC-Rio. This study was financed in part by the Coordenação de Aperfeiçoaento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

PUC-Rio - Certificação Digital Nº 1912701/CA

Abstract

Barbosa Vieira da Silva, Caio; Garcia, Alessandro Fabricio. **Exploring the Social Aspects of Design Decay**. Rio de Janeiro, 2021. 89p. Dissertação de mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Code development has been performing collaboratively for a long time. Platforms, such as GitHub, contribute to this process with various mechanisms. Pull Request is a mechanism that allows developers to submit their contributions to a project. Then, these changes can be discussed, analyzed, and reviewed before being integrated into the repository. One of the goals of this process is to avoid a phenomenon called design decay. It occurs when poor design structures are introduced in a project. As a result, the project may become difficult to maintain and evolve. Existing techniques use source code symptoms (e.g., code smells) to identify the manifestation of design decay. Nevertheless, such symptoms can only be used to identify design decay that is already present in the project. Thus, in this dissertation, we investigated the exploration of three social aspects to predict the manifestation of design decay on open-source projects as follows. *Communication Dynamics* represents information about contributor's roles and temporal aspects of their discussions. *Discussion Content* is the information being exchanged among participants of a contribution. Finally, *Organizational Dynamics* represents characteristics of the team organization. The manifestation of these social aspects along software development can induce behaviors that possibly affect the design quality. However, no previous study has investigated the influence of such social aspects on the manifestation of design decay. Thus, we aim to shed light on how these three aspects influence the design decay. To achieve this goal, we introduced a suite of metrics for characterizing social aspects in pull-based software development. Then, we analyzed seven open-source projects, mining both their commits and pull requests. Our results reveal that: (i) many social metrics, e.g., Discussion Length, can be used to discriminate between pull requests that impact on the manifestation of design decay from the ones that do not impact; (ii) various factors of communication dynamics, such as Number of Users, are related to design decay. Nevertheless, temporal factors of communication dynamics outperform the participant roles as indicators of design decay; and (iii) aspects related to organizational dynamics, such as the number of newcomers, are surprisingly not associated with design decay manifestation.

Keywords

Social Aspects; Design Decay; Data Mining

PUC-Rio - Certificação Digital Nº 1912701/CA

Resumo

Barbosa Vieira da Silva, Caio; Garcia, Alessandro Fabricio. **Explorando os Aspectos Sociais do Decaimento de Design**. Rio de Janeiro, 2021. 89p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O desenvolvimento de código vem sendo executado de forma colaborativa há muito tempo. Plataformas, como o GitHub, contribuem para esse processo com vários mecanismos. Pull Request é um deles, e permite aos desenvolvedores enviarem suas contribuições para um repositório, onde essas mudanças podem ser discutidas e revisadas antes de serem integradas ao código principal. Um dos objetivos desse processo é evitar um fenômeno chamado *design decay*, que ocorre quando estruturas de pobres de design são introduzidas no código fonte. Como resultado, o projeto pode se tornar difícil de manter e evoluir. As técnicas existentes usam sintomas de código fonte (e.g., mal cheiros de código) para identificar a manifestação de *design decay*. No entanto, esses sintomas só podem identificar *design decay* que já se ocorreu. Assim, nesta dissertação, investigamos três aspectos sociais para prever a manifestação de design decay em projetos de código aberto. *Dinâmica de comunicação* representa informações sobre os papéis dos contribuidores e aspectos temporais das discussões. *Conteúdo da discussão* é a informação sendo trocada entre participantes de uma contribuição. Finalmente, *dinâmica organizacional* representa as características da equipe. A manifestação desses aspectos sociais ao longo do desenvolvimento de software pode induzir comportamentos que possivelmente afetam a qualidade do código. No entanto, nenhum estudo anterior investigou a sua influência no *design decay*. Assim, buscamos evidências sobre como esses três aspectos influenciam na manifestação de *design decay*. Para atingir esse objetivo, nós introduzimos um conjunto de métricas para caracterizar aspectos sociais num modelo de desenvolvimento baseado em pull requests. Então, nós analisamos sete projetos, extraindo seus commits e pull requests. Nossos resultados revelam que: (i) métricas sociais podem ser usadas para discriminar as pull requests que impactam na manifestação de *design decay* daquelas que não impactam; (ii) vários fatores da dinâmica da comunicação estão relacionados ao *design decay*. No entanto, os fatores temporais superam os fatores dos papéis dos participantes como indicadores de *design decay*; e (iii) aspectos relacionados à dinâmica organizacional, como o número de novatos, surpreendentemente, não estão associados a manifestação de *design decay*.

Palavras-chave

Aspectos Sociais; Decaimento de Design; Mineração de Dados

PUC-Rio - Certificação Digital Nº 1912701/CA

Table of contents

PUC-Rio - Certificação Digital Nº 1912701/CA

PUC-Rio - Certificação Digital Nº 1912701/CA

List of figures

PUC-Rio - Certificação Digital Nº 1912701/CA

List of tables

PUC-Rio - Certificação Digital Nº 1912701/CA

List of Abreviations

DL – Discussion Length
HL – High-Level
LL – Low-Level
MTBC – Mean Time between Comments
NWD – Number of Words in Discussion
NWPCD – Number of Words per Comment in Discussion
PR – Pull Request
RQ – Research Question

PUC-Rio - Certificação Digital Nº 1912701/CA

1
Introduction

Software development does not only consist of technical activities, but also of social activities that emerge from the collaboration among developers [13, 21, 68]. These social activities consist of developers communicating with each other by exchanging their knowledge while aiming at cooperatively producing high-quality software. This is known as the pull-based development model [13]. This model allows developers, who play different positions, to submit *Pull Requests*. Each pull request explicitly describes which changes were made in the source code. Therefore, the code changes can be reviewed, commented, and discussed. For instance, in GitHub, developers can interact through different venues of discussion [63]: (i) discussions on the pull request itself, (ii) discussions on a specific line within the pull request, and (iii) discussions on a specific commit within the pull request. Moreover, these discussions may be related or not to the complexity and impact of the changes submitted.

With the predominance of the use of distributed version control systems, such as git, several software projects start to adopt the pull-based software development model, specially the open source ones [33]. However, the pull request dynamics used to happen elsewhere via external tools (i.e., Bugzilla, Gerrit). Given such increasingly popular dynamics, the code environments are integrating and improving their tools so the developers can communicate better and engage more in discussions.

Social aspects capture communication activities between developers and their interpersonal relations [28]. Many of those social aspects, which are fostered by such platforms widely, influence the quality of produced code. Thus, social aspects are increasingly intrinsic to software development. They can no longer be ignored by one understanding influential factors on software quality and productivity. In fact, prior studies consistently report that discussions on design structure are frequent in this development model [24, 54, 85]. Nonetheless, different social activities in pull-based development may contribute to avoiding, reducing, or accelerating design decay.

Design decay is a phenomenon in which developers progressively introduce code with poor design structures into a system [55]. It is caused by design decisions that negatively impact quality attributes, such as maintainability and

PUC-Rio - Certificação Digital Nº 1912701/CA

extensibility [35, 66, 38]. An example of design decay is when a class is overloaded with multiple unrelated functionalities, making it difficult to use and increasing the chances of causing ripple change effects on other classes. In this context, design decay can be measured through the quantification of source code symptoms – also popular known as code smells. These symptoms are indicators of structural design degradation in the scope of classes, methods, and code blocks [61]. The decay in a certain program unit is characterized by the number of smells as well as the number of different smells types [66, 48]. Moreover, the presence of decay indicates the existence of design problems [66, 83], which may directly hinder one or more non-functional requirements, such as performance and maintainability.

Different social aspects may capture communication activities between developers and their interpersonal relations [28]. They are central to the characterization of social software engineering, which is the application of processes, methods, and tools to enable community-driven creation, management, deployment, and use of software in online environments [32]. Examples of key social aspects are *communication dynamics*, *discussion content*, and *organizational dynamics* [9, 81]. The *communication dynamics* determines how the communication flows among developers, who also play specific roles along the change under development (e.g., number of core developers). The *discussion content* determines characteristics of the contents of comments exchange among developers (e.g., number of snippets in discussion). Finally, the *organizational dynamics* represent the aspects of the team as a whole (e.g., number of newcomers or developers leaving the organization).

The aforementioned social aspects can be directly or indirectly related to design quality for various reasons. First, the *communication dynamics* can reveal the importance of the participant roles, as the temporal aspects of the discussion, and their impact on design decay. Second, the *discussion content* is key when we need to understand what is being discussed, and if some type of content is able to increase or decrease the design decay. Third, the *organizational dynamics* reveals the importance of team size and gender diversity when investigating design decay on collaborative software organizations. Thus, the manifestation of these social aspects along software development can induce behaviors that possibly affect the design quality of the code under development.

PUC-Rio - Certificação Digital Nº 1912701/CA

1.1 Problem Statement and Limitations of Related Work

As aforementioned, developers communicate with each other during software development in collaborative coding environments. They exchange knowledge and ideas about the code being developed to keep or improve the software quality. In the pull-based development model, a developer submits her/his code for review. After the discussions, which are either directly or indirectly related to the code, it may be merged to the main branch of the repository. Such discussions involve multiple social aspects, i.e., *discussion content*, *communication dynamics*, and *organizational dynamics*. Since these aspects are intertwined with design and implementation decisions, they may dictate or influence the future of the code being produced and the overall design of the system. Unfortunately, the relationship between these key social aspects and design decay has not been studied so far. Consequently, developers and software organizations lack evidence to support or monitor certain social aspects that could positively or negatively influence design decay. More specifically, we do not know which social metrics can be used to discriminate between design impactful and unimpactful pull requests. Moreover, we know little about the influence of social aspects on design decay. Thus, we formalize our general research problem as follows.

PUC-Rio - Certificação Digital Nº 1912701/CA

> **General Problem.** Developers and software organizations lack evidence to support or monitor certain social aspects that could positively or negatively influence the design decay.

In order to solve our general problem, we assessed the relationship between social aspects and design decay through a multi-case study [6]. This first study provided evidence that many social metrics can be used to discriminate between a design impactful and unimpactful pull request. The sample analyzed in this study contained 5 projects and assessed two social aspects dimensions: (i) *communication dynamics*, which represent the dynamic of the discussion activity, such as the role of participants involved in a discussion; and (ii) *discussion content*, which represents the interaction of developers during the exchange of messages and obtained information about the content of each message. Moreover, we noticed that different social metrics tend to be indicators of design decay when analyzing both aspects in separate or together. Nevertheless, this study mainly focused on basic activities surrounding the pull request workflow, leaving behind a lot of social information that could be explored. Examples include information on the

organization surrounding the project and deeper analysis on the content of comments.

Thus, in a subsequent study, we performed an in-depth empirical analysis for better understanding: (i) the social aspects of discussion content already approached by our first study [6]; (ii) new dimensions of social aspects; and (iii) design symptoms decrease by validating with Self-Affirmed Refactorings. In order to produce this in-depth analysis, we: (i) increased the number of projects in the dataset; (ii) designed and implemented new social metrics of *discussion content* and *communication dynamics*; (iii) introduced a new dimension of social aspects, i.e., *organizational dynamics*; and (iv) applied rigorous statistical analysis over the methods used in the first study. Similarly, we computed the design decay symptoms using the open-source version of a tool called DesigniteJava [60]. This tool is able to identify 27 decay symptoms: 17 design symptoms, and 10 implementation symptoms. Finally, in order to confirm the cases where a decrease on the design symptoms occurred, we applied the methodology of AlOmar *et al.* [2] to collect Self-Affirmed Refactorings on the commit messages and comments.

PUC-Rio - Certificação Digital Nº 1912701/CA

The Distinction Between Impactful and Unimpactful Pull Requests using Social Metrics. Previous studies [33, 54] tend to consider only product and process metrics of code contributions performed via pull requests that affect the quality of a software system. However, as aforementioned, different social aspects (measured via social metrics) can also directly affect the software quality. Unfortunately, little is known if there is a statistically significant difference between social measures for impactful pull requests and unimpactful ones. In this context, a pull request is *impactful* if there is a variation (increase or decrease) on design decay symptoms when considering all commits (the sum of their symptoms) of this pull request, i.e., the density of symptoms has increased. Conversely, a pull request is *unimpactful* if the design decay symptoms remained the same considering all commits of the pull request, sum of their symptoms is equal to zero. A clear understanding of what social metrics can be used to distinguish impactful and unimpactful pull requests using social metrics can be beneficial to future research in many ways. For instance, we discovered that the metric Mean Time Between Comments is able to differentiate between these two types of pull requests. Then, future works will know which metric they should use in algorithms or machine learning models, without the need to test all metrics. In summary, we formalize our first specific problem as follows.

Problem 1. There is a lack of empirical evidence on which social metrics are able to distinguish between impactful and unimpactful pull requests.

Influence of Social Aspects on Design Decay: Empirical Evidence is Quite Scarce. The relationship between social aspects and design decay has not been studied so far. In fact, most of the previous studies focus on investigating the relations of social aspects with code review [77, 76], post-release defects [9, 26, 18], pull-request acceptance [10, 59] or software vulnerabilities [42]. Despite this vast body of knowledge, as aforementioned, the literature still lacks empirical evidence about the influence of different social aspects on design decay. Hence, it is unclear if key social aspects, such as communication dynamics, discussion contents, and organizational dynamics have a positive or negative relationship with design decay. A clear understanding about the influence of social aspects on design decay can reveal if the combination of these multiple social aspects, or the use of each aspect in isolation, results in a better indication of the design decay.

Problem 2. We know little about the influence of communication dynamics, discussion content, and organizational dynamics on design decay.

1.2 Main Research Contributions

In order to address our research problems (Section 1.1), this Master's dissertation focused on performing two retrospective studies using data from open-source software projects. We retrospectively analyzed data available in the repository. Our studies aimed at improving the knowledge of the social software development communities on how their behavior can be harmful to the design of the code being developed. First, we investigated the interplay between social aspects and design decay in an empirical study involving seven projects. We analyzed the degree of relationship between each social aspect an design decay. To perform this study we designed and implemented a set of social metrics, which are our first contribution. Each social aspect is represented by a specific suite of metrics.

Contribution 1: We have designed and implemented a comprehensive suite of 22 metrics to enable us to observe the influence of social aspects on design decay.

PUC-Rio - Certificação Digital Nº 1912701/CA

Then, we investigated how the social metrics performed with respect to their relation to design decay symptoms in two studies. In our first analysis, we verified if social metrics are able to differentiate design changes in pull requests. Then, we created a statistical model in order to investigate to what extent these metrics are related to design decay. Finally, our findings can be used to improve developers behaviors in social coding communities.

> **Contribution 2:** We reported a set of findings on how social metrics can be used to indicate design decay.

Other Contributions. We also consider relevant three other contributions. First, a framework to collect social data from GitHub repositories. This framework makes it easier the work of researchers, by presenting an interface to the GitHub API v3. Moreover, this framework is also generic, which allows the introduction of new endpoints easily. Finally, others can reuse, extend and tailor this framework to their purposes. As the second contribution, we offer a comprehensive dataset of social data mined from seven open-source projects. Third, we also developed an R script to evaluate the metrics on two statistical models: *Wilcoxon Rank Sum Test* and a *Multiple Logistic Regression*. The former is able to identify which metrics have a correlation with the variable(s) being analyzed. The latter analyzes the influence of each metric in the presence of each other, i.e., analyzes if different metrics are representing the same behavior as indicators of the outcome variable (design decay).

Publications. The contributions this Master's dissertation are reported in various publications that are listed in Table 1.1. At the time of this dissertation, 5 papers have been either published or accepted.

Table 1.1: Publications and Submissions

Title	Conference	Status	Type
Revealing the Social Aspects of Design Decay: A Retrospective Study of Pull Requests	*Brazilian Symposium on Software Engineering 2020*	Published	Master's Research
How Does Modern Code Review Impact Software Design Degradation? An In-depth Empirical Study	*International Conference on Software Maintenance and Evolution 2020*	Published	Collaboration
Refactoring from 9 to 5? What and When Employees and Volunteers Contribute to OSS	*Symposium on Visual Languages and Human-Centric Computing 2020*	Published	Master's Research
On Relating Technical, Social Factors, and the Introduction of Bugs	*International Conference on Software Analysis, Evolution and Reengineering 2020*	Published	Master's Research
Predicting Design Impactful Changes in Modern Code Review	*Mining Software Repositories 2021*	Accepted	Collaboration

1.3 Dissertation Outline

The remainder of this dissertation, which is mainly structured as a compilation of two of our technical papers (one published and one to be submitted), is organized as follows.

PUC-Rio - Certificação Digital Nº 1912701/CA

Chapter 2 presents the background, which introduces: (i) pull request-based development model; (ii) design decay symptoms; and (iii) the social metrics either used or defined in this dissertation. Moreover, we also discuss the related work.

In **Chapter 3**, we present our first study, which is an investigation of the relations between social aspects and design decay symptoms. In this study, we first assess if the social metrics are able to be used to discriminate between impactful and unimpactful pull requests. Moreover, we evaluate if the social aspects are related to design decay symptoms. This study consists of the paper "Revealing the Social Aspects of Design Decay: A Retrospective Study of Pull Requests", which was accepted on the Brazilian Symposium on Software Engineering (SBES) in 2020.

Chapter 4 presents our second study, that replicates the first study by doing an in-depth analysis. Similar to the study in Chapter 3, in this study we apply the same methodology, however, we aim to mitigate three main limitations: (i) number of projects; (ii) high-level approach on the discussion content aspect metrics; (iii) number of social aspects. This study consists of the paper "Social Aspects of Design Decay: A Replication Study", which is going to be submitted in a future conference. Finally, **Chapter 5** summarizes the conclusions of our work, presenting the main contributions, implications, as well the future work.

PUC-Rio - Certificação Digital Nº 1912701/CA

2
Background and Related Work

This chapter contains the background and related work of this dissertation. Section 2.1 clarifies the pull request-based development model; Section 2.2 discusses design decay and its symptoms. We focus on presenting symptoms that will be considered in the context of this dissertation. Section 2.3 presents the social aspects that will be addressed in this work; Finally, Section 2.4 concludes this chapter.

2.1
Pull Request-Based Development Model

The GitHub has many features related to source code on its environment, one of the more known ones are the Pull Requests:

> *Pull requests let you tell others about changes you've pushed to a branch in a repository on GitHub.*[1]

The pull request mechanism starts when a developer submits his code change to a repository. On the submission, they should describe their changes, e.g., new features, bug fixes, improvements. Later on, the changes are examined by code reviewers of the repository. Finally, the code can be merged in a branch of the repository, closed (not accepted) or abandoned (no answer for a long time).

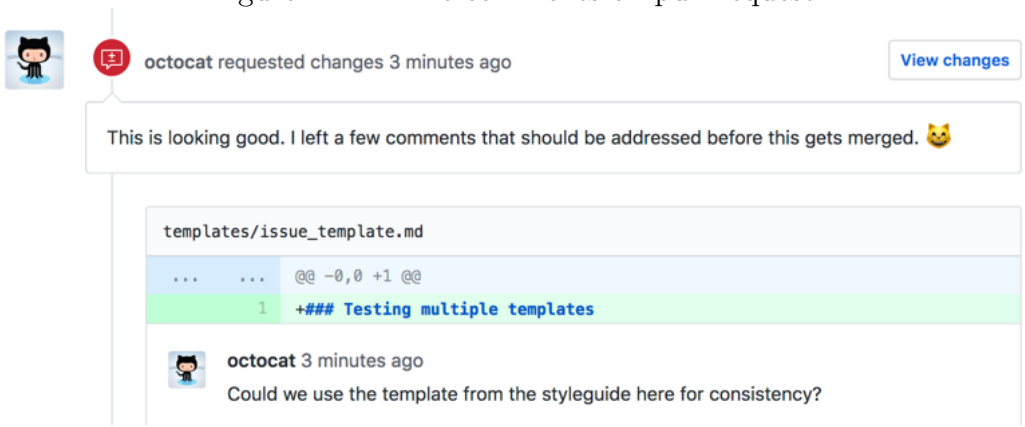
During the pull request opening to the pull request merging (or closing, if rejected), a developer can discuss their changes with more experienced members of the community in two ways: (i) **in line comments** (or **review comments**), this comment happen at the level of a file, as the code reviewers, which usually are experienced members of the repository, make their comments in line, attached to the code snippet that was reviewed and changed; (ii) **conversation comments** (or discussion comments), this type of comment is not related to a commit individually, nor an individual file, but to a conversation surrounding the whole life cycle of a pull request. We can see an example of (i) on Figure 2.1, where the user **octocat** makes an in line comment

[1] https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests

PUC-Rio - Certificação Digital Nº 1912701/CA

"*Could we use the template from the styleguide here for consistency?*" inside a pull request, specifically to the line 1 of the file *templates/issue_template.md*. These types of comments are used for code review. Moreover, on Figure 2.2 we can see an example of (ii), where two users, **tonychacon** and **schacon**, are discussing a change about the time delay of a LED, in this discussion, they do not talk about code, but the main idea behind the change **tonychacon** is going to make. On the remainder of this work we will only address the (ii) discussions, the pull requests *conversations*, as we want to explore the comments that are explicitly related to their collaborative work of the software team. In line comments are often not produced as part of the discussion of two or more developers. Instead, they are very specific comments to document very local changes.

Figure 2.1: In line comments on pull request



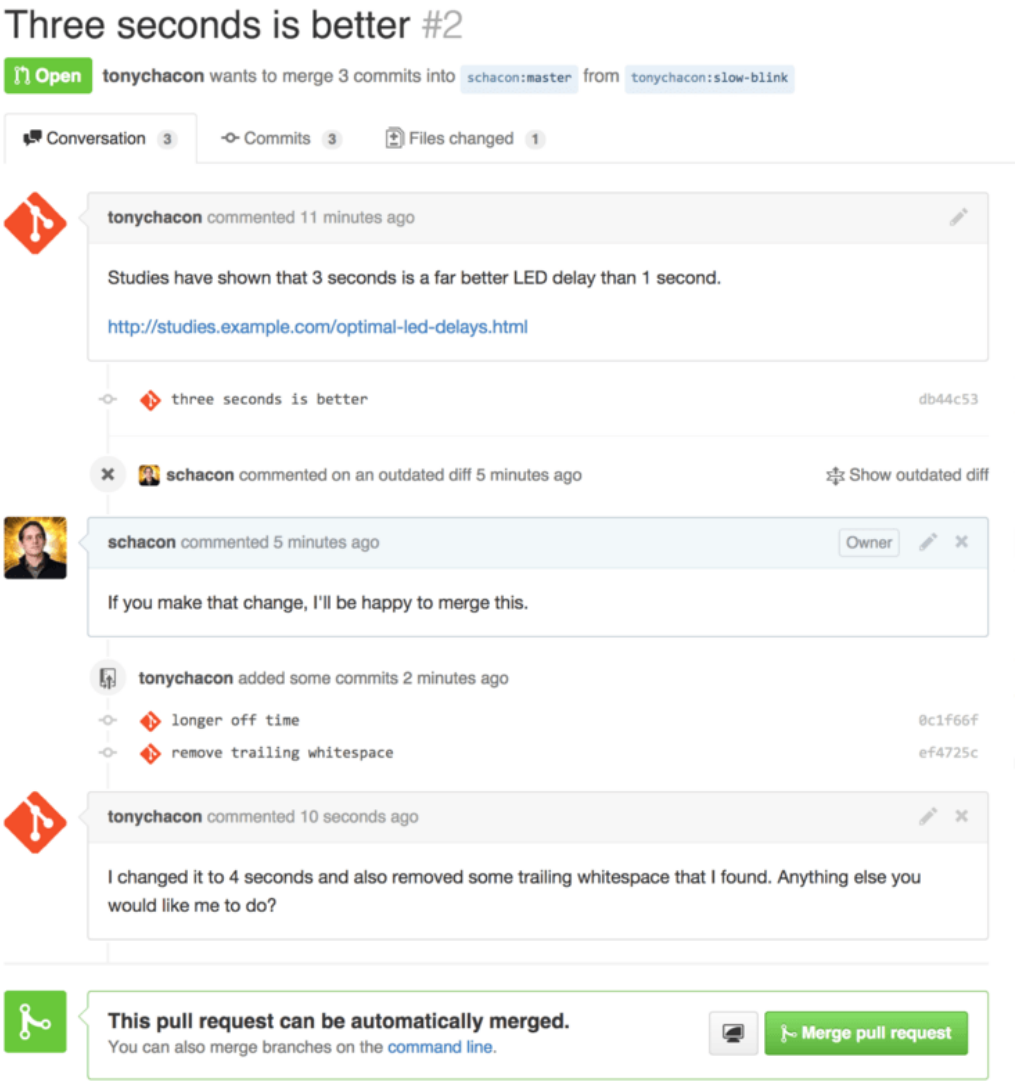
PUC-Rio - Certificação Digital Nº 1912701/CA

2.2 Design Decay and Its Symptoms

The design of a software results from a set of decisions made by the developers along time [71, 72]. However, this design can decay due the introduction of poor code structures, which are named decay symptoms [20, 66, 77], an example of design decay symptom is the long method smell [61]. Code smell is a characteristic of the code that gives insights about deeper problems. Moreover, design decay is caused by the increase on the values of the density or diversity of the design decay symptoms. We calculate the density of design decay symptoms by counting the number of single code smells found in an instance of source code. Conversely, diversity is the number of different types of smells that can be found in an instance of source code.

Several studies [1, 36, 65, 66, 48, 17] analyzed the design decay in different levels of granularity. For instance, Oizumi *et al.* [48] investigated if

Figure 2.2: Conversation on pull request



the symptoms appear in refactored classes with higher diversity and density. They concluded that even in refactoring classes, design level symptoms could indicate design problems. Furthermore, Ahmed *et al.* [1] observed that projects get worse over time in terms of design problems, with the density characteristic as the best indicator. However, in this study, we analyze a wide variety of degradation symptoms when compared to similar studies. We found these works relevant to our study due their findings about design decay symptoms and characteristics of design problems.

In this work, we take into account two categories of design decay symptoms: low-level structural smells (LL) and high-level structural smells (LL) [61]. These two categories were selected since they are able to identify smells of multiple granularities and able to easily validated, for future work. Table 2.1 lists the 27 symptoms types investigated in our study, where the

PUC-Rio - Certificação Digital Nº 1912701/CA

HL smells and LL smells are presented in the upper and bottom halves of the table, respectively. LL smells are indicators of fine-grained structural decay; their scope is generally limited to methods and code blocks [61]. For instance, the long parameter list, empty catch block and long method smells are examples of low-level. HL smells are symptoms that may indicate structural decay impacting on object-oriented principles such as abstraction, encapsulation, modularity, and specialization [37, 61]. An example of HL smells that maybe used for finding design decay is insufficient modularization and unitilized abstraction [61]. This symptom occurs in classes that are large and complex, possibly due to the accumulation of responsibilities. Symptoms of such categories can be automatically detected using a state-of-the-practice tool called DesigniteJava [60].

Table 2.1: Design decay symptoms investigated in this study [61]

Symptom Types and Description
Imperative Abstraction: when an operation is turned into a class
Multifaceted Abstraction: an abstraction that has more than one responsibility assigned to it
Unutilized Abstraction: an abstraction that is left unused
Unnecessary Abstraction: an abstraction that is actually not needed in the system
Deficient Encapsulation: the accessibility of one or more members of an abstraction is more permissive than actually required
Unexploited Encapsulation: when a client class that uses explicit type checks instead of exploiting the variation in types already encapsulated within a hierarchy
Broken Modularization: when data and/or methods that should have been into a single abstraction are spread across multiple abstractions
Insufficient Modularization: when an abstraction that has not been completely decomposed
Hub Like Modularization: when an abstraction has dependencies (both incoming and outgoing) with a large number of other abstractions.
Cyclic Dependent Modularization: when two or more abstractions depend on each other directly or indirectly
Rebellious Hierarchy: when a subtype that rejects the methods provided by its supertype(s)
Wide Hierarchy: when an inheritance hierarchy that is too wide
Deep Hierarchy: when an inheritance hierarchy that is excessively deep
Multipath Hierarchy: when a subtype inherits both directly as well as indirectly from a supertype leading to unnecessary inheritance paths in the hierarchy.
Cyclic Hierarchy: when a supertype in a hierarchy that depends on any of its subtypes
Missing Hierarchy: when a design segment uses conditional logic instead of polymorphism.
Broken Hierarchy: a supertype and its subtype conceptually do not share an “is a“ relationship
Abstract Function Call From Constructor: a constructor that calls an abstract method
Complex Conditional: a conditional statement that is complex
Complex Method: a method that has high cyclomatic complexity
Empty Catch Block: a catch block of an exception that is empty
Long Identifier: an identifier that is excessively long
Long Method: a method that is too long to understand
Long Parameter List: a method that accepts a long list of parameters
Long Statement: a statement that is excessively long
Magic Number: when an unexplained number is used in an expression
Missing Default: a switch statement that does not contain a default case

2.3 Social Aspects

According to Ibrahim *et al.* [28], social aspects aim at capturing interpersonal relations and communication activities between developers. Thus, in this dissertation, we define *social aspects* as dimensions that characterize so-

PUC-Rio - Certificação Digital Nº 1912701/CA

cial properties related to software engineering activities. Previous studies have investigated social aspects in different software engineering activities. Below we discuss the ones that are closely related to this work.

Influence of social aspects during code review. Code review is a software engineering activity directly related to the pull-based development model. Recent studies have investigated the code review activity from different perspectives [42, 59, 41, 77]. Uchôa *et al.* [77], for example, observed that discussions about design may not be enough to avoid design decay and that certain social-related factors, such as long discussions, are responsible for increasing design decay symptoms. Ruangwan *et. al.* [59] investigated how many reviewers did not respond to a review invitation. The authors found that the more reviewers were invited to a patch, the more likely it was to receive a response. Meneely *et. al.* [42] conducted an empirical investigation to understand if the Linus' law applies to security vulnerabilities in the context of code reviews. They concluded that code files reviewed by more reviewers are more likely to be vulnerable.

Social aspects on open-source environments. There are also studies that investigate social aspects in the context of open-source environments [84, 74, 79, 8]. Tsay *et. al.* [74], for instance, found that the social connection between submitters and core members has a strong positive association with pull-requests acceptance. Yu *et. al.* [84] identified patterns of social connections among developers by mining the follow-networks. They also found that the investigated repositories provide transparent work environments for developers, promoting innovation, knowledge sharing, and community building.

Relation of social aspects with quality. Regarding the relation of social aspects with software quality, we found some studies that investigate how social aspects are related with defects [9, 11, 18]. Falcão *et al.* [18] investigated the relationship between social, technical factors, and the introduction of defects. The authors identified that both social and technical factors can discriminate between buggy and clean commits, such as ownership level of developers' commit, and social influence. Bettenburg and Hassan [9] investigated the impact of social interaction measures on post-release defects. They observed that social information can not be used as a substitute to traditional product and process metrics used in defect prediction models. Finally, Bird *et al.* [11] examined the relation between ownership and failures in two large industrial software projects. Their results show that the number of developers has a strong positive correlation with failures.

As presented above, to the extent of our knowledge, no previous study

PUC-Rio - Certificação Digital Nº 1912701/CA

investigated whether social aspects can be used in pull-based development for predicting the manifestation of design decay. In fact, pull requests conversations may contain different social aspects that capture communication dynamics, content and the impact of interpersonal relations [81]. Moreover, these social aspects can be categorized on many social dimensions [81, 9]. Wiese *et al.* [81] did a mapping study, compiling all studies that investigated *social metrics* in software engineering. Based on this study, we selected and grouped metrics into three social aspects, namely *communication dynamics*, *discussion content*, and *organizational dynamics*. The selected metrics can be collected in the context of pull requests to assess the relation of social aspects with design decay symptoms.

We consider that our selected aspects are representative because they are able to capture information about organizations, developers, and their communications. **Communication Dynamics** represents the role of participants involved in a discussion or temporal aspects of the messages. For instance, a discussion that involves developers with different roles (e.g., core developers, organization members, contributors, or newcomers). **Discussion Content** represents the interaction of developers during the exchange of messages and the content of each message. Examples of discussion content elements include the presence of code snippets along the interaction, and the number of words per comment. Finally, **Organizational Dynamics** represents the aspects of the team as a whole. For example, number of newcomers or developers entering or leaving the organization. In the next subsections we present more details about the social aspect and their corresponding metrics.

PUC-Rio - Certificação Digital Nº 1912701/CA

2.3.1 Communication Dynamics

In this section we present the *communication dynamics* dimensions, namely *participation roles*, *size*, and *time*. For each dimension, we present their set of metrics, that are related to the communication between participants of pull requests. Moreover, we are able to measure the characteristics of contributors, the time spawn of the discussions and their size. Next, we present details about each metric, which is grouped with other metrics of the same dimension, as follows.

2.3.1.1 Participation Roles

On the GitHub API [2], there are different types of association tags that an user can have: (i) NONE; (ii) FIRST_TIMER; (iii) FIRST_TIME_CONTRIBUTOR; (iv) CONTRIBUTOR; (v) COLLABORATOR; (vi) MEMBER; and (vii) OWNER. Based on those tags, we defined three different categories of developers, namely: *users*; *contributors*; and *core developers*, to compose the participation roles dimension. We define them as follows.

Number of Users measures the number of unique *users* that somehow interacted in a discussion along a pull request, which either opened, commented, merged, or closed. This role represents the common users, labeled (i) NONE, (ii) FIRST_TIMER, and (iii) FIRST_TIME_CONTRIBUTOR. We believe that these three tags allow us to identify pull requests discussions with the presence of users without expertise or with just a quite limited experience on the source code. Our expectation for this metric is that the number of users participating in a discussion may influence the design quality. A higher number of users may help to improve design quality or it may contribute to the incidence of design decay.

Number of Contributors measures the number of contributors that interacted in some form in the pull request (opened, commented, merged or closed). This role is represented by the developers with the tag association (iv) CONTRIBUTOR and (v) COLLABORATOR. These tags represent developers that commited more than once on the repository or have commit permission in the repository, respectively. The rationale behind this metric is that a higher number of contributors may help to reduce design decay.

Number of Core Developers measures the number of core developers that interacted in some form and with any other kind of user in a pull request (opened, commented, merged or closed). This role is represented by the developers with the tag association (vi) MEMBER and (vii) OWNER. These tags represent the developers that are part of the project organization or are owners of the repository. This metric allows us to identify pull request discussions with the presence of these core developers. Similarly to the previous metric, we expect that a higher number of core developers involved in the discussion of a pull request may result in higher design quality. The rationale is that core developers are very often aware of the design decisions made so far in the project. Thus, the resulting source code change would likely have a better design quality.

[2]https://docs.github.com/en/rest

PUC-Rio - Certificação Digital Nº 1912701/CA

Pull Request Opened By measures the type of user that has opened each pull request or issue associated with the pull request. The user might be an *Employee* or *Temporary* one. Employees are active contributors and core developers. Conversely, temporary users are contributors that do not actively work on the project or does not work for the software organization that maintains the open source project. We believe that issues or pull requests opened by temporaries have more risk of degrading the code. Conversely, issues or pull requests opened by employees are expected to result in a better design quality.

2.3.1.2 Size

Number of Comments quantifies the number of comments that were made within a pull request. The rationale is that discussions with a higher number of comments around a code change would find possible design problems. As a result, the design quality is expected to be improved or, at least, maintained.

2.3.1.3 Time

Mean Time Between Comments represents the sum of the time between all comments of a pull request weighted by the number of comments. A higher time between comments (e.g., a long pause in an otherwise fast-paced discussion) may be related to design decay.

Discussion Duration, which is also called *Discussion Length*, measures the time in days that a pull request lasted. That is, Discussion Duration represents the number of days elapsed between the creation and merging (or closing, if not merged) of a pull request. We believe that the longer is the discussion, the higher will be the chance of problems being explained and solved. Thus, a longer discussion is likely to help to avoid design decay.

Time Between Creation and First Comment measures the time in days between the pull request creation and the first comment on that pull request. We conjecture that the longer the time between the pull request opening and the first comment, the higher the chance of the developer not really engage on solving possible problems, leading to design decay.

Time Between Last Comment and Merge measures the time in days between the last comment on the pull request and the pull request merge (or close, if not merged). In this case, we also believe that the longer the time between the last comment and the closing of the pull request, the higher the

PUC-Rio - Certificação Digital Nº 1912701/CA

chance of the author not engaging on new minor changes. As a result, such a pull request would have more chance of introducing design degradation.

2.3.2 Discussion Content

In this section, we present the metrics related to the *discussion content* aspect. With this social aspect, we expect to measure characteristics related to the content of discussions. This aspect comprehends two dimensions: content size and keyword related. Next, we describe each metric that composes the discussion content aspect.

2.3.2.1 Content Size

Number of Snippets in Discussion measures the number of snippets within each comment of a pull request. Those snippets are detected by the number of ‘‘‘ (three backticks, which is the syntax that opens a snippet in markdown) divided by two (opening and closing). The higher the number of snippets in a discussion, the clearer the users are trying to explain their thoughts and intents. Therefore, avoiding confusion and possibly reducing design decay.

Snippet Size represents the sum of the size of all snippets found on comments in a pull request. The bigger the size of snippets in a discussion, the clearer the users are trying to pass a message to other developers. Therefore, by avoiding misunderstanding, we believe that design decay may also be avoided.

Mean Snippet Size measures the snippet size weighted by the total number of snippets. The higher the mean of snippets in a discussion, the clearer the users are sharing their decisions. Thus, we expect that this would help to avoid confusion and design decay.

Number of Words in Discussion measures the sum of words in all comments of a pull request. For calculating this metric, we applied a preprocessing in the text of comments for removing contractions, stop words, punctuation, and replacing numbers. Discussions with a high number of words are related to more complex changes. Thus, we believe that such changes may be more susceptible to design decay.

Number of Words per Comment in Discussion represents the sum of words in all comments weighted by the number of comments in a pull request. In this case, we also applied the preprocessing for removing contractions, stop words, punctuation, and for replacing numbers. Similarly to the previous

PUC-Rio - Certificação Digital Nº 1912701/CA

metric, we also expect that discussions with a high weighted number of words are related to more complex changes, which may induce design decay.

2.3.2.2 Keyword Related

Number of Design Keywords represents the number of times a design-related keyword is found in the title or comments of a pull request. Changes with design keywords may show that developers were concerned about design, which may indicate that they tried to avoid design decay. The keywords used in this analysis are: *design*, *architect*, *dependenc*, *requir*, *interface*, *servic*, *artifact*, *document*, *behavior*, and *modul*.

Number of Refactoring Keywords measures the number of refactoring-related keywords, which identifies the pull requests that a refactoring keyword in their title or comments. Changes with refactoring keywords may show that developers were concerned about improving the source code design. Therefore, this metric may be associated with a better design quality. The keywords used in this analysis are: *refactor*, *mov*, *split*, *fix*, *introduc*, *decompos*, *reorganiz*, *extract*, *merg*, *renam*, *chang*, *restructur*, *reformat*, *extend*, *remov*, *replac*, *rewrit*, *simplif*, *creat*, *improv*, *add*, *modif*, *enhanc*, *rework*, *inlin*, *redesign*, *cleanup*, *reduc*, and *encapsulat*.

Density of Design Comments measures the mean of the number of design keywords per comments. Our rationale with this metric is that, the higher the mean of design related comments, the higher is the concern with design quality. Thus, as a result, we expect the chances of design decay to be reduced.

Density of Refactoring Comments represents the measurement of the mean number of refactoring keywords occurrences per comment. As in the previous metric, the higher the mean of refactoring comments, the smaller the chances of design decay.

2.3.3 Organizational Dynamics

In this last aspect, we group the metrics related to the characteristics and aspects of the organization. Such characteristics and aspects include information about the diversity, size increase, and reduction of contributors involved in the project. Below we present details about each metric of this dimension.

Team Size measures the number of active developers in the past 90 days. A higher amount of active developers may result in more engagement

PUC-Rio - Certificação Digital Nº 1912701/CA

for the (design) discussions of each pull request. Moreover, the workload of the current developers in the team is reduced. Thus, we expect that the higher the team size, the smaller will be the incidence of design decay. For this metric, we collected all developers that interacted on issues or pull requests in a period of 90 days. However, this developer should have committed at least one time in the repository.

Gender Diversity represents the proportion of male/female contributors on the project's team. We believe that, the higher the gender diversity on a team, the better will be the team performance. Thus, with a better performance, we believe that design decay would be avoided. We used the framework proposed by Vasilescu *et al.* [78] to find the gender of the users, based on their name and location.

Number of Newcomers measures the number of new contributors on the past 90 days. A higher amount of newcomers means that many contributors are not experienced in the project. Therefore, we believe that, the pull requests involving newcomers are more likely to result in design decay. For this metric, we collected all developers that interacted on issues and pull requests in a period of 90 days, but not before this period. However, this developer should have committed at least one time in the repository.

Number of Developers Leaving measures the number of developers that previously contributed to the project but did not contributed on the past 90 days. We conjecture that more developers leaving a project can decrease the engagement of the community, resulting in design decay. For this metric, we collected all developers that interacted on issues or pull requests before a period of 90 days, but do not interacted in the last 90 days. However, this developer should have committed at least one time in the repository.

PUC-Rio - Certificação Digital Nº 1912701/CA

2.4 Summary

This chapter provided the background to support the understanding of this dissertation. We presented basic concepts, used throughout the next dissertation chapters. We also discussed related work reporting studies on social aspects and design decay. The next two chapters present the empirical studies that we conducted for addressing the problems listed on Section 1.1. For this purpose, we analyzed the impact of social aspects on the design decay, to understand their influence and their place as indicators of increase or decrease of design decay symptoms. The implementation of all metrics described in this chapter can be obtained in our replication package [7].

3
Revealing the Social Aspects of Design Decay

One of the main goals of the pull-based development model is to improve the changes under development. Among the objectives of the pull-based model, there is the reduction of design decay occurrence. In this context, two central social aspects may contribute to combating or adversely amplifying design decay. First, design decay may be avoided, reduced or accelerated depending whether the communication dynamics among developers – who play specific roles – is fluent and consistent along a change. Second, the discussion content itself may be decisive to either improve or deteriorate the structural design of a system. Finally, since we are preliminary assessing the social aspects, the organizational dynamics aspect, that contains metrics that are not directly related to a single pull request, we will only be address it on the study 2 (see Chapter 4).

As we discussed in Chapter 2, there is no previous study about the part played by social aspects on either avoiding or amplifying design decay. Previous studies only investigates technical aspects of design decay or confirms the high frequency of design discussions in pull-based software development. Thus, in this chapter we present the paper *"Revealing the Social Aspects of Design Decay A Retrospective Study of Pull Requests"* [6], which was published and presented at the *XXXIV Brazilian Symposium on Software Engineering (SBES)*.

This paper reports a retrospective study aimed at understanding the role of communication dynamics and discussion content on design decay. We focused our analysis on 11 social metrics (see Chapter 2, Section 2.3) related to these two aspects as well as 4 control technical metrics typically used as indicators of design decay (see Chapter 2, Section 2.2. We analyzed more than 11k pull request discussions mined from five large open source software systems.

Our findings reveal that many social metrics can be used to discriminate between design impactful and unimpactful pull requests. This finding motivates us to further investigate the social aspects in this matter. Second, many dimensions of communication dynamics are particularly related to design decay. However, the temporal dimension of communication dynamics outperformed the participant roles' dimension as indicators of design decay. Finally, we noticed certain social metrics tend to be indicators of design decay when

PUC-Rio - Certificação Digital Nº 1912701/CA

analyzing both aspects are considered together. Such results contribute for addressing research problems 1 and 2 (see Section 1.1) of this dissertation.

Revealing the Social Aspects of Design Decay

A Retrospective Study of Pull Requests

Caio Barbosa, Anderson Uchôa, Daniel Coutinho
PUC-Rio, Rio de Janeiro, Brazil
{csilva,auchoa,dcoutinho}@inf.puc-rio.br

Filipe Falcão, Hyago Brito, Guilherme Amaral
UFAL, Alagoas, Brazil
{filipebatista,hpb,gvma}@ic.ufal.br

Vinicius Soares, Alessandro Garcia
PUC-Rio, Rio de Janeiro, Brazil
{vsoares,afgarcia}@inf.puc-rio.br

Baldoino Fonseca, Marcio Ribeiro
UFAL, Alagoas, Brazil
{baldoino,marcio}@ic.ufal.br

Leonardo Sousa
Carnegie Mellon University, California, USA
leo.sousa@sv.cmu.edu

3.1 Introduction

Open-source environments, such as GitHub, promote social coding activities. These social activities consist of developers continually communicating and sharing their knowledge along a code change until it is submitted as a pull request [22, 23]. This is known as the pull-based development model [13]. This model allows developers, who play different roles, to submit, review, comment and discuss code contributions to a software project [63]. The pull-based development model is widely used by open-source communities.

One of the goals of pull-based development model is to encourage the improvement of the change under development, including its beneficial impact on the design structure. In fact, recent studies consistently report discussions about design structure are frequent in this development model [12, 85, 49]. However, social activities in pull-based development may contribute to avoiding, reducing or accelerating design decay. Design decay is a phenomenon in which developers progressively introduce code with poor design structures into a system [20, 66].

Two social aspects are central to open-source coding environments: *communication dynamics* and *discussion content* (or *communication content*) [81, 9]. The former determines how the communication flows among developers, who also play specific roles along the change under development. The latter determines characteristics of the contents of comments exchange among developers. These two social aspects may be related to design quality for various reasons, such as two examples described as follows. First, design decay may be avoided, reduced or accelerated depending whether the communication dynamics of developers with specific roles is fluent and consistent along a change.

PUC-Rio - Certificação Digital Nº 1912701/CA

Second, the characteristics of content of comments can determine the quality of the discussion, and therefore be decisive to either improve or deteriorate the structural design of a system.

Unfortunately, the relationship between these key social aspects and design decay has not been studied so far. Prior works either investigate technical aspects of design decay [17, 47, 27] or confirms the high frequency of design discussions in pull-based software development [75, 57]. Studies of social aspects focus on investigating their relations with post-release defects [9, 26, 18], pull request acceptance [10, 59], and software vulnerabilities [42]. Despite this vast body of knowledge, no study has performed a retrospective investigation on the relationship between key social aspects in pull-based development and design decay. Hence, it is unclear if social aspects, such as communication dynamics and discussion contents, have a positive or negative relationship with design decay.

This paper reports a retrospective study aimed at understanding the role of communication dynamics and discussion content on design decay. We focused our analysis on 11 social metrics related to these two aspects as well as 4 control technical metrics typically used as indicators of design decay. We analyzed more than 11k pull request discussions mined from five large open-source systems. Our findings reveal that many social metrics can be used to discriminate between design impactful and unimpactful pull requests. Second, various factors of communication dynamics are related to design decay. However, temporal factors of communication dynamics outperformed the participant roles' factors as indicators of design decay. Finally, we noticed certain social metrics tend to be indicators of design decay when analyzing both aspects together.

Section 3.2 provides background information and related work. Section 3.3 presents our motivating example. Section 3.4 describes our study settings. Section 3.5 presents the study results. Section 3.6 discusses threats to validity. Finally, Section 3.7 concludes the paper and suggests future work.

3.2 Background and Related Work

3.2.1 Pull Request Discussion and Social Aspects

Pull requests contributions are increasing on open-source environments, such as GitHub. Many open-source projects have guidelines to ensure the use of pull requests for this task [22, 23]. The pull request mechanism is basically

PUC-Rio - Certificação Digital Nº 1912701/CA

started by a developer who submits his code contributions to a repository describing his changes, e.g., new features, bug fixes, improvements. Next, the changes are scrutinized by code reviewers of the organization until they are merged or abandoned. In parallel, both contributors and reviewers can interact through different discussions [63]: (i) discussions on the pull request itself, (ii) discussions on a specific line within the pull request, and (iii) discussions on a specific commit within the pull request. Moreover, these discussions may be related or not to the complexity and impact of the changes submitted.

These discussions are not trivial since it may involve different social aspects that capture communication activity among developers and measures the impact of interpersonal relations [81]. Examples of social aspects are *communication dynamics* (or discussion dynamics) and *discussion content*. Communication dynamics represent the role of participants involved in a discussion or temporal aspects of the messages. For instance, a discussion that involves developers with different roles (e.g., members, contributors, or newcomers). On the other hand, the discussion content represents the interaction of developers during the exchange of messages and obtained information about the content of each message. For instance, the presence of code snippets, and the number of words per comment.

PUC-Rio - Certificação Digital Nº 1912701/CA

3.2.2 Design Decay and its Symptoms

Software design results from a series of decisions made during software development [71, 72]. However, along with software development, a software design may decay due to the progressive introduction of poor structures into the system, i.e., decay symptoms [20, 66, 77]. Such design decay is caused by a successive increase in the density of symptoms along with software evolution.

Aimed at minimizing and removing decay symptoms, developers need to identify and to refactor source code locations impacted by design decay. Previous studies [66, 46, 83] have identified five categories of symptoms upon which developers often rely to identify structural decay. Such studies observed that developers tend to combine multiples decay symptoms by considering dimensions such as density, and diversity to determine if there is code decay.

In this work, we focus on two categories of symptoms: low-level and high-level structural smells [61]. *Low-level structural smells* are indicators of fine-grained structural decay; their scope is generally limited to methods and code blocks [61]. For instance, the Long Method smell. *High-level structural smells* are symptoms that may indicate structural decay impacting on object-oriented characteristics such as abstraction, encapsulation, modularity, and

hierarchy [37, 61]. An example of high-level structural smells that may be used for finding design decay is Insufficient Modularization [61]. This symptom occurs in classes that are large and complex, possibly due to the accumulation of responsibilities. Symptoms of such categories can be automatically detected using a state-of-the-practice tool called DesigniteJava [60].

3.2.3 Related Work

Social aspects in code review. There are multiple studies about the influence of social aspects during code review [42, 59, 41, 77]. For instance, Ruangwan *et. al.* [59] investigated how many reviewers did not respond to a review invitation. The authors found that the more reviewers were invited to a patch, the more likely it was to receive a response. Nevertheless, Meneely *et. al.* [42] attempted to investigate Linus' law to identify if it applies to security vulnerabilities empirically. In this study, the authors concluded that code files reviewed by more reviewers are more likely to be vulnerable. While those works focus on analysing aspects surrounding code review tasks, our study aims at investigating the social aspects occurring in discussions that are not directly related to source-code.

Social aspects in open-source environments. Previous studies [84, 74] have investigated different social aspects in open-source environments, such as GitHub. Yu *et. al.* [84] not only identified social patterns among developers by mining the follow-networks, but also found that those repositories provide transparent work environments for developers, promoting innovation, knowledge sharing, and community building. Additionally, Tsay *et. al.* [74] found that the social connection between submitters and core members has a strong positive association with pull-requests acceptance. Previous work focused on open-source environments and their relations. In order to complement these studies, we will use some of the social aspects observed, such as communication dynamics, to assess their influence on design decay.

Effect of social aspects on code quality. Bettenburg and Hassan [9] investigated the impact of social interaction measures on post-release defects. The authors observed that social information can not be used as a substitute to traditional product and process metrics used in defect prediction models. Bird *et al.* [11] examined the relationship between ownership and failures in two large industrial software projects. They found that the number of developers has a strong positive relationship with failures. Falcão *et al.* [18] investigated the relationship between social, technical factors, and the introduction of bugs. The authors identified that both social and technical factors can discriminate

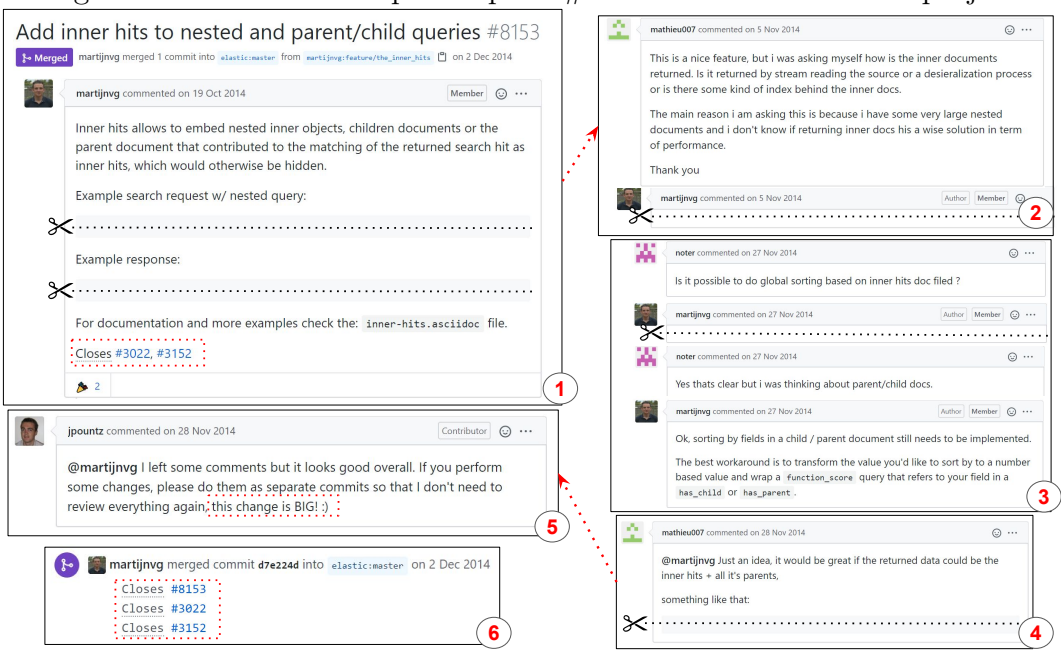
PUC-Rio - Certificação Digital Nº 1912701/CA

between buggy and clean commits, such as ownership level of developers' commit, and social influence. These works use social and technical metrics to understand the influence over the presence of code defects. Our work differs from these studies, by analyzing if and what extent metrics related to two social aspects are good indicators of the design decay.

Our work differs from the existing ones in several ways: (i) we analyzed social aspects not only related to code review tasks; (ii) while most studies are focused on analyzing the influence of social aspects on software defects and vulnerabilities, we investigated the influence of social aspects on design decay; and (iii) we used a multiple logistic regression model technique to evaluate which social aspects indicate design decay separately and together.

3.3 Motivating Example

Figure 3.1: Discussion in pull request #8153 from Elasticsearch project



This section presents a real example in which design decay was introduced in the system after merging a pull request. We aim to highlight possible aspects surrounding the discussion that could have possibly indicated that the changes would influence a design decay. For this purpose, we adopt merged pull request #8153 from the Elasticsearch project, to motivate our study. Figure 3.1 illustrates this pull request in two parts that represent two periods: before and after the merge. We discussed our example step-by-step as follows.

This pull request was titled "*Add inner hits to nested and parent/child queries*" and its main goal was the addition of a new feature. As seen in step

PUC-Rio - Certificação Digital Nº 1912701/CA

①, the discussion starts with the author reporting that his pull request is related to Issues #3022 and #3152. This discussion evidence the existence of multiple concerns being addressed in the same pull request. Moreover, after a few rounds of review by two different reviewers, this pull request was also mentioned on another Issue (# 761). Further in the discussion (step ②), we can observe that another participant, an user, joins the discussion by asking the author for information about the performance of the feature being developed, in his use case. The author promptly answers the user and returns to discussing his code changes.

Perhaps due to the precedent set, another participant, also an user, joins the discussion (step ③) and also asks something about the feature, and again, the author answers it; but this time, reaches the conclusion that the question relates to another unreleased feature. After this, the user who asked the first question comments again (step ④) giving suggestions about how the feature should be implemented, which this time are never answered by the author.

A few rounds of review later, and the Pull Request is merged. The changes contained in this pull request ended up being large, with one reviewer later saying "*If you perform some changes, please do them as separate commits so that I don't need to review everything again, this change is BIG!*" (shown in step ⑤). When the change was merged, a new issue (#8153) was also associated to the pull request, further indicating that new concerns were added during the development of the code changes (step ⑥).

After the Pull Request was merged, four more users commented asking questions related to the feature being introduced: "*can I bump our use case against this to see if I am understanding this feature correctly?*", "*Has this work be slated for a particular release yet?*", and so on. The discussion in this pull request continued for seven months after the merge, containing multiple questions and suggestions for possible improvements.

In this example, we can observe many social aspects surrounding the discussion, such as: (i) different participant roles (core developers of the organization, contributors and users); (ii) temporal aspects (the time span of the pull request); and, (iii) size and content of comments (snippets being used). One could argue that, some of those social aspects could indicate or even be responsible for changes that lead to introduction of the design decay symptoms into the system. For instance, the presence of users (i.e. participants that never committed on the repository) commenting and raising extra concerns to a pull request, could have raised the complexity of the changes. As a consequence, an increase in the design decay symptoms could have happened.

This example shows a scenario where several confounding aspects created

PUC-Rio - Certificação Digital Nº 1912701/CA

a situation where design decay symptoms were unknowingly increased in the system. By analyzing some of those aspects, such as process metrics relating to the changes and metrics about social aspects related to the discussion, we hope to improve our understanding on some of the situations and behaviors that guide those discussions and the development contained within them, and by doing that, we hope to also improve our knowledge about how those aspects can influence design decay.

3.4 Study Settings

3.4.1 Goal and Research Questions

We relied on the Goal Question Metric template [82] to describe our study goal as follows: *analyze* social aspects; *for the purpose of* understanding their impact on design decay; *concerning* changes in structural design quality; *from the viewpoint of* software developers when performing code changes; *in the context of* five open-source systems. We introduce each research question (RQs) as follows.

$\mathbf{RQ}_1$: *Are social metrics related to design decay?* – $\mathbf{RQ}_1$ aims at investigating if there is a statistically significant difference between social measures for impactful pull requests and unimpactful ones. We consider that a merged pull request is impactful when an *increase* or *decrease* in design decay was observed as a result of merging the pull request changes. Conversely, unimpactful pull requests are merged pull requests that do not affect on the design decay. Thus, by answering $\mathbf{RQ}_1$, we will be able to understand which social metrics are more related or not with impactful pull requests.

$\mathbf{RQ}_2$: *To what extent the communication dynamics influence the design decay?* – During software development, the influence of different social aspects may contribute to design decay. Thus, differently from the previous research question, $\mathbf{RQ}_2$ aims at investigating to what extent multiple social metrics related to the communication dynamics aspect influence design decay. Thus by answering $\mathbf{RQ}_2$ we can evidence whether each social metric, by considering the presence of others, can be used as indicators of design decay.

$\mathbf{RQ}_3$: *To what extent the discussion content influence the design decay?* – Similar to RQ_2, we investigate as multiples social metrics related to the discussion content aspect influence design decay. By answering $\mathbf{RQ}_3$, we also can compare which social aspects, i.e., communication dynamics and discussion content when analyzed in isolation, are sufficient or not to indicate a design

PUC-Rio - Certificação Digital Nº 1912701/CA

decay. Furthermore, we can reveal if the combination of these multiple social aspects results in a better indication of the design decay. Our goal is to provide a set of metrics that can be used, in the context of social aspects on discussions inside pull requests, to indicate the increase or the decrease of design decay symptoms. By doing this, we can shed light on future work on social aspects and design decay.

3.4.2 Study Steps and Procedures

Step 1: Selecting open-source systems. From GitHub, we selected five open-source Java projects that widely adopt pull request-based development. We selected only open-source projects to allow study replication. To select them, we followed criteria based on related studies [74, 18]. We selected systems that matched with the following criteria: (i) systems that use pull request reviews as a mechanism to receive and evaluate code contributions; (ii) systems that have at least 1k commits and pull requests; (iii) systems that are at least 5 years old and are currently active. Moreover, we selected this criteria to avoid known mining perils [31]. Finally, we focused on Java systems due to constraints of the DesigniteJava tool [60] (see Step 2). Table 3.1 provides details about each selected system. The first column shows the names of each selected system and the remaining columns present: system's domain; number of commits; number of pull-requests; and period considered in this study.

Table 3.1: Software systems investigated in this study

System	Domain	# Commits	# Pull-requests	Time span	LOC
Elasticsearch	Search Engine	17,251	4598	2011-2018	734,514
Presto	Query Engine	1,958	1,542	2012-2019	635,760
Netty	Framework	4,071	147	2011-2019	279,572
OkHttp	HTTP client	9.690	4,013	2012-2019	36,686
RxJava	Library	4,140	1,299	2013-2016	103,609

Step 2: Detecting multiple design decay symptoms. We used the DesigniteJava tool [60] to detect a total of 27 decay symptoms types: 17 high-level structural smells, and 10 low-level structural smells. Hence, for each system, we identified these decay symptoms by considering each pull request that has been submitted and merged during the project history. For each merged pull request, we have downloaded a snapshot of each commit related to this pull request and its parent. Then, we accessed the difference between them, by following this methodology, we are guaranteeing that the introduced design decay symptoms were solely introduced by the code change in the pull request, this way we avoid the Rebase effect [51, 52]. This is due to such a pull request being the only potential point in time in which the code could be changed. Table 3.2 lists the 27 symptoms types investigated

PUC-Rio - Certificação Digital Nº 1912701/CA

in our study, where the high-level structural smells and low-level structural smells are presented in the upper and bottom halves of the table, respectively. The descriptions, detection strategies, and thresholds for each symptom are available in our replication package [7].

Table 3.2: Degradation symptoms investigated in this study

High-level symptoms
Imperative Abstraction, Multifaceted Abstraction, Unutilized Abstraction, Unnecessary Abstraction, Deficient Encapsulation, Unexploited Encapsulation, Broken Modularization, Insufficient Modularization, Hub Like Modularization, Cyclic Dependent Modularization, Rebellious Hierarchy, Wide Hierarchy, Deep Hierarchy, Multipath Hierarchy, Cyclic Hierarchy, Missing Hierarchy, Broken Hierarchy [61].
Low-level symptoms
Abstract Function Call From Constructor, Complex Conditional, Complex Method, Empty Catch Block, Long Identifier, Long Method, Long Parameter List, Long Statement, Magic Number, Missing Default [61].

Step 3: Computing design decay indicators in terms of density symptoms. Based on previous studies [66, 48, 50, 77, 76, 86], we have selected the density of symptoms as indicators of design decay[1]. For this purpose, for each target system, we computed the difference of the indicator for each decay symptom, i.e., high-level and low-level structural smells, by considering all merged pull requests collected. We computed the density as a sum of the aggregate value of the number of instances of symptoms types in each smelliness file for each version of the system before and after the merged pull request. In summary, a positive difference in the density of symptoms indicates an *increase* in the design decay as a result of the merged pull request, therefore, there is a worsening on the design. Similarly, a negative difference in density of symptoms indicates a *decrease* of the design decay as a result of the merged pull request. Finally, a difference equal to zero in the density of symptoms indicates that there has been no structural design change. In total, we have computed the four indicators for 11,599 merged pull requests. We provide all computed indicators in our replication package [7].

Step 4: Calculating control metrics and social aspects. Table 3.3 shows the 15 metrics that we have used to measure certain social aspects occurring parallel to the code development. The first part of Table 3.3 describes the control variables that we computed to avoid some factors that may affect our outcome if not adequately controlled. As control variables, we used *product* and *process* metrics, which have been shown by previous research to be correlated with design decay [54, 33]. The second part of Table 3.3 describes the metrics that we considered as independent variables to measure certain social aspects. We have grouped each metric in two categories, each one

[1]We also compute diversity as an indicator. However, we did not observe any difference in the results of density and diversity. Thus, we decided to use only density.

PUC-Rio - Certificação Digital Nº 1912701/CA

representing a social aspect. *Communication dynamics* represent the dynamic of the discussion activity, such as the role of participants involved in a discussion or temporal aspects of the messages. Finally, *discussion content* represents the interaction of developers during the exchange of messages and obtained information about the content of each message. For instance, the number of snippets written in a discussion. We emphasize that these metrics are extensively used by previous works as reported in [81] to measure the social aspects. Moreover, all two categories investigated in our study suggest social aspects that may be favorable or not the structural design change.

PUC-Rio - Certificação Digital Nº 1912701/CA

PUC-Rio - Certificação Digital Nº 1912701/CA

Table 3.3: Control and independent variables used in our study.

Type	Metrics	Description	Rationale
		Control variables	
Product	Patch Size	Number of files being subject of review.	Large patches can be more prone to be analyzed for how the involved classes are designed.
	Diff Size	Difference of the sum of the Lines of Code metric computed on the version before and the version after the review of all classes being subject of review	Large classes are hard to maintain and can be more prone to be refactoring [53]
	Diff Complexity	Difference of the sum of the Weighted Method per Class metric computed on the version before and after review of all classes being subject of review.	Classes with high complexity are potential candidates to be refactored
Process	Patch Churn	Sum of the lines added and removed in all the classes being subject to review.	Large classes are hard to maintain and can be more prone to be subject to refactoring [19, 76].
		Independent variables	
Communication Dynamics Aspect	Number of Users	Number of unique users that interacted in any way in a discussion inside a Pull Requests (opened, commented, merged or closed)	This metric allows us to identify discussions with the presence of common users, constant contributors, experienced developers or core members of the project [9]. The classification method can be found on [7].
	Number of Contributors	Number of unique contributors that interacted in any way in a Pull Request (opened, commented, merged or closed)	
	Number of Core Developers	Number of unique core developers that interacted in any way in a Pull Request (opened, commented, merged or closed)	
	Pull Request Opened By	The type of user that has opened each pull request. The user might be an Employee or Temporary. Employees are active contributors and code developers. Conversely, temporary are developers that do not actively work on the project or does not work for the software organization	Pull Requests opened by temporaries have more risk of increasing design symptoms [81]. The classification method can be found on [7].
	Number of Comments	Number of comments inside a Pull Request.	Discussions with a high number of comments around a code change would find possible design symptoms, improving or maintaining the quality [9].
	Mean Time Between Comments	Sum of the time between all comments of a Pull Request weighted by the number of comments.	A higher time between comments (e.g., a long pause in an otherwise fast-paced discussion) are related to design decay [9].
	Discussion Length	Time in days that a Pull Request lasted (difference of creation and closing days).	The longer is the discussion, the higher the chance of problems being explained and solved, avoiding design decay [81].
Discussion Contents Aspect	Number of Snippets in Discussion	The number of snippets inside each comment of a Pull Requests. Those snippets are detected by the number of ‴ (syntax that opens a snippet in markdown) divided by two (opening and closing).	The higher the number of snippets in a discussion, the clearer the users are trying to pass a message. Therefore, avoiding confusion and possibly design decay [9].
	Snippet Size	Sum of the size of all snippets found on comments in a Pull Request.	
	Number of Words in Discussion	Sum of the all words of each comment inside a Pull Request. Here we applied the preprocessing in the text removing contractions, stop words, punctuation, and replacing numbers.	Discussions with a high number of words are related to more complex changes, that may lead to design decay [9].
	Number of Words per Comment in Discussion	Sum of the all words of each comment inside a Pull Request weighted by the number of comments. Here we applied the preprocessing in the text removing contractions, stop words, punctuation, and replacing numbers.	Discussions with a high weighted number of words are related to more complex changes, that may lead to design decay [9].

Step 5: Assessing the relationship between social aspects and impactful pull requests. We use a statistical approach to determine which social metrics are able to discriminate between impactful pull requests and the unimpactful ones. We observe that the social metrics are not normally distributed [39]. Thus, we use the *Wilcoxon Rank Sum Test* [80] to decide whether a social metric is *statistically different* for impactful pull requests when compared to the unimpactful ones. The test was conducted using the customary .05 significance level.

Step 6: Evaluating the influence of multiple social aspects on design decay. We assess the influence of each social aspect over the design decay. For this purpose, we created a *multiple logistic regression* model for each aspect, by considering each metric that composes an aspect in the presence of each one other. Additionally, we also created a *multiple logistic regression* model that combines all social aspects and their related metrics together. All the social aspects and their related metrics presented in Table 3.3 are predictors in the model, and the outcome variable is whether there was decay on the design symptoms related to the merged pull request. We choose a *multiple logistic regression* approach due to the fact that we are studying the effect of multiple predictors (i.e., the metrics) in a binary response variable. We remove from our models the metrics that have a pair-wise correlation coefficient above 0.7 [14] to avoid the effects of *multicollinearity*.

Furthermore, we measure the relative impact to understand the magnitude of the effect of the metrics over the possibility of a merged pull request degrading the system design. We estimate the relative impact using the odds ratio [15]. In our study, odds ratios represent the increase or decrease in the odds of a pull request degrading the system occurring per "unit" value of a predictor (metric). An odds ratio < 1 indicates a decrease in these odds (i.e., a risk-decreasing effect), while > 1 indicates an increase (i.e., a risk-increasing effect). Most of our metrics presented a heavy skew. To reduce it, we apply a $\log_2$ transformation on the right-skewed predictors and a x^3 transformation on the left-skewed. Moreover, we normalize the continuous predictors in the model to provide normality. As a result, the mean of each predictor is equaled to zero, and the standard deviation to one.

To ensure the statistical significance of the predictors, we employ the customary p-value $< .05$ for each predictor in the regression models. Finally, we also report the amount of deviance accounted for by our *multiple logistic regression* models, in terms of the *D-squared* [25]. Similar to *R-squared* [39] for linear regression models, the *D-squared* represents the goodness-of-fit of logistic regression model, measured by the residual deviance (i.e., the deviance

PUC-Rio - Certificação Digital Nº 1912701/CA

that is unexplained by the model). A perfect model has no residual deviance and its *D-squared* takes the value of 1.

3.5 Results and Discussion

3.5.1 Social Metrics and Impactful Pull Requests

We address $\mathbf{RQ}_1$ by understanding which social metrics can discriminate between impactful pull requests and unimpactful pull requests. As described in Step 3 of Section 3.4.2, we consider that a merged pull request is impactful when it *increases* or *decreases* the design decay. Conversely, unimpactful pull requests do not affect the design decay. Table 3.4 shows the results of the *Wilcoxon Rank Sum test* [80] grouped by social aspects metric, design decay symptom, i.e., low-level and high-level structural smells, and system. Each row represents the *p*-values of the metrics obtained as results of the *Wilcoxon Rank Sum test* for each grouping. The last column (All) represents the results from all systems combined. The cells in gray represent the *p*-values that obtained statistical significance (i.e., *p*-value $< .05$), where there is a valid distinction between impactful and unimpactful pull requests.

Table 3.4: Results of the wilcoxon rank sum test grouped by social metrics, design decay symptom and software system

Social Metrics	Elasticsearch		Netty		Okhttp		Presto		RxJava		All	
	High-level	Low-level	High-level	Low-level	High-level	Low-level	High-level	Low-level	High-level	Low-level	High-level	Low-level
# Comments	<.001	<.001	.019	.019	<.001	<.001	<.001	<.001	.026	.026	<.001	<.001
# Users	.008	.008	.060	.060	.093	.093	<.001	<.001	.156	.156	<.001	<.001
# Contributors	.002	.002	.701	.701	<.001	<.001	<.001	<.001	.006	.006	<.001	<.001
# Core Devs	<.001	<.001	.117	.117	.002	.002	.006	.006	.251	.251	<.001	<.001
MTBC	<.001	<.001	.001	.001	<.001	<.001	<.001	<.001	.001	.001	<.001	<.001
DL	<.001	<.001	.034	.034	<.001	<.001	<.001	<.001	.065	.065	<.001	<.001
# of Snippets	<.001	<.001	.022	.022	.007	.007	<.001	<.001	.021	.021	<.001	<.001
NWD	<.001	<.001	.005	.005	<.001	<.001	<.001	<.001	.003	.003	<.001	<.001
NWPCD	<.001	<.001	.004	.004	<.001	<.001	<.001	<.001	.016	.016	<.001	<.001
Snippets Size	<.001	<.001	.022	.022	.011	.011	<.001	<.001	.018	.018	<.001	<.001

The relationship with impactful pull requests. Table 3.4 reveals some interesting conclusions. First, we observed that many metrics were *statistically different* for impactful pull requests when compared to unimpactful ones. This observation is consistent in all projects analyzed. Moreover, note that the Discussion Length (DL) and the Number of Contributors (# Contributors) metrics are statistically different in 4 out of 5 projects (80%). We also observed that the Number of Users and Number of Core Developers metrics reached statistical significance in 40% and 60%, respectively. Additionally, social metrics related to the participant role (communication dynamics) were the ones that presented a more unstable behavior, only 66% of the cases were statistically different. These results show that social metrics can differentiate impactful pull requests from the unimpactful pull requests. Metrics from both

PUC-Rio - Certificação Digital Nº 1912701/CA

social aspects might be good indicators to identify potential impactful pull requests. This result evidence the rationale that pull requests with high values for these metrics need more attention and concern of software organizations. For such cases, software organizations could monitor significant changes in the values of these metrics. Such changes may indicate an increase or decrease in design quality.

Finding 1: Social aspects are able to differentiate impactful pull requests from the unimpactful ones. Besides, software organizations could monitor significant changes to avoid design decay.

3.5.2 Communication Dynamics and Decay

We address $\mathbf{RQ}_2$ by understanding the influence of the *communication dynamics* over the design decay, we assessed the effect of each metric that composes this aspect in the presence of each one other. We have applied a *multiple logistic regression* to support this assessment (Step 6 of Section 3.4.2). The results of this analysis are summarized in Table 3.5. Each row contains the results of the metrics for each project, divided by high-level structural smells and low-level structural smells. The last column presents the *D-squared* of the regression model and the percentage increase or decrease in the *D-squared* compared to a model with only control metrics. The last row contains the results for the data of all projects combined. Moreover, the grey cells represent the statistically significant metrics (p-value $< .05$) and the arrows represent the following behavior: risk-increasing (arrow up) and risk-decreasing (arrow down). Finally, the blank cells represent metrics that were removed from the model for being collinear. We discuss the results as follows.

Table 3.5: Results of the odds ratio analysis for the communication dynamics

System	Symptom	Control Variables				Communication Dynamics Aspect							D-squared
		PS	PC	DC	DS	# Commts	# Users	# Contrib	# Core Devs	OBTemp	MTBC	DL	
elasticsearch	High Level	0.576 ↓	6.992 ↑			0.805	1.076	1.358	1.1	2.006	1,18	1.023	0.077 (-10.46%)
	Low Level	0.576 ↓	6.992 ↑			0.805	1.076	1.358	1.1	2.006	1,18	1.023	0.077 (-10.46%)
netty	High Level				1.835 ↑	1.038	0.997	0.987	1.037	4.336	1.185 ↑	1.302 ↑	0.432 (42.57%)
	Low Level				1.835 ↑	1.038	0.997	0.987	1.037	4.336	1.185 ↑	1.302 ↑	0.432 (42.57%)
okhttp	High Level		3.572 ↑				1.07	0.921	0.978	0.473	1.227 ↑	1.025	0.240 (7.14%)
	Low Level		3.572 ↑				1.07	0.921	0.978	0.473	1.227 ↑	1.025	0.240 (7.14%)
presto	High Level		10.753 ↑			0.747	0.627	0.68	1.265	4.266	1,914	1.022	0.214 (239.68%)
	Low Level		10.753 ↑			0.747	0.627	0.68	1.265	4.266	1,914	1.022	0.214 (239.68%)
rxjava	High Level				1.875 ↑		0.906	1.176	0.938	1.088	1,223	0.89	0.045 (28.57%)
	Low Level				1.875 ↑		0.906	1.176	0.938	1.088	1,223	0.89	0.045 (28.57%)
all	High Level		3.815 ↑			0.911	1.099 ↑	1.001	0.949	1.046	1.178 ↑	1.071	0.209 (409.75%)
	Low Level		3.815 ↑			0.911	1.099 ↑	1.001	0.949	1.046	1.178 ↑	1.071	0.209 (409.75%)

Risk-increasing effect of communication dynamics. Table 3.5 shows that one out of the five metrics related to a specific role of the developer (i.e, user) involved in a discussion, i.e., the Number of Users (# Users), has a risk-increasing tendency. Additionally, all metrics related to temporal aspects of communication i.e., Mean Time between Comments (MTBC) and Discussion

PUC-Rio - Certificação Digital Nº 1912701/CA

Length (DL) also presented a risk-increasing tendency. More precisely, the metric # Users was statistically significant when we combined the data of all projects. Conversely, the Discussion Length was statistically significant only for the Netty project. Finally, the Mean Time between Comments was statistically significant for two projects (Netty and OkHttp), and when we combined all data.

These results reinforce two rationales. First, pull requests with a high mean time between comments are related to design decay. In other words, pull request discussions with a high delay between the comments lead to poor workflow with little communication dynamics between developers, indicating a risk-increasing effect. The delay is usually related to either the lack of interest or the fact the proper knowledge is being forgotten along the conversation. Second, a higher number of users participating in a discussion may hinder the communication dynamics, mainly when these users are not familiar with a system feature, or just by filling the discussion environment with meaningless messages. For such cases, the developers involved can be induced to increase the complexity of the change. Such changes can lead to an increase in design decay as observed in our motivating example (Section 3.3).

Moreover, the result about pull requests with long discussions which presented a decay risk-increasing effect was surprising. One may expect that longer discussion increases the chances of issues being explained and resolved, avoiding design decay. Such a result suggests that long discussions do not increase the developer's engagement on being conscious with the design structure, increasing the chances of design decay. We observe this phenomenon in the pull request #1388 from the OkHttp project, for instance.

This pull request was opened by a contributor that explained their code changes in detail through two big comments (131 and 179 words each).

Almost a month later, a core member of the project reviews the code and replies to the author with an apology for the delay in the review. Next, the author replies to the reviewer a week later.

After a long review process, the core member asked to author why the tests were failing. Then, the author replied with a big comment (213 words) answering the concern about the test of the core member. Moreover, the pull request merge only would happen three weeks later, when, again, the reviewer apologized for his delay. In the end, this pull request did induce an increase in the design decay, confirming our rationale for `MTBC` and `DL` metrics.

These observations suggest that future empirical studies should focus more on the temporal aspects of communication dynamics, which have been neglected so far. When one has to focus on a reduced set of metrics, such

PUC-Rio - Certificação Digital Nº 1912701/CA

metrics of temporal factor should be part of his priorities. Also, the core members of organizations should pay more attention to this kind of social factor to avoid design decay.

> **Finding 2:** Social metrics related to temporal aspects of communication dynamics work better as indicators of design decay increase than metrics related to the role of participants.

Risk-decreasing effect of communication dynamics. The data presented in Table 3.5 also allows us to observe that metrics that measure the communication flow among developers during a pull request discussion provide a better indication of the design decay than metrics that measure the role of the developers. By complementing the previous finding, we also observed that metrics related to the communication dynamics aspect, in general, help to characterize only the risk-increasing effect. This happens even in the presence of control metrics.

> **Finding 3:** In general, the aspect of communication dynamics is a better indicator of increase in design decay symptoms.

The strength of communication dynamics metrics compared to current models. Finally, we also assess to what extend the metrics related to communication dynamics are complementary with the control variables. As shown in Table 3.5, we used four control variables that represent variables widely used in the current models of design decay analysis (Patch Size, Diff Size, Diff Complexity, and Patch Churn). We note that in the three cases where communication dynamics metrics were statistically significant in comparison with the control variables, the *D-squared* of the models only increased, ranging from 7.14% (OkHttp) up to 409.75% (all data combined). Such finding indicates that the use of communication dynamics metrics increases the explanatory power of design decay models when compared to models with only control metrics.

> **Finding 4:** The metrics of communication dynamics can increase the explanatory power of current design decay models. Thus, metrics from this aspect are relevant indicators of design decay.

PUC-Rio - Certificação Digital Nº 1912701/CA

3.5.3 Discussion Content and Decay

By following the same procedures of the previous research question, in $\mathbf{RQ}_3$ we aim to understand the influence of the *discussion content* over the design decay. For this purpose, we also assessed the effect of each metric that composes this aspect in the presence of each one other. The results of this analysis are summarized in Table 3.6, in which we also used the *odds ratio* technique to explain the effect of this social aspect over the design decay. Similarly to Table 3.5, the grey cells represent the statistically significant metrics (p-value $< .05$), and the arrows represent the risk-increasing (arrow up) and risk-decreasing (arrow down) effects. The blank cells are the metrics missing due to *collinearity*. Finally, the last column presents the *D-squared* of the regression models.

Table 3.6: Results of the odds ratio analysis for the discussion content

System	Symptom	Control Variables				Discussion Content Aspect				D-squared
		PS	PC	DC	DS	Number of Snippets	NWD	NWPCD	Snippet Size	
elasticsearch	High Level		4.488 ↑		0.817	7.772	1.187		0.137	0.050 (-41.86%)
	Low Level		4.488 ↑		0.817	7.772	1.187		0.137	0.050 (-41.86%)
netty	High Level			1.882 ↑		1.075		1.125	1.115	0.319 (5.28%)
	Low Level			1.882 ↑		1.075		1.125	1.115	0.319 (5.28%)
okhttp	High Level	2.391 ↑				1.508		1.018	0.764	0.215 (-4.01%)
	Low Level	2.391 ↑				1.508		1.018	0.764	0.215 (-4.01%)
presto	High Level	4.461 ↑						1.141	13.848	0.128 (103.17%)
	Low Level	4.461 ↑						1.141	13.848	0.128 (103.17%)
rxjava	High Level				1.858 ↑	0.332		0.804 ↓	3.542	0.045 (28.57%)
	Low Level				1.858 ↑	0.332		0.804 ↓	3.542	0.045 (28.57%)
all	High Level	2.023 ↑				1.097	9.907 ↑	0.105 ↓	0.968	0.109 (165.85%)
	Low Level	2.023 ↑				1.097	9.907 ↑	0.105 ↓	0.968	0.109 (165.85%)

Risk-increasing effect of the discussion content. Table 3.6 presents that only the metric Number of Words in Discussion (`NWD`) presented a risk-increasing tendency when we combined the data of all projects. To understand this result we also need to address another metric of the table, the Number of Words per Comment in Discussion (`NWPCD`), which presented a risk-decreasing effect, since the rationale of these two metrics is linked. The metric `NWD` indicated that the higher the number of words in a discussion the bigger would be the risk of design decay. However, the metric `WPCD` shows us that the higher is the number of words, but weighted by the number of comments, the smaller would be the risk of design decay. Moreover, these two metrics do not conflict, but they are complementary as indicators of design decay.

We observed that the `NWD` can indicate an increase on design decay when: (1) a discussion has a high number of comments and a high number of words, however, these words are concentrated only in a few comments; and (2) a discussion have a high number of comments and a low-mid amount of words in each comment. Additionally, these high numbers of words concentrated on few comments may be indicators that only few participants were truly engaged or providing useful pieces of information. Hence, the case (2) happens when the

PUC-Rio - Certificação Digital Nº 1912701/CA

conversation: (i) does not contain a set of relevant information, (ii) message contents do not say much.

To exemplify the cases, we can observe the Pull Request #1388 from the OkHttp project, already discussed in this paper, regarding the case (1) aforementioned. The discussion on this pull request presented seven comments, three of them containing over one hundred words (131, 178 and 213), and they were all made by the author of the pull request. However, the other four remaining ones did not have more than 35 words, which were made by a core member working on reviewing the pull request.

These results imply that the size of a discussion is related to design decay. The developers should evaluate the quality of their comments since their size alone do not avoid the design decay.

> **Finding 5:** Discussions with a high number of words, when it is not accompanied by a high number of words per comment, work as indicators of design decay increase.

Risk-decreasing effect of the discussion content As stated in the previous finding, we also observed that the metric Number of Words per Comment in Discussion (NWPCD) presented a risk-decreasing effect. Moreover, this behavior was observed on the RxJava project, and when we combined all data of projects. This result also suggests that, even in situations where the number of comments is low, but the number of words per comment is high, there is a high volume of information that may be related to the complexity of the change. As well as discussion on changes that may affect the structural quality of the system. In other words, there is a concentration of useful comments.

> **Finding 6:** In general, the discussion content can indicate the increase (number of words in discussion) and the decrease (number of words per comment in discussion) of design decay symptoms.

The strength of discussion content metrics compared to current models. Similarly to RQ_2, we assess how discussion content metrics can complement control metrics. Table 3.6 presents the *D-squared* measure of the studied models and the percentage increase or decrease in this measure when compared to current models. In summary, we observed two cases in which discussion content metrics were statistically significant: the *D-squared* of the models increased 28.57% (RxJava) and 165.85% (all data combined). These results indicate that the discussion content metrics are also able to increase the explanatory power of design decay models.

PUC-Rio - Certificação Digital Nº 1912701/CA

Finding 7: Discussion content metrics may be significant indicators of design decay due to the increase of explanatory power when included in current models.

The communication dynamics aspect vs. discussion content aspect. By comparing the results obtained by each social aspect in isolation (Tables 3.5 and 3.6), we observed that the discussion content metrics help to characterize both risk-increasing and risk-decreasing effects. However, the communication dynamics metrics, in general, help to characterize only the risk-increasing effect. This happens even in the presence of control metrics. In summary, this result indicates the future studies should consider both aspects, i.e., communication dynamics and discussion content. However, the discussion content aspect can be prioritized when there is a need for a reduced number of metrics. The metrics of this aspect when combined with control metrics help to characterize both risk-increasing and risk-decreasing of design decay.

Finding 8: The metrics related of discussion content provide a better indication of design decay than communication dynamics metrics, even in the presence of control metrics. In any case, both social aspects have shown to be good selecting indicators of design decay.

When looking at all social aspects together, which behavior do we see? Table 3.7 presents the results of the odds ratio analysis when we combined our two social aspects (*communication dynamics* and *discussion content*) and using the projects combined data. This table reveals interesting results. First, when we combine the two social aspects, new metrics related to the communication dynamics appear as statistically significant: Number of Comments and Number of Core Developers. Both metrics presented a risk-decreasing effect. Such a result reveals that the communication dynamics is also a good indicator of a decrease in design decay. Finally, the combination of social aspects did not lead to any of the previously discussed metrics being irrelevant.

These observations suggest that depending on the models you want to build and the aspects to be observed, new social metrics can rise as indicators of design decay. Table 3.7 allows us to observe that the addition of both social aspects' metrics to the current models resulted in an increase in the *D-squared* by 197.56%. These results indicate that communication dynamics and discussion content accounted for a major contribution of deviance compared to current models. Such a result highlights the importance of social aspects when studying design decay.

PUC-Rio - Certificação Digital Nº 1912701/CA

Table 3.7: Results of the odds ratio analysis with both social aspects together for all project data

Metrics	High level	Low level
Control variables		
Patch Size	1.971 ↑	1.971 ↑
Diff Size		
Diff Complexity		
Patch Churn		
Communication Dynamics		
Number of Users	1.109 ↑	1.109 ↑
Number of Contributors	0.885	0.885
Number of Core Developers	0.848 ↓	0.848 ↓
Opened By: Temporaries	0.902	0.902
Number of Comments	0.346 ↓	0.346 ↓
Mean Time Between Comments	1.081	1.081
Discussion Length	1.153 ↑	1.153 ↑
Discussion Content		
Number of Snippets in Discussion	1.334	1.334
Snippet Size	0.76	0.76
Number of Words in Discussion	15.288 ↑	15.288 ↑
Number of Words per Comment in Discussion	0.211 ↓	0.211 ↓
D-squared	0.122 (197.56%)	0.122 (197.56%)

Finding 9: When combining both social aspects in the model, two new metrics appear on the model as risk-decreasing indicators, suggesting that different metrics can emerge as design decay indicators in different situations.

PUC-Rio - Certificação Digital Nº 1912701/CA

3.6 Threats to Validity

Construct and Internal Validity. This study analyzes a range of 27 types of decay symptoms. Thus, our findings might be biased by these types, even though they are commonly investigated by previous works [48, 60, 61]. We have used the DesignateJava tool to detect these symptoms. Thus, aspects such as precision and recall may have influenced the results of this study. However, DesigniteJava has been used successfully in recent studies [60, 4, 48, 77], and previous work [62] indicated a precision of 96% and a recall of 99%. The metrics chosen to represent the social aspects may not fully represent all the possible interactions among developers that could lead to design decay. To mitigate this threat, we choose social aspects already analyzed by previous work and metrics of process and product, that are commonly used for design decay analysis.

Conclusion and External Validity. We carefully performed our descriptive and statistical analyses. Regarding the descriptive analysis, three paper authors contributed to the computation of the merged pull request impact on the density of symptoms. With respect to the statistical analysis, the metrics used in this study did not follow a normal distribution due to high skewness. To mitigate that, we used the non-parametric Wilcoxon Rank Sum

Test [80] on RQ_1 and, for the regression analysis, we reduced the heavy skew of our metrics by applying $\log_2$ and x^3 transformations. Moreover, since multi-collinearity of predictors may heavily affect the results of a multiple regression model [14], we removed from our models the predictors with pair-wise correlations above 0.7 (see Section 3.4). We also normalized the continuous predictors in the model to ensure normality. Furthermore, in our regression models, we controlled some factors that may affect our outcomes via control variables (see Section 3.4.2). Finally, we have investigated design symptoms in Java software systems only. Thus, our study results may be biased by the underlying code structure of Java-based systems, although we highlight that Java is one of the most popular programming languages in both industry and academia.

3.7 Conclusion and Future Work

This work investigated the relationship between two social aspects, *communication dynamics* and *discussion content*, and design decay. For that, we collected pull request data from five open-source systems, assessed 27 types of design decay symptoms, 4 control variables and 11 social metrics related to two social aspects.

From that analysis, we reported the following findings: (i) social aspects can distinguish impactful and unimpactful pull requests; (ii) temporal factors of the communication dynamics are better indicators of an increase in design decay than the developers' roles; (iii) the communication dynamics is the best indicator of increase on the design decay; (iv) social aspects should be included in current models of design decay analysis as they improve the explanatory power of the models. We believe that these findings can benefit both developers and software organizations, as they can be more aware of what their behaviors can impact the code they are producing.

As future work, we intend to assess the importance of contribution that are external factors to the analyzed projects. For instance, we plant to better understand other social aspects that relate to design decay, such as developers' experience and their interactions in other communities. These two aspects are not inherently developed along the present project, but instead accumulated from previous developers participation in other software projects. Moreover, we intend to expand our work on design decay to analyze a wider amount of technical (e.g., software metrics) and social (e.g., social network measures) aspects.

PUC-Rio - Certificação Digital Nº 1912701/CA

3.8 Summary

In the study presented in this chapter, we contributed for addressing the two specific problems of this dissertation (see Section 1.1). Regarding the first problem – i.e., lack of evidence on which social metrics distinguish impactful from unimpactful pull requests – we observed that all social metrics, from both aspects, are indeed able to distinguish impactful and unimpactful pull requests. Therefore, software organizations can monitor significant changes in such social metrics to understand design changes.

As previously mentioned, this study also contributed for addressing our second research problem: *"There little knowledge on the influence of communication dynamics, discussion content, and organizational dynamics on design decay"*. Regarding this problem, we only investigated the first two social aspects (i.e., communication dynamics and discussion content). Our results showed that temporal factors of the communication dynamics are better indicators of an increase in design decay than the role of developers. We also observed that communication dynamics metrics are better indicators of design decay when compared to the discussion content metrics.

In the next chapter, we present a differentiated replication of the study presented in this chapter. One of the goals of the replication is to overcome some of the weaknesses that affect our experimental design (more detail in the Chapter 4). We also aim to validate the results presented here in a larger dataset (5 vs. 7 systems). Finally, we aim to completely address our second research problem by investigating the *organizational dynamics* aspect together with the aspects investigated in this chapter.

PUC-Rio - Certificação Digital Nº 1912701/CA

4
On the Relationship between Social Aspects and Design Decay

Design decay is an important concern during software development. Empirical studies have demonstrated that different technical and social aspects are related to the increase and reduction of design decay [77, 76, 6, 86]. In this context, three central social aspects may contribute to increasing or reducing design decay: (i) *communication dynamics* among developers; (ii) *discussion content* itself may be decisive to either improve or deteriorate the structural design of a system; and (iii) *organizational dynamics* that represent the aspects of the team as a whole. The aspects (i) and (ii) were investigated in Chapter 3. However, the aspect related to *organizational dynamics* and how it influences design decay have not yet been explored.

Consequently, the lack of empirical investigation of this social aspect, does not allow us to reveal, for instance, the importance of team size and gender diversity on the design decay. Moreover, the manifestation of these three social aspects together along software development can induce behaviors that possibly affect the design quality of the code under development.

Thus, this chapter introduces a replication study aimed at better understanding the characteristics of *communication dynamics*, *discussion content*, and *organizational dynamics* on design decay. We report a differentiated replication of the work presented in Chapter 3 in which we (i) overcome some of the weaknesses that affect their experimental design, (ii) answer the same research questions of the original study on a much larger dataset (5 vs. 7 systems), and (iii) analyze the influence of new key social aspect – *organizational dynamics* – on design decay. Our results confirm and reassess the findings of the replicated study, and reveal that various factors of organizational dynamics are related to design decay. Such results are complementary to the obtained results in Chapter 3. Moreover, the study reported in their chapter also contributes to addressing research problems 1 and 2 (see Section 1.1) of this Master dissertation.

PUC-Rio - Certificação Digital Nº 1912701/CA

On the Relationship between Social Aspects and Design Decay

A Replication Study

Caio Barbosa, Anderson Uchôa, Willian Oizumi
PUC-Rio, Rio de Janeiro, Brazil
{csilva,auchoa,woizumi}@inf.puc-rio.br

Vinicius Soares, Daniel Coutinho, Alessandro Garcia
PUC-Rio, Rio de Janeiro, Brazil
{vsoares,dcoutinho,afgarcia}@inf.puc-rio.br

4.1 Introduction

During the pull request development model, developers communicate among each other in order to improve the code they are developing [13, 21, 68, 77]. This interchange of expertise between the code author and participants of the pull request is surrounded by many social aspects, for instance, the content of the messages being exchanged [81, 6]. These social aspects can be found in modern collaborative software development platforms, such as GitHub [1], through mechanisms as commit messages, pull requests conversations, issues comments, and organizational data.

With the need to maintain or increase the quality of the code, many studies assess factors related to design decay [66, 46, 83]. A program code is considered to have design decayed if it is more difficult to make a code change than it should be [16]. Moreover, design decay can be measured by multiple symptoms, also popularly known as code smells. These symptoms are indicators of structural design decay in the scope of methods and code blocks [61]. An example of design decay symptoms is the Long Method smell.

As number of newcomers on projects increases, the pull request development model becomes mandatory in many open-source communities. There are even extensive guidelines explaining how to make a good pull request [22, 23]. The existence of such guidelines is explained by the need for improving and maintaining the quality of the systems [40, 85, 34]. In this vein, each participating developer inspects the code owned by another developer and discusses the possible problems and the ways to improve it with the community. However, social software development may not always be beneficial. For example, previous studies [5, 43, 56, 77, 76] even suggest that code review activities may increase design decay in certain circumstances.

Barbosa *et al.* [6] conducted a study focusing on social aspects surrounding pull request conversations, which are discussions not directly related to source code but happening in parallel to code reviews. They used the Designite [60] tool to build a dataset of 27 decay symptoms from 23.280 commits,

[1] https://github.com

PUC-Rio - Certificação Digital Nº 1912701/CA

related to merged pull requests of five open-source projects hosted on GitHub: elasticsearch, presto, rxjava, netty, and okhttp. They mined 11.600 pull requests to generate a dataset of eleven social metrics for those systems. Finally, they applied two statistical tests: (i) Wilcoxon Rank Sum Test, to assess if social metrics were able to differentiate between impactful pull requests (decay increased or decreased) and non-impactful pull requests (decay did not change); and (ii) Multiple Logistic Regression, to evaluate the influence of each metric, in the presence of others metrics, on the design decay symptoms. Finally, they concluded that the social aspects are able to differentiate between both types of pull requests (impactful and non-impactful) and that multiple social aspects metrics are related to both increase and decrease of the design decay symptoms.

In Barbosa *et al.* [6], their main limitations are: (i) number of projects; (ii) only two social aspects (communication dynamics and discussion content); (iii) the high level of discussion content metrics – only dimensions related to its size; and (iv) their assumption of increases and decreases of design decay symptoms only relies on a quantitative tool (Designite). Thus, this study focus on overcoming these limitations by: (i) *Increasing the number of projects from 5 to 7;* and (ii) *Addition of a new social aspect.* While the previous considered aspects cover a large amount of information, they lack in social information about the organization itself. In order to improve upon this limitation, our study introduces a new social aspect, *organizational dynamics*.

In summary, our findings are presented in three ways: (i) confirmed; (ii) controversial, and (iii) new results. As confirmed results, we had that:

- Different social metrics work as indicators of design change in different contexts, confirming results from the original work.
- Number of Users, Meant Time Between Comments, and Discussion Length carried on their relevance as design decay indicators
- The discussion content metrics Number of Words in Discussion and Number of Words Per Comment in Discussion stood out as the best social metrics to identify design decay;
- When analyzing multiple social aspects and multiple projects at the same time, different social metrics emerge as indicators of both increase and decrease of design decay symptoms. For instance, the Team Size, Time Between Creation and First Comment, and Time Between Last Comment and Merge presented themselves as statistically significant in the model with all aspects and all projects.

However, our controversial results showed:

PUC-Rio - Certificação Digital Nº 1912701/CA

– Number of Core Developers, and the Opened By metrics presented a risk of increasing the design decay symptoms in this study.

Finally, as new results, we observed:

– Number of Refactoring Keywords and Refactoring Keyword Density presented promising results in relation to design decay symptoms;
– The presence of newcomers and the number of developers leaving the organization showed their importance in the impact on the design of the system;
– Gender Diversity did not present any relation to design decay, which is an interesting finding for gender studies.
– The organizational dynamics aspect showed promising result to indicate the decrease of design decay symptoms.

The remainder of this paper is organized as follows. Section 4.2 discusses the background and related work. Section 4.3 describes the study design. Our results are discussed in Section 4.4. The Threats to Validity and their mitigation are presented on Section 4.5, and Section 4.6 concludes our paper and presents future work.

PUC-Rio - Certificação Digital Nº 1912701/CA

4.2 Background and Related Work

Social aspects in software development. Multiple studies have analyzed the importance of social aspects in the software development process. Tamburri et al [69], for instance, collected social metrics to characterize and compute "social debt", they aimed at observing how social problems can directly affect decision-making in software development. They concluded that social debt lead to more error-prone decisions. Tamburri *et al.* [70] also performed a study about social aspects in industrial projects and provided the definition of common problems that can occur, together with potential mitigation to these problems. The main differences between these studies and our work are that they focus on how the social aspects of software engineering can cause on external software problems. Our work quite diverges from it by understanding how social aspects can cause decay-related problems that are internal to the software.

Social aspects versus software quality. When looking at the relationship between social aspects and software quality, Betternburg *et al.* [9] studied how social aspects can affect the quality of released software. The authors found out a set of aspects that had a statistically significant connection

to defects: (i) low code churn (which was used as the baseline); (ii) low number of external resources (links); and (iii) high variance of the time between comments in the discussions. Nagappan *et al.* [45] analyzed the relationship between organizational structure metrics and software quality. They concluded that a machine learning model trained with organizational structure metrics, e.g., % of organizational contributing to development, and level of code ownership, was more effective at predicting software quality loss than a model trained with technical code metrics, e.g., code churn, code complexity. Our work differs from these other two because we combine both organizational and social metrics, and we also analyze a much wider suite of metrics – which does not require direct access to the developers themselves, and can be mined from repositories.

Social aspects of quality improvement. In terms of the social aspects of software quality improvement, we can find the presence of self-affirmed refactorings. AlOmar *et al.* [2] performed a study of how refactorings can be detected not in the code changes, but also through the social discussions between developers. Along with this, Soares *et al.* [64] performed a study to understand the relationship between such self-affirmed refactorings and the improvements to the code, showing that developers do tend to discuss refactorings when they are performing more complex changes. Our work differs from these other two due to the difference in context. In our work, we analyzed how the presence of this social aspect, in refactoring situations, relates to the improvement of decay.

PUC-Rio - Certificação Digital Nº 1912701/CA

4.3 Study Design

Section 4.3.1 introduces our study goal and research questions. Finally, Section 4.3.2 describes the study steps and procedures.

4.3.1 Goal and Research Questions

The *goal* of our study is to replicate the study of Barbosa *et al.* [6], as well as possibly confirm, refute or refine their findings. Their work investigated to what extent the social aspects influence design decay. To this end, they analyzed social metrics in commits of pull requests – specifically, the metrics on communication dynamics and discussion content dimensions. They analyzed 11k pull requests on five open-source projects. We are considering all the three research questions of the preliminary study and complementing with a new

question. All the four questions are described below with their corresponding explanation.

Our goal is to perform an in-depth analysis on the influence of social aspects on design decay symptoms, as already assessed in Barbosa et al [6]. We introduce the RQs as follows:

RQ_1: *Are social metrics related to design decay?* – As aforementioned, this RQ mirrors RQ_1 from the preliminary work. Their approach to answer this RQ was to investigate if there was a statistically significant difference between social aspects and impactful/unimpactful pull requests. They did so by analyzing 11k pull requests, extracted from five open-source projects hosted on GitHub. Their definition of an *Impactful Pull Request* was a pull request in which an *increase* or *decrease* in design decay was observed as a result of merging the pull request. Conversely, an *Unimpactful Pull Request* was one in which their merging did not affect the design decay. However, the size of their dataset and the number of analysed metrics could somehow obfuscate their findings. To find this statistical significance, they applied the *Wilcoxon Rank Sum Test* [80]. Further details are described in Section 4.3.2, Step 6.

RQ_2: *To what extent do the communication dynamics influence design decay?* and **RQ_3:** *To what extent do the discussion content influence design decay?* In order to address RQ_2 and RQ_3, we also went beyond the preliminary study. We derived and assessed two additional metrics in the communication dynamics dimension, and four new metrics in the discussion content dimension for this analysis. These additions enable us to achieve a deeper understanding of these aspects, as we also increased the number of projects. Finally, we applied the Bonferroni correction to the p-values [39] in order to decrease the number of type I errors. Since the multiple logistic regression runs the model N x N times, being N the number of metrics, this models often have statistical errors that spread over the p-values. This correction is able to minimize this error spreading.

RQ_4: *To what extent do the organizational dynamics influence design decay?* – This RQ expands on the analysis performed by the previous work [6]. By applying the same methodology for RQ_2 and RQ_3, we investigate if any of a set of multiple social metrics related to the organizational dynamics dimension relate to design decay. Our motivation is to further investigate other social aspect, so that organizations can have a more holistic view of the importance of certain aspects of their team in collaborative coding communities.

Our final goal is to provide a set of metrics that can be used. Being produced in the context of social aspects monitored along discussions in pull requests. Monitoring social metrics in this context can early to indicate

PUC-Rio - Certificação Digital Nº 1912701/CA

the increase or the decrease of design decay symptoms before changes are submitted and merged into the main branch. By doing this, we can shed light on future work on social aspects, and design decay, for software communities, so they can monitor their social behaviors in order to decrease the quality of the software being produced.

4.3.2 Study Steps and Procedures

In this section, we will describe our study steps. First, we follow the same steps of Barbosa et al [6], and then complement them with other steps.

Step 1: Selecting open-source systems. In this step, we first follow the criteria defined by Barbosa et al [6], which are: (i) systems that use pull request reviews as a mechanism to receive and evaluate code contributions; (ii) systems that have at least 1k commits and pull requests; and (iii) systems that are at least 5 years old, and are currently active. These criteria were selected by them in order to avoid known perils [31] of mining software repositories. Finally, the focus on Java systems was due to the constraints of the DesigniteJava tool [60]. As an addition to this step, we choose two more projects to be added to our study: ExoPlayer and Spring Security. This amount of new projects is justified by the high memory and storage cost of the data through the extraction with the DesigniteJava [60] and the GitHub API. For instance, ExoPlayer needed a computer with 85 GBs of RAM and 1,5 TB of storage to fully support the execution of the tool. Moreover, its run time was of 2 weeks. Thus, this is the main limitation imposed on the number of projects that we could add. It also explains why the preliminary work kept their number at five only. Table 4.1 provides details about each selected system. The columns respectively show: the names of each selected system; system's domain; number of commits; number of pull-requests; and time period considered in this study.

Table 4.1: Software systems investigated in this study

System	Domain	# Commits	# Pull-requests	Time span
Elasticsearch	Search Engine	17,251	4598	2011-2018
Presto	Query Engine	1958	1542	2012-2019
Netty	Framework for Network	4071	147	2011-2019
OkHttp	HTTP client	9690	4013	2012-2018
RxJava	Android Library	4140	1299	2013-2019
ExoPlayer	Music Player	10688	959	2014-2020
Spring Security	Security Library	9225	1536	2004-2020

Step 2: Detecting multiple design decay symptoms. In this step, we applied the same methodology of the original work, complemented by a validation using self-affirmed refactorings (see Step 5). We used the DesigniteJava tool [60] to detect the same 27 decay symptoms types – 17

PUC-Rio - Certificação Digital Nº 1912701/CA

high-level structural smells, and 10 low-level structural smells (see Table 4.2). Moreover, we only considered commits from merged pull requests. For each pull request PR_i, we downloaded a snapshot of each commit C_i that has been part of PR_i. Next, we calculated the difference between C_i and C_{i-1} in order to guarantee that the introduced design symptoms belonged only to C_i. By doing this, we avoid the rebase effect [51, 52], due to such a pull request being the only potential point in time in which the code could be changed. The descriptions, detection strategies, and thresholds for each symptom are available in our replication package [7].

Table 4.2: Degradation symptoms investigated in this study

High-level symptoms
Imperative Abstraction, Multifaceted Abstraction, Unutilized Abstraction, Unnecessary Abstraction, Deficient Encapsulation, Unexploited Encapsulation, Broken Modularization, Insufficient Modularization, Hub Like Modularization, Cyclic Dependent Modularization, Rebellious Hierarchy, Wide Hierarchy, Deep Hierarchy, Multipath Hierarchy, Cyclic Hierarchy, Missing Hierarchy, Broken Hierarchy [61].
Low-level symptoms
Abstract Function Call From Constructor, Complex Conditional, Complex Method, Empty Catch Block, Long Identifier, Long Method, Long Parameter List, Long Statement, Magic Number, Missing Default [61].

Step 3: Computing design decay indicators in terms of density symptoms. We first relied on similar steps as those from Barbosa et al [6], selecting the density and diversity of symptoms as indicators of progressive design decay. This decision is also grounded on decisions and observations made in previous studies [66, 48, 50, 77, 86]. In summary, a positive difference in the density (or diversity) of symptoms indicates an *increase* of the design decay as a result of the merged pull request. Therefore, it represents a design worsening. Similarly, a negative difference in the density of symptoms indicates a *decrease* of the design decay as a result of the merged pull request. Finally, a difference equal to zero in the density of symptoms indicates that there has been no harmful design structure change. This difference is calculated by the following methodology. For each commit C_i of the dataset, the tool generates code smell data for C_i and C_{i-1}, where C_{i-1} represents the parent commit of C_i, to compare the smells present on both commits. By doing that, they can see how many smells were added or removed, resulting in the calculation of density and the diversity. In total, we have computed the four indicators, i.e., density and diversity for high and low-level decay symptoms. A total of 11,599 merged pull requests were considered. We provide all computed indicators in our replication package [7].

Step 4: Detecting self-affirmed refactorings. We apply the empirical framework of AlOmar et al [3] to extract the self-affirmed refactorings (SAR) from the messages of pull request commits. The SAR is a confirmation, made

PUC-Rio - Certificação Digital Nº 1912701/CA

by the developers themselves, that they have performed a refactoring and, thus, that the code design should have to be structurally improved. This framework uses a model based on Azure Machine Learning [44], which was built upon a dataset containing commit messages of over three thousand projects. It is a single-class model that determines if a commit message contains a SAR, or if it does not. With this algorithm, the authors were able to achieve an accuracy of 98%.

Step 5: Calculating control metrics and social aspects. The work of Barbosa et al [6] computed 11 metrics related to social aspects. In order to increase the number of social aspects we want to analyze, we added 11 new metrics to this step. The computation of these metrics was performed using in the same three steps performed in the preliminary work: (i) collect the issues, pull requests, and related commits and comments from the selected projects through the GitHub API; (ii) extract the information needed for calculating the metrics from the collected data, and (iii) calculate the metrics following the methodology of each metric. Tables 4.4, 4.5 and 4.6 show both the 11 original metrics and the new 11 ones, totalling the 22 metrics we have used to measure social aspects affecting and interfering with the code development artifacts. Table 4.3 shows the control variables, which we computed in order to avoid factors that may affect our outcome, if not adequately controlled. For this end, we used *product* and *process* metrics, which have been shown by previous research to be correlated with design decay [54, 33]. Tables 4.4, 4.5 and 4.6 describe the metrics that we considered as independent variables to measure certain social aspects. We have grouped each metric in one of three categories, each one representing a dimension of a social aspect. *Communication dynamics* represents the dynamic of the discussion activity, such as the role of participants involved in a discussion or temporal aspects of the messages. *Discussion content* represents the interaction of developers during the exchange of messages, and information about the content of each message, such as, the number of snippets written in a discussion. Finally, *Organizational dynamics* represents the aspects of the team as a whole, such as the size of the team in the time of a discussion.

Step 6: Assessing the relationship between social aspects and impactful pull requests. We use the same statistical approach to determine which social metrics are able to discriminate between impactful and unimpactful pull requests, much like the original work. We also observe that the social metrics are not normally distributed [39]. Thus, we use the *Wilcoxon Rank Sum Test* [80] to decide whether a social metric is *statistically different* for impactful pull requests, when compared to the unimpactful ones. The test

PUC-Rio - Certificação Digital Nº 1912701/CA

Table 4.3: Control Variables

Type	Metrics	Description	Rationale
Product	Patch Size	Number of files being subject of review.	Large patches can be more prone to be analyzed for how the involved classes are designed.
	Diff Size	Difference of the sum of the Lines of Code metric computed on version before and after review of all classes being subject of review	Large classes are hard to maintain and can be more prone to be refactoring [53]
	Diff Complexity	Difference of the sum of the Weighted Method per Class metric computed on the version before and after review of all classes being subject of review.	Classes with high complexity are potential candidates to be refactored
Process	Patch Churn	Sum of the lines added and removed in all the classes being subject to review.	Large classes are hard to maintain and can be more prone to be subject to refactoring.

was conducted using the customary .05 significance level. Furthermore, we also used the *Cliff's Delta (d)* measure [24] in **RQ1** as a means to evaluate how strong is the difference between impactful and unimpactful pull requests, in terms of the analysed metrics. Similarly to the *Wilcoxon Rank Sum* test, the *Cliff's Delta (d)* is a non-parametric effect size measure. We employed a well-known classification [58] in order to interpret the Cliff's Delta (d) effect size. It defines four categories of magnitude: *negligible*, *small*, *medium*, and *large*.

Step 7: Evaluating the influence of multiple social aspects on design decay. We assess the influence of each social aspect over the design decay. For this purpose, we rely on a *multiple logistic regression* model created by Barbosa et al [6] for each aspect. In this model, we analyze each group of social aspects separately. Moreover, the *multiple logistic regression* calculates the odds ratio using each metric in the presence of each one other. All the social aspects and their related metrics presented in Tables 4.4, 4.5 and 4.6 are used as predictors in the model, and the outcome variable is whether or not there was decay on the design symptoms related to the merged pull request. We choose a *multiple logistic regression* approach due to the fact that we are studying the effect of multiple predictors (i.e., the metrics) in a binary response variable. We removed the metrics that have a pair-wise correlation coefficient above 0.7 [14] from our models to avoid the effects of *multicollinearity*. We also applied the Bonferroni [39] correction on the p-values to assure their validity.

To measure the relative impact, we aim at understanding the magnitude of the effect of the metrics over the possibility of a merged pull request on degrading the system design. To this end, we estimate the relative impact using the odds ratio [15]. Odds ratios represent the increase or decrease in the odds of an event happening. In our case, we measure the odds of the merge of a pull request degrading the system occurring per "unit" value of a predictor (metric).

PUC-Rio - Certificação Digital Nº 1912701/CA

An odds ratio below 1 indicates a decrease in these odds (i.e., a risk-decreasing effect), while above 1 indicates an increase (i.e., a risk-increasing effect). Since most of our metrics presented a heavy skew, we needed to reduce them. To do so, we applied a $\log_2$ transformation on the right-skewed predictors, and a x^3 transformation on the left-skewed. Moreover, we normalized the continuous predictors in the model to provide normality. As a result, the mean of each predictor is equaled to zero, and the standard deviation to one. To ensure the statistical significance of the predictors, we employ the customary p-value $<$ .05 for each predictor in the regression models.

Table 4.4: Communication Dynamics Dimension

Metrics	Description	Rationale
Number of Users	Number of unique users that interacted in any way in a discussion inside a pull request (either opened, commented, merged or closed)	These three metrics allows us to identify discussions with the presence of common users, constant contributors, experienced developers or core members of the project
Number of Contributors	Number of unique contributors that interacted in any way in a pull request (either opened, commented, merged or closed)	
Number of Core Developers	Number of unique core developers that interacted in any way in a pull request (opened, commented, merged or closed)	
Pull Request Opened By	The type of user that has opened each issue or pull requests. The user might be an Employee or Temporary. Employees are active contributors and code developers. Conversely, temporary users are contributors that do not actively work on the project or does not work for the software organization	Issues or pull requests opened by temporaries have more risk of degrading the code
Number of Comments	Number of comments inside a Pull Request.	Discussions with a high number of comments around a code change would find possible design problems, improving or maintaining the quality
Mean Time Between Comments	Sum of the time between all comments of a pull request weighted by the number of comments.	A higher time between comments (e.g., a long pause in an otherwise fast-paced discussion) are related to code decay
Discussion Duration (Discussion Length)	Time in days that a pull request lasted (difference of creation and closing days).	The longer is the discussion, the higher the chance of problems being explained and solved, avoiding code decay
Time Between Opened and First Comment	Time in days between the pull request opening and the first comment on that pull request	The longer the time between the opening and the first comment, the higher the chance of the developer do not really engage on solving possible problems, leading to design decay
Time Between Last Comment and Merge	Time in days between the last comment on the pull request and the pull request closing	The longer the time between the last comment and closing of the pull request, the higher the chance of the author does not engage on new minor changes, leading to design decay.

4.4 Results and Discussion

We will focus our discussion on contrasting the results of this replication study with those of the previous study (Chapter 3). Thus, we will present our findings in three different ways: (i) confirmed observations: those discussions concerning the findings from the original study that ware also observed in this replication; (ii) controversial observations: discussions about the findings from

PUC-Rio - Certificação Digital Nº 1912701/CA

Table 4.5: Discussion Content Dimension

Metrics	Description	Rationale
Number of Snippets in Discussion	The number of snippets inside each comment of a pull requests. Those snippets are detected by the number of "' (syntax that opens a snippet in markdown) divided by two (opening and closing)	The higher the number of snippets in a discussion, the clearer the users are trying to pass a message. Therefore, avoiding confusion and possibly code decay
Snippet Size	Sum of the size of all snippets found on comments in a pull request	The bigger the size of snippets in a discussion, the clearer the users are trying to pass a message. Therefore, avoiding confusion and possibly code decay
Mean Snippet Size	Snippet Size weighted by the number of snippets	The higher the mean of snippets in a discussion, the clearer the users are trying to pass a message. Therefore, avoiding confusion and possibly code decay
Number of Words in Discussion	Sum of the all words of each comment inside a pull request. Here we applied the preprocessing in the text removing contractions, stop words, punctuation, and replacing numbers	Discussions with a high number of words are related to more complex changes, that may lead to code decay
Number of Words per Comment in Discussion	Sum of the all words of each comment inside a pull request weighted by the number of comments. Here we applied the preprocessing in the text removing contractions, stop words, punctuation, and replacing numbers	Discussions with a high weighted number of words are related to more complex changes, that may lead to code decay
Number of Design Keywords	Sum of the all words of each comment inside a pull request that contains a keyword from the following list: design, architect, dependenc, requir, interface, servic, artifact, document, behavior, modul	Changes with design keywords may show that developers were concerned about design
Number of Refactoring Keywords	Sum of the all words of each comment inside a pull request that contains a keyword from the following list: refactor, mov, split, fix, introduc, decompos, reorganiz, extract, merg, renam, chang, restructur, reformat, extend, remov, replac, rewrit, simplif, creat, improv, add, modif, enhanc, rework, inlin, redesign, cleanup, reduc, encapsulat	Changes with refactoring keywords may show that developers were concerned about design
Density of design comments	Mean of 'design' keyword per comments	The higher the mean of 'design' comments, the smaller the chances of design decay
Density of refactoring comments	Mean of 'refactoring' keyword per comments	The higher the mean of 'refactoring' comments, the smaller the chances of design decay

Table 4.6: Organizational Dynamics Dimension

Metrics	Description	Rationale
Team Size	Number of Active Developers on the past 90 days	A bigger amount of active developers can engage more the community on discussions, avoiding design decay
Gender	Proportion of male/female contributors on the team	Gender diversity on a team leads to better team performance. Thus, avoiding design decay
Number of Newcomers	Number of new contributors on the past 90 days	A bigger amount of newcomers can introduce less experience on the changes, leading to design decay
Number of Developers Leaving	Number of developers that previously contributed to the project but did not contribute on the past 90 days	More developers leaving can decrease the engagement of the community, leading to design decay

the original study that were not observed here or had their behavior inverted; and (iii) new observations: new findings only derived in this replication study.

PUC-Rio - Certificação Digital Nº 1912701/CA

4.4.1 Social Metrics and Impactful Pull Requests

We report the results of our analysis replicating the methodology defined in the original work [6]. In this $\mathbf{RQ}_1$ we aimed to confirm if the original work's social metrics can discriminate between impactful pull requests and unimpactful pull requests. Moreover, we also evaluated new social metrics implemented in this study. As described in Step 3 of Section 4.3.2, we consider that a merged pull request is impactful when it *increases* or *decreases* the design decay. Conversely, unimpactful pull requests do not affect the design decay.

Table 4.7 shows the results of our analysis. The first column represents the social metrics, while the remaining ones describe the metrics and their respective magnitudes of the Cliff's Delta related to each studied project. In addition, we use the (+) symbol to indicate if d was positive, and the symbol (−) otherwise. In addition, we use four levels to measure the strength of a magnitude: (small) *; (medium) **; (large) ***; and (negligible) We do not use a symbol to represent this last level. The cells in gray represent the *p-values* of the metrics that obtained statistical significance in the *Wilcoxon Rank Sum* test.

The relationship with impactful pull requests. We observe that the process and product metrics behaved as expected, being statistically significant in 28 out of the 30 cases, resulting in medium and large strength level in 21 cases. Concerning the social metrics, we can see that this replication, using a restrict approach, made most of social metrics lose their statistic significance. The only two projects has now show significance were Elasticsearch and Presto projects. We observed that 19 out of 23 cases (Elasticsearch) and 9 out of 23 cases (Presto) were significant in these projects. However, the higher magnitude was "small" on 6 cases. Nevertheless, three to four new metrics showed themselves as significant in these projects, as some others from the original work also kept as they were. Finally, the two new projects, namely ExoPlayer and Spring-boot; showed the worst behavior with social metrics, having only two significant cases on the former and none on the latter.

Finding 1: The original social metrics kept their behavior observed in the original work, while others weakened with our stricter methodology used in this replication. The metrics from the organizational dynamics showed to be promising in 6 out of 7 projects, meaning that can be more consistently used to differentiate impactful and unimpactful pull requests.

PUC-Rio - Certificação Digital Nº 1912701/CA

Table 4.7: Statistical Significance (*p-value*) of the *Wilcoxon Rank Sum Test* and the Cliff's Delta (*d*) Magnitude Classification

Metric	Elastic	ExoPlayer	Netty	OkHTTP	Presto	RxJava	Spring Security
PS	(+) **	(+) *	(+) ***	(+) **	(+) ***	(+) ***	(+) ***
PC	(+) **	(+) **	(+) ***	(+) ***	(+) ***	(+) ***	(+) ***
DC	(+) *	(+)	(+) ***	(+) *	(+) **	(+) **	(+) ***
DS	(+) *	(+)	(+) ***	(+)	(+) **	(+) **	(+) ***
# Comments	(+) *	(−)	(+) *	(+) *	(+) *	(+)	(−) *
# Users	(+)	(+)	(+)	(+)	(+)	(+)	(+)
# Contributors	(+)	(−)	(−)	(+) *	(+)	(+)	(−)
# Core Devs	(+)	(−)	(+)	(+)	(+)	(+)	(−) *
# Employees	(−)	(+)	(−)	(+)	(+)	(−)	(−)
OBTemp	(+)	(−)	(+)	(+)	(−)	(+)	(+)
MTBC	(+)	(−)	(+) *	(+)	(+) *	(+)	(−)
DL	(+) *	(−) *	(+) *	(+)	(+) *	(+)	(−)
TBF	(+)	(−)	(−)	(+)	(+) *	(−)	(−) *
TBL	(+)	(−)	(+)	(+)	(+)	(+)	(−) *
# Patches	(+)	(+)	(+)	(+)	(+)	(+)	(−)
# Words	(+) *	(+)	(+) *	(+) *	(+) *	(+)	(−)
# Words/Comment	(+) *	(+)	(+) *	(+) *	(+) *	(+)	(−)
Snippet Size	(+)	(+)	(+)	(+)	(+)	(+)	(+)
Design	(+)	(+)	(+)	(+)	(+)	(+)	(+)
Refactoring	(+) *	(+)	(+)	(+)	(+) *	(+)	(−)
Design Density	(+)	(+)	(+)	(+)	(+)	(+)	(+)
Ref. Density	(+) *	(+)	(+)	(+)	(+)	(+)	(−) *
Team Size	(−)	(−) *	(−) **	(−)	(−)	(−)	(−)
# Male Devs	(+) *	(−)	(+) *	(+) *	(+)	(+)	(−)
# Female Devs	(+)	(−)	(+)	(−)	(+)	NA	(+)
# Newcomers	(+)	(−) *	(−) **	(−)	(−)	(−)	(−)
# Leaving Devs	(−)	(−) *	(−) **	(−)	(+)	(−)	(−) *

4.4.2 Communication Dynamics and Decay

The results of this analysis are summarized in Table 4.8. Each row contains the results of the metrics for each project. The last row contains the results for the data of all projects combined. Moreover, the grey cells represent the statistically significant metrics (p-value $< .05$) and the arrows represent the following behavior: risk-increasing (arrow up) and risk-decreasing (arrow down). Finally, the blank cells represent metrics that were removed from the model for being collinear. We discuss the results as follows.

Table 4.8: Results of the odds ratio analysis for the communication dynamics

System	Control Variables				Communication Dynamics Aspect									
	PS	PC	DC	DS	# Comments	# Users	# Contributors	# Core Devs	# Employees	OBTemp	MTBC	DL	TBCFC	TBLCM
elasticsearch	1.081	2.179 ↑			1.059	1.001	1.046	1.051		1.32 ↑	1.062	1.23 ↑	0.963	
netty		26.082 ↑			1.563	0.698	0.681	0.61		1.756		1.286	0.599	1.221
okhttp		8.506 ↑		0.522 ↓	0.804	1.118	1.243	1.037	0.967		1.078	1.109	0.942	
presto		5.561 ↑				1.179 ↑	0.849 ↓	0.958		0.862		1.052	1.082	1.01
rxjava				2.273 ↑		0.941	1.124	0.958		0.997	1.231	0.96	0.926	
security				2.603	0.408	1.156	2.056	1.091		1.704		1.074		0.669
exoplayer		2.826 ↑	0.542 ↓			1.195	1.377	0.256	1.181		1.28	0.587	1.081	0.592
all		3.724 ↑			0.905	1.03	1.072	1.145 ↑	0.947		1.135 ↑	1.108 ↑	1.013	

Confirmed observations. In the original study, Number of Users (# Users), Mean Time between Comments (MTBC), and Discussion Length (DL) presented a statistically significant increase in the risk of decay. In this replication, # Users continued its risk-increasing tendency in most projects.

PUC-Rio - Certificação Digital Nº 1912701/CA

However, we only observed a statistically significant result for the Presto project. Thus, we manually analyzed a sub-set of pull requests from the Presto project to better understand its result.

Based on the aforementioned analysis, we identified some scenarios that may explain the risk-increasing tendency. First, we observed that pull requests involving controversial changes tend to also involve a higher number of users. In some cases, the controversy was related to the impact on design quality, which helps to explain the risk-increasing tendency. We observed this scenario, for example, in the pull request #11302[2] of Presto. This pull request involved 44 participants (including one bot) and ended up not being merged.

The Mean Time between Comments (MTBC) metric was statistically significant on only three projects in the preliminary study – Netty, OkHttp, and when we combined all data. In this study, this behavior was just preserved only when we combined all data. Finally, the Discussion Length (DL) metric showed a bigger risk-increasing tendency in this replication: from only the Netty project to Elasticsearch and all data combined. These results are expected as we believe that when a discussion is sparse and the reviewers take long time to reply, the developer may lose motivation to properly solve some problems.

PUC-Rio - Certificação Digital Nº 1912701/CA

> **Finding 2:** Discussions that take long time or that its comments are very sparse are related to the increase of design decay symptoms.

Controversial observations. Differently from the previous study, in this replication, the Number of Core Developers (# Core Devs) and Opened By Temporary (OBTemp) metrics presented a statistically significant risk-increasing tendency. First, # Core Devs when combining data from all projects. The OBTEMP was not significant in the past study. However, this result is expected, due the fact that temporary developers are often less experient on the code being developed. Moreover, the Number of Core Devs is a surprising result, but looking to our data, we see that most cases with decrease of degradation of symptoms there were no core developers acting in the pull request. However, these cases happened in early stages of the project in which the core developers are also usually doing development work and leave the contributors work more freely. This situation also explains the behavior of the Number of Contributors metric, which presented a risk-decreasing tendency on the Presto project.

[2]https://github.com/prestodb/presto/pull/11302

Finding 3: Pull requests opened by temporary developers have a higher chance of design decay. However, when these temporary users become contributors, a higher number of contributors in a pull request decreases the change of decay.

New observations. In this replication, we also introduced two new communication dynamics metrics: Time Between Creation and First Comment (TBCFC) and Time Between Last Comment and Merge (TBLCM). Nevertheless, we could not observe statistically significant results for any of the cases. Moreover, the tendency for both of them was not homogeneous across the different projects. Finally, we also observed multiple cases of collinearity, specially for the TBLCM metric.

4.4.3 Discussion Content and Decay

By following the same procedures of the previous research question, in $\mathbf{RQ}_3$ we aim to understand the influence of the discussion content over the design decay. The results of this analysis are summarized in Table 4.9, in which we also used the odds ratio technique to explain the effect of this social aspect over the design decay. Similarly to Table 4.8, the grey cells represent the statistically significant metrics (p-value $< .05$), and the arrows represent the risk-increasing (arrow up) and risk-decreasing (arrow down) effects. Finally, the blank cells are the metrics missing due to collinearity.

Table 4.9: Results of the odds ratio analysis for the discussion content

System	Control Variables				Discussion Content Aspect							
	PS	PC	DC	DS	# Snippets	# Words	# Words per Comment	Snippet Size	Design	Refactoring	Design Density	Ref. Density
elasticsearch		2.97 ↑	0.65 ↓		0.634		1.046	1.594		7.98 ↑	1.072	0.142 ↓
netty	8.029 ↑						0.93	12.599			1.363	1.029
okhttp	0.614 ↓	10.074 ↑			2.925	0.998		0.353			0.991	1.019
presto		5.7 ↑			1.409		0,838 ↓	0.737		2.47	1.006	0.447
rxjava				2.263 ↑	0.143		0,775 ↓	6.796		35.494 ↑	1.045	0.036 ↓
security	3.544 ↑							0.894		441.727	0.964	0.001
exoplayer		2.705 ↑	0.591 ↓		1.03						0.915	1.19
all	2.387 ↑				0.977	12.503 ↑	0.088 ↓	1.006	1.43	0.916	0.727	1.119

Confirmed observations. By performing a closest comparison of our results with that obtained by Barbosa et al [6], we observed that the metrics Number of Words per Comment in Discussion (NWPCD) and Number of Words in Discussion (NWD) remained statistically significant. More specifically, the metric NWPCD presented statistically significant for the Presto system and remained statistically significant for the RxJava system, and when we combine the data of all systems. In the case of the metric NWD, it remained statistically significant only when we for all the data combined. In addition, in this replication, both metrics NWD and NWPCD remained their risk tendency effect on design decay, risk-increasing, and risk-decreasing effect, respectively.

PUC-Rio - Certificação Digital Nº 1912701/CA

To exemplify the metric NWPCD, we can observe the pull request #1409 from the RxJava project. The goal of this pull request was to "Avoiding OperatorObserveOn from calling subscriber.onNext(..)". This pull request involved three developers and one bot. The discussion on this pull request presented 26 comments, with an NWPCD of 9, indicating a high number of words scattered across comments. In addition, such comments were all made by the three developers involved in this pull request, indicating a high engagement of the developers in providing useful pieces of information to reduce the design decay in the system. Conversely, as mentioned by [6] and observed in the pull request #1388 from the OkHttp project, the metric NWD indicate an increase on design decay when a discussion has a high number of comments and a high number of words, however, these words are concentrated only in a few comments.

Finding 4: Our results confirm the main findings of Barbosa et al [6] with respect to the influence of discussion content on design decay. Discussions with a high number of words, when it is not accompanied by a high number of words per comment, work as indicators of design decay increase.

Controversial and new observations. We did not observe any controversial results in this matter compared to the original work. By addicting four new metrics, Number of Design Keywords (NDK), Number of Refactoring Keywords (NRK), the density of design comments (Design Density), and Density of Refactoring Keywords (Ref. Density), in the discussion content dimension, new observations have emerged. In this case, one out of two metrics related to the presence of a specific keyword on the title or comments of pull requests, i.e., NRK, has a risk-increasing tendency. Moreover, the metric NRK was statistically significant for the Elasticsearch and RxJava systems. Regarding the two metrics related to the density of specific keywords per comment, only the metric Density of Refactoring Keywords has presented a risk-decreasing tendency.

In other words, the metric NDK indicated that the higher changes that contain the 'refactoring' keyword, the bigger would be the risk of design decay. However, the metric Density of Refactoring Keywords shows us that the higher the mean of 'refactoring' comments, the smaller the chances of design decay. Moreover, these two metrics do not conflict, but they are complementary as indicators of design decay. We observed that the metric NRK may indicate an increase in design decay when: (1) the developers are implementing a new feature or enhancing an existing feature; and (2) due to the overload of the "refactoring" term, which is also a common reason for indicating any structural

PUC-Rio - Certificação Digital Nº 1912701/CA

change, in this sense, the developers may claim a refactoring was being made when the change actually consisted of a feature improvement.

To exemplify the cases, we observe the pull request #797 from the RxJava project, regarding the case (1) aforementioned. The goal of this pull request was to "Scheduler Outer/Inner", more specifically, the developer was intended "to simplify scheduling and make it easier to do the right thing". Regarding case (2) we can observe in the pull request #2928 from the RxJava project, that the refactoring term was applied to indicate a feature improvement: "It seems the merge is quite sensitive to method structuring: just by refactoring the scalar emission path I got some notable performance back [...]"

Finding 5: We found that discussions with a high number of refactoring keywords, but when not weighted by the number of comments, has a higher risk of design decay. This behavior is seen also in the NWD and NWPCD metrics.

4.4.4 Organizational Dynamics and Decay

Besides investigating communication dynamics and discussion content, we also analyzed the impact of organization dynamics on design decay in this replication. Table 4.10 shows the results for four organization dynamics metrics, namely Team Size, Gender (which is divided on Number of Male Developers (# Male Devs) and Number of Female Developers (# Female Devs)), Number of Newcomers (# Newcomers), and Number of Leaving Developers (# Leaving Devs).

Team size and gender not related to design decay. Regarding team size, it is possible to observe in Table 4.10 that collinearity affected all cases (blank cells), meaning that this metric is behaving in the same way as the another, so we may not be certain about its use for predicting design decay. Moreover, we also observed that gender seems not to be suitable for predicting design decay. Such a result is explained by the fact that most contributors gender that could be easily recognizable are male (95%). In fact, gender has been recognized as a barrier for the computer science field [29]. As a result, the majority of contributors are male or prefer to not reveal their gender.

Finding 6: The Team Size did not show any relation to design decay, as well the Gender Diversity.

Number of newcomers as a risk-decreasing metric. Contrary to the expectation, we observed that the # Newcomers metric may be related

PUC-Rio - Certificação Digital Nº 1912701/CA

to reduction of design decay. Results for Elasticsearch, Netty, RxJava, and a combination of all projects showed a risk-decreasing tendency. This means that the higher the number of newcomers the smaller is the incidence of design decay. Moreover, Elasticsearch was the only project that presented a statistically significant result.

We conjecture that the risk-decreasing tendency can be explained by the fact that a higher number of newcomers results in a higher number of contributors. Thus, with more contributors, the workload of the project can be better distributed. This may help each contributor to perform high-quality work, both in development and code review tasks. Another factor related to newcomers is the fact that they often face several barriers before starting to contribute [67]. Therefore, we believe that only newcomers who are able to devote more effort and time to the project manager become contributors, which may result in higher design quality. Finally, we also believe that core developers end up reviewing newcomers' contributions with higher attention and with stricter criteria. As a result, only pull requests of high quality and relevance produced by newcomers are accepted in the projects.

Finding 7: Surprisingly, the number of newcomers that performing code changes do not lead to a risk-increasing effect on design decay.

Positive impact of number of leaving developers. Another surprising result was regarding the # Leaving Devs metric. We observed a statistically significant risk-decreasing tendency in three projects: Elasticsearch, Presto, and ExoPlayer. This result is surprising because it indicates that the higher the number of developers leaving the project, the lower the incidence of design decay. We conjecture that a high rate of developers leaving the project means that the project usually receives contributions from diverse people, even if for short periods. Such a diversity of contributions can be beneficial in maintaining the quality of the design.

Finding 8: When we see a high number of developers leaving the organization, we also observe an decrease in the design decay symptoms.

In summary, in this section we discussed the impact of organizational dynamics on the incidence of design decay. We were able to identify two metrics (# Newcomers and # Leaving Devs) that can help to predict future design problems. Nevertheless, such metrics should be further investigated through more quantitative and qualitative analyses in order to better understand whether and why they are relevant to design quality.

PUC-Rio - Certificação Digital Nº 1912701/CA

Table 4.10: Results of the odds ratio analysis for the organizational dynamics

System	Control Variables				Organizational Dynamics Aspect				
	PS	PC	DC	DS	Team Size	# Male Devs	# Female Devs	# Newcomers	# Leaving Devs
elasticsearch		3.195↑	0.623 ↓			1.096		0.864 ↓	0.837 ↓
netty		13.885 ↑				1.155		0.382	0.424
okhttp		8.461 ↑		0.527 ↓		1.024		1.039	0.89
presto				2.292 ↑		1.033			0.907↓
rxjava		11.337 ↑				1.039		0.808	0.91
security				2.484		0.55			0.358
exoplayer		2.578 ↑	0.558 ↓			0.747			0.638↓
all		3.92 ↑				1.058		0.941	1.026

4.4.5 All Metrics and Aspects

Since the original work did this analysis, in order to understand if different factors would emerge from different situations, we also think this replication needs to address them as well. Table 4.11 shows the results of the Multiple Logistic Regression when we combined all projects data and analyzed all social aspects in only one model.

Table 4.11: Results of the odds ratio analysis with all social aspects together for all project data

Metrics	
Control variables	
Patch Size	
Diff Size	
Diff Complexity	
Patch Churn	1.233↑
Communication Dynamics	
Number of Users	1.061
Number of Contributors	0.917
Number of Core Developers	0.968
Opened By: Temporaries	0.966
Number of Comments	
Mean Time Between Comments	
Discussion Duration	1.286↑
Time Between Opened and First Comment	1.097↑
Time Between Last Comment and Closing	0.921↓
Discussion Content	
Number of Snippets in Discussion	
Snippet Size	0.961
Mean Snippet Size	
Number of Words in Discussion	15.098↑
Number of Words per Comment in Discussion	0.082↓
Has Design Keyword	1.153
Has Refactoring Keyword	1.031
Density of Design Comments	0.917
Density of Refactoring Comments	0.978
Organizational Dynamics	
Team Size	0.795↓
Number of Male Developers	
Number of Female Developers	1.042
Number of Newcomers	
Number of Developers Leaving	0.949

Confirmed observations. We observed that the control variables were collinear to each other, meaning that in a large scale analysis, only one could be necessary. Concerning the social metrics, we can see that Discussion Length, Number of Words in Discussion, and Number of Words per Comment in

PUC-Rio - Certificação Digital Nº 1912701/CA

Discussion kept their behavior compared to the original work. We think that this solidifies their place as best social metrics to understand design decay.

Controversial observations. Differently from the original work, the Communication Dynamics aspect, we see that the Number of Users, Number of Comments, and Number of Core Developers did not appear as statistical significant. We think that this needs to be fully addressed in study with a much larger dataset, since these metrics are having different behaviors in many different situations. It might be also that many other project-specific factors end up strongly influencing the results.

New observations. As new results, we found that the Team Size, Time Between Creation and First Comment, and Time Between Last Comment and Merge appeared as statistically significant, different from results from RQ2 and RQ4. We think these are metrics that also need further analysis to understand what is their real power to relate to design decay symptoms.

Finding 9: Time Between Creation and First Comment, and Time Between Last Comment and Merge showed promising results on their relation with design decay symptoms and should be considered in future work.

PUC-Rio - Certificação Digital Nº 1912701/CA

4.5 Threats to Validity

Construct and Internal Validity. Our work used the DesignateJava tool to detect degradation symptoms and characterize changes. Therefore, while several works [60, 4, 48, 77, 6] have previously used this tool, the types of symptoms detected by the tool and aspects such as its precision and recall may introduce some bias to the results of this work. Nevertheless, the symptoms detected by the tool have been investigated in previous works [48, 60, 61, 6]. A previous work [62] has also indicated that DesignateJava has a precision of 96% and a recall of 99%. In order to confirm some of the results of the Designite Tool, we validated the cases where the design symptoms showed improving (the design decay symptoms went lower) with self-affirmed refactorings, that are known for improve the quality of the source code.

This study's methodology may also not fully represent all possible aspects that could influence design decay. Besides using social aspects, metrics, and tools previously studied by previous works, we aimed to mitigate this threat by increasing the number of projects, adding a new social aspect, making our statistical analysis stricter, and increasing the number of metrics analysed.

Conclusion and External Validity. In order to mitigate potential problems with external validity, we performed our analyses carefully and attentively. In the descriptive analysis, some of our authors contributed to reviewing and further refining the results. With respect to the statistical analysis, we did not use a normal distribution due to the skewness of the data, to avoid mismatch between the statistical method and the dataset. Thus, we instead used the Wilcoxon Rank Sum Test [80] on RQ_1 – which is non-parametric.

Together with the Cliff's Delta, we could strict the analysis made on RQ_1. For the regression analysis, we performed additional transformations ($\log_2$ and x^3) to reduce the skewness of the metrics. Along with this, we also removed the predictors with pair-wise correlations above 0.7 – this was done in order to avoid the multicollinearity of predictors from potentially affecting the results of the multiple regression model [14]. Furthermore, in order to ensure normality, we normalized the continuous predictors in the model. Finally, we also controlled some of the factors that could affect our outcomes through the usage of control variables, in the regression models (see Section 4.3.2). In this work, due to previously-mentioned constraints, we limited our investigation of design symptoms to only systems developed in Java. Due to this, our results might have bias towards the code structure of Java-based systems – however, this threat can also be mitigated by the fact that Java is one of the most popular programming languages, both in the industry and in academia.

4.6 Conclusion and Future Work

This paper reported a replication of a previous work by Barbosa et al [6], increasing the number of projects, assessing a wide number of metrics in the social aspects analyzed on previous work, and adding a new social aspects.

When doing this replication, we found several bottlenecks on the data extraction step. First, the design decay symptoms collection using the Designite tool [60] needed a high-end server to run. Second, there were many mining perils [30] that we had to make an avoidance strategies on the scripts. Finally, the data from the GitHub API took many weeks to be collected, since we are limited to 5000 calls per hour by the API.

Concerning our study results, we had mixed outcomes. By applying a rigorous approach on the analysis, when compared to the previous study, made our results diverge, in different levels, from the original work. On the first research question, the results went from almost 100% of the metrics being statistically significant to 4 to 7 metrics in each project, if we did

PUC-Rio - Certificação Digital Nº 1912701/CA

not consider the elasticsearch, that maintained his results. However, from our second research question to fourth, we had results that remained the same, others that were controversial, and also new observations.

The findings that we reached on this replication paper were: (i) different social metrics are able to differentiate impactful and unimpactful pull requests in different projects; (ii) number of users, mean time between comments, and discussion length proved that they are the best metrics from the communication dynamics aspects to indicate increase on design decay symptoms; (iii) the number of core developers need further analysis to really understand its behavior; (iv) regarding the discussion content, metrics related to the number of words in discussion and refactoring keywords are good indicators of increase in design decay symptom, while the weighted version of these metrics (NWPCD and Refactoring Keywords Density) showed the opposite effect, being the best indicators of decrease of the design decay symptoms; finally, (vi) in the organizational dynamics, the new social aspects analysed in this study, we were able to see that this aspects is related to decrease on the design decay symptoms, specifically the number of newcomers and number of leaving developers.

Regarding the future work, we can see many approaches that can be taken from here. First, we need to apply a qualitative study over our findings, to understand the point of view of the real developers behind these social metrics. With their opinions, we will be finally able to create real design-aware social coding guidelines to open-source communities. Then, we want to create a machine learning model to act as a design decay detector using only social metrics aspects. By doing that, we want to create a GitHub Bot to react over social behaviors on issues and pull requests of the projects.

PUC-Rio - Certificação Digital Nº 1912701/CA

4.7 Summary

This chapter aimed at replicating the study made in Chapter 3. One of the goals of the replication is to overcome some of the weaknesses that affect our experimental design (more detail in Chapter 4). We also aim to validate the results presented here in a larger dataset (5 vs. 7 systems). Finally, we aim to completely address our second research problem – *"There little knowledge on the influence of communication dynamics, discussion content, and organizational dynamics on design decay"* – by investigating the *organizational dynamics* aspect together with the aspects investigated in Chapter 3.

Our results reveal that: (i) different social metrics work as indicators of design change in different contexts; (ii) Number of Users, Meant Time Between Comments, and Discussion Length showed their relevance as design decay

indicators; (iii) Number of Refactoring Keywords and Refactoring Keyword Density presented promising results in relation to design decay symptoms; (iv) the discussion content metrics Number of Words in Discussion and Number of Words Per Comment in Discussion stood out as the best social metrics to identify design decay; and, finally, (v) When analyzing multiple social aspects and multiple projects at the same times, different metrics emerge as indicators of both increase and decrease of design decay symptoms. In the next chapter, we revisit the main Master's dissertation contributions and present new challenges and opportunities for improvement and future work that have emerged along the studies conducted in the context of this Master's dissertation.

PUC-Rio - Certificação Digital Nº 1912701/CA

5
Final Conclusions

Design decay is harmful to software maintenance and evolution. However, metrics related to source code may often not be enough to indicate it early, as they can only be obtained once the code change is concluded. Aims to avoid design decay, we could explore additional information surrounding the software development activities, such as conversations associated with issues and pull requests, even before the code change is merged into the main branch. These conversations enable us to identify or infer many social aspects, which may, in turn, contributes to indicate an increase or decrease in design decay symptoms.

Since social aspects have different dimensions, it is important to investigate these dimensions individually and as a group. Our goal was to understand what is the role of each of them and also combining their strengths. So we can present to social communities on collaborative coding environments insights and follow-ups in how they should behave to avoid the perils of design decay.

This dissertation aims to reveal the social aspects possibly related to the design decay in collaborative software communities. First, we analyze the relationship between two social aspects: communication dynamics and discussion content over the design decay symptoms in five open-source projects. In addition, we also evaluate the role of more social measures associated with the organizational dynamics aspect. In the end, we explored additional social metrics in the replication study. In summary, the main contributions and their possible impact on collaborative software communities are described as follows.

Contribution 1: *A Software Framework to Collect and Evaluate Social Metrics.* In this work, we designed and implemented 22 social metrics (Chapter 2), based on the mapping study of Wiese *et al.* [81]. We only implemented the ones that apply to the scope of a single pull request at the time. To support these metrics, we also designed and implemented a framework that collects the data from the GitHub API as a main source to the metrics. Moreover, this framework and the metrics are implemented with the Python language, and their implementation can be found in our replication package [7].

Contribution 2: *Set of Insights about Social Aspects on Design Decay.* There was little understanding about how social aspects related to design decay prior to this dissertation. Our goal was to assess these metrics in a preliminary

PUC-Rio - Certificação Digital Nº 1912701/CA

study (see Chapter 3) to investigate if it was possible for social metrics to be able to indicate design quality changes of the code being produced. Moreover, in this same study, we investigated the odds ratio of each social metric to indicate a risk of increase or decrease the design decay symptoms in a pull request. Finally, we conducted a replication study (see Chapter 4), to validate and mitigate the limitations of the preliminary study.

Contribution 3: *Statistical Models to Evaluate Social Aspects.* We assess the social metrics and design decay by two statistical models, and we also present these models as contributions for this work. By doing this, we allow full replication of our work and also permit the study to be assessed in future work.

Contribution 4: *Social Aspects vs Design Decay.* Finally, we leave as contribution our set of empirical findings. They can be used for social collaboration communities to create their own guidelines.

5.1 Implications and Future Work

This dissertation provides several findings which lead to implications for researchers, tool developers, and practitioners. Those implications are discussed as follows.

Social metrics are able to complement current design decay tools. Our findings say that many social metrics are able to differentiate between pull requests that changed the design on some level, from the ones that the design remained the same. Moreover, they also assert that social metrics, grouped by social aspects, are able to indicate both increase and decrease of design decay symptoms. These findings reveal that social metrics have a saying in design quality concerns. There are several tools [60, 73] that try to indicate whether the design has decayed or not, based on many metrics, e.g., code attributes, code smells. However, those tools alone may not be the best detectors, since our study indicated that social metrics are able to complement these current strategies. Finally, as our study suggests, those metrics should be considered when creating a design decay detector tool.

Social Aspects as early detectors. Differently from current tools and strategies, the social aspects are data that is present since early on the code development, i.e., first, an issue is created, and then a pull request is submitted (which contains the source code). In addition, the discussions relating to the code being developed, even in pull requests, happen before a new committed is added to the pull request, i.e., developers discuss what they intend to do before start coding. With that in mind, one can argue that the information

PUC-Rio - Certificação Digital Nº 1912701/CA

that wraps the social aspects happens prior to the code being made. Thus, with our findings, tools can be created to predict if a discussion is leading to an increase in the design decay symptoms or a decreasing of these symptoms.

As future work, we plan to assess the relation of cause and effect of social aspects and design decay. Moreover, our work also enable one to build design decay predictors inside this social communities; Our two studies confirm that different social metrics work as indicators of design change in different contexts: (i) `Number of Users`, `Meant Time Between Comments`, and `Discussion Length` showed their relevance as design decay indicators; (ii) the discussion content metrics `Number of Words in Discussion` and `Number of Words Per Comment in Discussion` stood out as the best social metrics to identify design decay; (iii) `Number of Refactoring Keywords` and `Refactoring Keyword Density` presented promising results in relation to design decay symptoms; (iv) The presence of newcomers and the number of developers leaving the organization showed their importance in the impact on the design of the system; (v) `Gender Diversity` did not present any relation to design decay, which is an interesting finding for gender studies; and (vi) When analyzing multiple social aspects and multiple projects at the same times different metrics emerge as indicators of both increase and decrease of design decay symptoms.

PUC-Rio - Certificação Digital Nº 1912701/CA

Bibliography

[1] AHMED, I.; MANNAN, U. A.; GOPINATH, R. ; JENSEN, C.. **An empirical study of design degradation: How software projects get worse over time**. In: PROCEEDINGS OF THE 10TH INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT (ESEM), p. 1–10. IEEE, 2015.

[2] ALOMAR, E.; MKAOUER, M. W. ; OUNI, A.. **Can refactoring be self-affirmed? an exploratory study on how developers document their refactoring activities in commit messages**. In: PROCEEDINGS OF THE 3RD INTERNATIONAL WORKSHOP ON REFACTORING (IWOR), p. 51–58. IEEE, 2019.

[3] ALOMAR, E. A.; MKAOUER, M. W. ; OUNI, A.. **Toward the automatic classification of self-affirmed refactoring**. Journal of Systems and Software, 171:110821, 2021.

[4] ALENEZI, M.; ZAROUR, M.. **An empirical study of bad smells during software evolution using designite tool**. i-Manager's Journal on Software Engineering, 12(4):12, 2018.

[5] BACCHELLI, A.; BIRD, C.. **Expectations, outcomes, and challenges of modern code review**. In: PROCEEDINGS OF THE 35TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 712–721. IEEE, 2013.

[6] BARBOSA, C.; UCHÔA, A.; COUTINHO, D.; FALCÃO, F.; BRITO, H.; AMARAL, G.; SOARES, V.; GARCIA, A.; FONSECA, B.; RIBEIRO, M. ; OTHERS. **Revealing the social aspects of design decay: A retrospective study of pull requests**. In: PROCEEDINGS OF THE 34TH BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING (SBES), p. 364–373, 2020.

[7] BARBOSA, C.. **Replication package**, 2020. Available at: `https://guriosam.github.io/revealing_social_aspects_of_design_decay/`.

PUC-Rio - Certificação Digital Nº 1912701/CA

[8] BATISTA, N. A.; BRANDÃO, M. A.; ALVES, G. B.; DA SILVA, A. P. C. ; MORO, M. M.. **Collaboration strength metrics and analyses on github**. In: 16TH WI, p. 170–178, 2017.

[9] BETTENBURG, N.; HASSAN, A. E.. **Studying the impact of social interactions on software quality**. Emp. Softw. Eng. (ESE), 18(2):375–431, 2013.

[10] BIRD, C.; GOURLEY, A.; DEVANBU, P.; SWAMINATHAN, A. ; HSU, G.. **Open borders? immigration in open source projects**. In: PROCEEDINGS OF THE 14TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), p. 6–6, 2007.

[11] BIRD, C.; NAGAPPAN, N.; MURPHY, B.; GALL, H. ; DEVANBU, P.. **Don't touch my code! examining the effects of ownership on software quality**. In: PROCEEDINGS OF THE 13TH ACM JOINT EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING (FSE), p. 4–14, 2011.

[12] BRUNET, J.; MURPHY, G. C.; TERRA, R.; FIGUEIREDO, J. ; SEREY, D.. **Do developers discuss design?** In: PROCEEDINGS OF THE 11TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), p. 340–343. ACM, 2014.

[13] DABBISH, L.; STUART, C.; TSAY, J. ; HERBSLEB, J.. **Social coding in github: transparency and collaboration in an open software repository**. In: PROCEEDINGS OF THE 15TH CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK AND SOCIAL COMPUTING (CSCW), p. 1277–1286, 2012.

[14] DORMANN, C. F.; ELITH, J.; BACHER, S.; BUCHMANN, C.; CARL, G.; CARRÉ, G.; MARQUÉZ, J. R. G.; GRUBER, B.; LAFOURCADE, B.; LEITÃO, P. J. ; OTHERS. **Collinearity: a review of methods to deal with it and a simulation study evaluating their performance**. Ecography, 36(1):27–46, 2013.

[15] EDWARDS, A. W.. **The measure of association in a 2× 2 table**. J. Royal Stat. Soc., 126(1):109–114, 1963.

[16] EICK, S. G.; GRAVES, T. L.; KARR, A. F.; MARRON, J. S. ; MOCKUS, A.. **Does code decay? assessing the evidence from change management data**. IEEE Transactions on Software Engineering, 27(1):1–12, 2001.

PUC-Rio - Certificação Digital Nº 1912701/CA

[17] EPOSHI, A.; OIZUMI, W.; GARCIA, A.; SOUSA, L.; OLIVEIRA, R. ; OLIVEIRA, A.. **Removal of design problems through refactorings: are we looking at the right symptoms?** In: PROCEEDINGS OF THE 27TH INTERNATIONAL CONFERENCE ON PROGRAM COMPREHENSION (ICPC), p. 148–153. IEEE Press, 2019.

[18] FALCÃO, F.; BARBOSA, C.; FONSECA, B.; GARCIA, A.; RIBEIRO, M. ; GHEYI, R.. **On relating technical, social factors, and the introduction of bugs**. In: PROCEEDINGS OF THE 27TH INTERNATIONAL CONFERENCE ON SOFTWARE ANALYSIS, EVOLUTION, AND REENGINEERING (SANER), p. 378–388, 2020.

[19] FERREIRA, I.; FERNANDES, E.; CEDRIM, D.; UCHÔA, A.; BIBIANO, A. C.; GARCIA, A.; CORREIA, J. L.; SANTOS, F.; NUNES, G.; BARBOSA, C. ; OTHERS. **The buggy side of code refactoring: Understanding the relationship between refactorings and bugs**. In: PROCEEDINGS OF THE 40TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: COMPANION PROCEEEDINGS, p. 406–407, 2018.

[20] FOWLER, M.. **Refactoring**. Addison-Wesley Professional, 1999.

[21] GIUFFRIDA, R.; DITTRICH, Y.. **Empirical studies on the use of social software in global software development–a systematic mapping study**. Information and Software Technology, 55(7):1143–1164, 2013.

[22] GOUSIOS, G.; PINZGER, M. ; DEURSEN, A. V.. **An exploratory study of the pull-based software development model**. In: PROCEEDINGS OF THE 36TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 345–355, 2014.

[23] GOUSIOS, G.; ZAIDMAN, A.; STOREY, M.-A. ; VAN DEURSEN, A.. **Work practices and challenges in pull-based development: the integrator's perspective**. In: PROCEEDINGS OF THE 37TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), volumen 1, p. 358–368. IEEE, 2015.

[24] GRISSOM, R. J.; KIM, J. J.. **Effect sizes for research: A broad practical approach.** Lawrence Erlbaum Associates Publishers, 2005.

[25] GUISAN, A.; ZIMMERMANN, N. E.. **Predictive habitat distribution models in ecology**. Ecological modelling, 135(2-3):147–186, 2000.

PUC-Rio - Certificação Digital Nº 1912701/CA

[26] HASSAN, A. E.. **Predicting faults using the complexity of code changes**. In: PROCEEDINGS OF THE 31ST INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 78–88, 2009.

[27] HOZANO, M.; GARCIA, A.; FONSECA, B. ; COSTA, E.. **Are you smelling it? investigating how similar developers detect code smells**. Inf. Softw. Technol. (IST), 93:130–146, 2018.

[28] IBRAHIM, W. M.; BETTENBURG, N.; SHIHAB, E.; ADAMS, B. ; HASSAN, A. E.. **Should i contribute to this discussion?** In: PROCEEDINGS OF THE 7TH IEEE WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), p. 181–190. IEEE, 2010.

[29] MAIN, J. B.; SCHIMPF, C.. **The underrepresentation of women in computing fields: A synthesis of literature using a life course perspective**. IEEE Transactions on Education, 60(4):296–304, 2017.

[30] KALLIAMVAKOU, E.; GOUSIOS, G.; BLINCOE, K.; SINGER, L.; GERMAN, D. M. ; DAMIAN, D.. **The promises and perils of mining github**. In: PROCEEDINGS OF THE 11TH WORKING CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), p. 92–101, 2014.

[31] KALLIAMVAKOU, E.; GOUSIOS, G.; BLINCOE, K.; SINGER, L.; GERMAN, D. M. ; DAMIAN, D.. **An in-depth study of the promises and perils of mining github**. Empirical Software Engineering, 21(5):2035–2071, 2016.

[32] KEYES, J.. **Social software engineering: development and collaboration with social networking**. CRC Press, 2016.

[33] KHOMH, F.; DI PENTA, M.; GUÉHÉNEUC, Y.-G. ; ANTONIOL, G.. **An exploratory study of the impact of antipatterns on class change- and fault-proneness**. Emp. Softw. Eng. (ESE), 17(3):243–275, 2012.

[34] KONONENKO, O.; BAYSAL, O. ; GODFREY, M. W.. **Code review quality: how developers see it**. In: PROCEEDINGS OF THE 38TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 1028–1038. IEEE, 2016.

[35] LI, Z.; AVGERIOU, P. ; LIANG, P.. **A systematic mapping study on technical debt and its management**. Journal of Systems and Software, 101:193–220, 2015.

PUC-Rio - Certificação Digital Nº 1912701/CA

[36] MANNAN, U. A.; AHMED, I.; ALMURSHED, R. A. M.; DIG, D. ; JENSEN, C.. **Understanding code smells in android applications**. In: PROCEEDINGS OF THE 3RD INTERNATIONAL CONFERENCE ON MOBILE SOFTWARE ENGINEERING AND SYSTEMS (MOBILESOFT), p. 225–236. IEEE, 2016.

[37] MARTIN, R. C.; MARTIN, M.. **Agile Principles, Patterns, and Practices in C# (Robert C. Martin)**. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[38] MARTINS, J.; BEZERRA, C.; UCHÔA, A. ; GARCIA, A.. **Are code smell co-occurrences harmful to internal quality attributes? a mixed-method study**. In: PROCEEDINGS OF THE 34TH BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING (SBES), p. 1 – 10, 2020.

[39] MCDONALD, J. H.. **Handbook of biological statistics**, volumen 2. Sparky House Publishing, 2009.

[40] MCINTOSH, S.; KAMEI, Y.; ADAMS, B. ; HASSAN, A. E.. **An empirical study of the impact of modern code review practices on software quality**. Emp. Softw. Eng. (ESE), 21(5):2146–2189, 2016.

[41] MELLO, R.; OLIVEIRA, R.; SOUSA, L. ; GARCIA, A.. **Towards effective teams for the identification of code smells**. In: PROCEEDINGS OF THE 10TH INTERNATIONAL WORKSHOP ON COOPERATIVE AND HUMAN ASPECTS OF SOFTWARE ENGINEERING (CHASE), p. 62–65, 2017.

[42] MENEELY, A.; TEJEDA, A. C. R.; SPATES, B.; TRUDEAU, S.; NEUBERGER, D.; WHITLOCK, K.; KETANT, C. ; DAVIS, K.. **An empirical investigation of socio-technical code review metrics and security vulnerabilities**. In: 6TH SSE, p. 37–44, 2014.

[43] MORALES, R.; MCINTOSH, S. ; KHOMH, F.. **Do code review practices impact design quality? a case study of the qt, vtk, and itk projects**. In: PROCEEDINGS OF THE 22ND INTERNATIONAL CONFERENCE ON SOFTWARE ANALYSIS, EVOLUTION, AND REENGINEERING (SANER), p. 171–180. IEEE, 2015.

[44] MUND, S.. **Microsoft azure machine learning**. Packt Publishing Ltd, 2015.

[45] NAGAPPAN, N.; MURPHY, B. ; BASILI, V.. **The influence of organizational structure on software quality**. In: PROCEEDINGS OF THE

PUC-Rio - Certificação Digital Nº 1912701/CA

30TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 521–530. IEEE, 2008.

[46] OIZUMI, W.; GARCIA, A.; SOUSA, L.; CAFEO, B. ; ZHAO, Y.. **Code anomalies flock together: Exploring code anomaly agglomerations for locating design problems**. In: PROCEEDINGS OF THE 38TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), 2016.

[47] OIZUMI, W.; SOUSA, L.; OLIVEIRA, A.; GARCIA, A.; AGBACHI, A. B.; OLIVEIRA, R. ; LUCENA, C.. **On the identification of design problems in stinky code: experiences and tool support**. J. Braz. Comput. Soc., 24(1):13, 2018.

[48] OIZUMI, W.; SOUSA, L.; OLIVEIRA, A.; CARVALHO, L.; GARCIA, A.; COLANZI, T. ; OLIVEIRA, R.. **On the density and diversity of degradation symptoms in refactored classes: A multi-case study**. In: PROCEEDINGS OF THE 30TH INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING (ISSRE), p. 346–357. IEEE, 2019.

[49] OLIVA, G. A.; STEINMACHER, I.; WIESE, I. ; GEROSA, M. A.. **What can commit metadata tell us about design degradation?** In: PROCEEDINGS OF THE 13TH INTERNATIONAL WORKSHOP ON PRINCIPLES OF SOFTWARE EVOLUTION (IWPSE), p. 18–27, 2013.

[50] OLIVEIRA, A.; SOUSA, L.; OIZUMI, W. ; GARCIA, A.. **On the prioritization of design-relevant smelly elements: A mixed-method, multi-project study**. In: PROCEEDINGS OF THE 13TH BRAZILIAN SYMPOSIUM ON SOFTWARE COMPONENTS, ARCHITECTURES, AND REUSE (SBCARS), p. 83–92, 2019.

[51] PAIXAO, M.; MAIA, P. H.. **Rebasing in code review considered harmful: A large-scale empirical investigation**. In: PROCEEDINGS OF THE 19TH INTERNATIONAL WORKING CONFERENCE ON SOURCE CODE ANALYSIS AND MANIPULATION (SCAM), p. 45–55, 2020.

[52] PAIXÃO, M.; UCHÔA, A.; BIBIANO, A. C.; OLIVEIRA, D.; GARCIA, A.; KRINKE, J. ; ARVONIO, E.. **Behind the intents: An in-depth empirical study on software refactoring in modern code review**. In: PROCEEDINGS OF THE 17TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), 2020.

PUC-Rio - Certificação Digital Nº 1912701/CA

[53] PALOMBA, F.; PANICHELLA, A.; ZAIDMAN, A.; OLIVETO, R. ; DE LUCIA, A.. **The scent of a smell: An extensive comparison between textual and structural smells**. IEEE Trans. Softw. Eng. (TSE), 44(10):977–1000, 2017.

[54] PALOMBA, F.; BAVOTA, G.; DI PENTA, M.; FASANO, F.; OLIVETO, R. ; DE LUCIA, A.. **On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation**. Emp. Softw. Eng. (ESE), 23(3):1188–1221, 2018.

[55] PARNAS, D. L.. **Software aging**. In: PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 279–287. IEEE, 1994.

[56] PASCARELLA, L.; SPADINI, D.; PALOMBA, F. ; BACCHELLI, A.. **On the effect of code review on code smells**. In: PROCEEDINGS OF THE 27TH INTERNATIONAL CONFERENCE ON SOFTWARE ANALYSIS, EVOLUTION, AND REENGINEERING (SANER), p. 1–12, 2019.

[57] RAHMAN, M. M.; ROY, C. K.. **An insight into the pull requests of github**. In: PROCEEDINGS OF THE 11TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), p. 364–367, 2014.

[58] ROMANO, J.; KROMREY, J. D.; CORAGGIO, J.; SKOWRONEK, J. ; DEVINE, L.. **Exploring methods for evaluating group differences on the nsse and other surveys: Are the t-test and cohen'sd indices the most appropriate choices**. In: ANNUAL MEETING OF THE SOUTHERN ASSOCIATION FOR INSTITUTIONAL RESEARCH, p. 1–51. Citeseer, 2006.

[59] RUANGWAN, S.; THONGTANUNAM, P.; IHARA, A. ; MATSUMOTO, K.. **The impact of human factors on the participation decision of reviewers in modern code review**. Emp. Softw. Eng. (ESE), 24(2):973–1016, 2019.

[60] SHARMA, T.; MISHRA, P. ; TIWARI, R.. **Designite: A software design quality assessment tool**. In: PROCEEDINGS OF THE 1ST INTERNATIONAL WORKSHOP ON BRINGING ARCHITECTURAL DESIGN THINKING INTO DEVELOPERS' DAILY ACTIVITIES, p. 1–4, 2016.

[61] SHARMA, T.; SPINELLIS, D.. **A survey on software smells**. J. Syst. Softw. (JSS), 138:158–173, 2018.

PUC-Rio - Certificação Digital Nº 1912701/CA

[62] SHARMA, T.; SINGH, P. ; SPINELLIS, D.. **An empirical investigation on the relationship between design and architecture smells**. Emp. Softw. Eng. (ESE), 2020.

[63] SILVA, M. C. O.; VALENTE, M. T. ; TERRA, R.. **Does technical debt lead to the rejection of pull requests?** In: PROCEEDINGS OF THE 12TH BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS (SBSI), 2016.

[64] SOARES, V.; OLIVEIRA, A.; PEREIRA, J. A.; BIBANO, A. C.; GARCIA, A.; FARAH, P. R.; VERGILIO, S. R.; SCHOTS, M.; SILVA, C.; COUTINHO, D. ; OTHERS. **On the relation between complexity, explicitness, effectiveness of refactorings and non-functional concerns**. In: PROCEEDINGS OF THE 34TH BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING (SBES), p. 788–797, 2020.

[65] SOUSA, L.; OLIVEIRA, R.; GARCIA, A.; LEE, J.; CONTE, T.; OIZUMI, W.; DE MELLO, R.; LOPES, A.; VALENTIM, N.; OLIVEIRA, E. ; OTHERS. **How do software developers identify design problems? a qualitative analysis**. In: PROCEEDINGS OF THE 31ST BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING (SBES), p. 54–63, 2017.

[66] SOUSA, L.; OLIVEIRA, A.; OIZUMI, W.; BARBOSA, S.; GARCIA, A.; LEE, J.; KALINOWSKI, M.; DE MELLO, R.; FONSECA, B.; OLIVEIRA, R. ; OTHERS. **Identifying design problems in the source code: A grounded theory**. In: PROCEEDINGS OF THE 40TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 921–931, 2018.

[67] STEINMACHER, I.; WIESE, I. S.; CONTE, T.; GEROSA, M. A. ; REDMILES, D.. **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), p. 72–78, 2014.

[68] STOREY, M.-A.; ZAGALSKY, A.; FIGUEIRA FILHO, F.; SINGER, L. ; GERMAN, D. M.. **How social and communication channels shape and challenge a participatory culture in software development**. IEEE Transactions on Software Engineering, 43(2):185–204, 2016.

[69] TAMBURRI, D. A.; KRUCHTEN, P.; LAGO, P. ; VAN VLIET, H.. **What is social debt in software engineering?** In: PROCEEDINGS OF THE 6TH INTERNATIONAL WORKSHOP ON COOPERATIVE AND HUMAN ASPECTS OF SOFTWARE ENGINEERING (CHASE), p. 93–96. IEEE, 2013.

PUC-Rio - Certificação Digital Nº 1912701/CA

[70] TAMBURRI, D. A.; KRUCHTEN, P.; LAGO, P. ; VAN VLIET, H.. **Social debt in software engineering: insights from industry**. Journal of Internet Services and Applications, 6(1):1–17, 2015.

[71] TANG, A.; ALETI, A.; BURGE, J. ; VAN VLIET, H.. **What makes software design effective?** Design Studies, 31(6):614–640, 2010.

[72] TAYLOR, R. N.; VAN DER HOEK, A.. **Software design and architecture the once and future focus of software engineering**. In: FOSE'07, p. 226–243. IEEE, 2007.

[73] TSANTALIS, N.; KETKAR, A. ; DIG, D.. **Refactoringminer 2.0**. IEEE Transactions on Software Engineering, 2020.

[74] TSAY, J.; DABBISH, L. ; HERBSLEB, J.. **Influence of social and technical factors for evaluating contribution in github**. In: PROCEEDINGS OF THE 36TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING (ICSE), p. 356–366, 2014.

[75] TSAY, J.; DABBISH, L. ; HERBSLEB, J.. **Let's talk about it: evaluating contributions through discussion in github**. In: PROCEEDINGS OF THE 22ND ACM JOINT EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING (FSE), p. 144–154, 2014.

[76] UCHÔA, A.; BARBOSA, C.; COUTINHO, D.; OIZUMI, W.; ASSUNÇAO, W. K.; VERGILIO, S. R.; PEREIRA, J. A.; OLIVEIRA, A. ; GARCIA, A.. **Predicting design impactful changes in modern code review: A large-scale empirical study**. In: PROCEEDINGS OF THE 18TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), p. 1–12, 2021.

[77] UCHÔA, A.; BARBOSA, C.; OIZUMI, W.; BLENILIO, P.; LIMA, R.; GARCIA, A. ; BEZERRA, C.. **How does modern code review impact software design degradation? an in-depth empirical study**. In: PROCEEDINGS OF THE 36TH INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION (ICSME), p. 1 – 12, 2020.

[78] VASILESCU, B.; POSNETT, D.; RAY, B.; VAN DEN BRAND, M. G.; SEREBRENIK, A.; DEVANBU, P. ; FILKOV, V.. **Gender and tenure diversity in github teams**. In: PROCEEDINGS OF THE 33RD ANNUAL ACM CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, p. 3789–3798, 2015.

PUC-Rio - Certificação Digital Nº 1912701/CA

[79] VASILESCU, B.; SEREBRENIK, A. ; FILKOV, V.. **A data set for social diversity studies of github teams**. In: PROCEEDINGS OF THE 12TH INTERNATIONAL CONFERENCE ON MINING SOFTWARE REPOSITORIES (MSR), p. 514–517, 2015.

[80] WHITLEY, E.; BALL, J.. **Statistics review 6: Nonparametric methods**. Critical care, 6(6):509, 2002.

[81] WIESE, I. S.; CÔGO, F. R.; RÉ, R.; STEINMACHER, I. ; GEROSA, M. A.. **Social metrics included in prediction models on software engineering: a mapping study**. In: PROCEEDINGS OF THE 10TH INTERNATIONAL CONFERENCE ON PREDICTIVE MODELS AND DATA ANALYTICS IN SOFTWARE ENGINEERING (PROMISE), p. 72–81, 2014.

[82] WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M.; REGNELL, B. ; WESSLÉN, A.. **Experimentation in Software Engineering**. Springer Science & Business Media, 1st edition, 2012.

[83] YAMASHITA, A.; ZANONI, M.; FONTANA, F. A. ; WALTER, B.. **Inter-smell relations in industrial and open source systems: A replication and comparative analysis**. In: PROCEEDINGS OF THE 31ST INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION (ICSME), p. 121–130. IEEE, 2015.

[84] YU, Y.; YIN, G.; WANG, H. ; WANG, T.. **Exploring the patterns of social behavior in github**. In: PROCEEDINGS OF THE 1ST INTERNATIONAL WORKSHOP ON CROWD-BASED SOFTWARE DEVELOPMENT METHODS AND TECHNOLOGIES (CROWDSOFT), p. 31–36, 2014.

[85] ZANATY, F. E.; HIRAO, T.; MCINTOSH, S.; IHARA, A. ; MATSUMOTO, K.. **An empirical study of design discussions in code review**. In: PROCEEDINGS OF THE 12TH NTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT (ESEM), p. 11. ACM, 2018.

[86] DE MELLO, R.; UCHÔA, A.; OLIVEIRA, R.; OIZUMI, W.; SOUZA, J.; MENDES, K.; OLIVEIRA, D.; FONSECA, B. ; GARCIA, A.. **Do research and practice of code smell identification walk together? a social representations analysis**. In: PROCEEDINGS OF THE 13TH INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT (ESEM), p. 1–6, 2019.

PUC-Rio - Certificação Digital Nº 1912701/CA