Beyond the Code: Investigating the Effects of Pull Request Conversations on Design Decay

Caio Barbosa*, Anderson Uchôa†, Daniel Coutinho*, Wesley K. G. Assunção‡*, Anderson Oliveira*,
Alessandro Garcia*, Baldoino Fonseca§, Matheus Rabelo†, José Eric Coelho†,
Eryka Carvalho†, Henrique Santos†
*Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil
†Federal University of Ceará (UFC), Itapajé, Brazil
‡North Carolina State University (NCSU), Raleigh, USA
§Federal University of Alagoas (UFAL), Maceió, Brazil

Abstract*—Background: Code development is done collaboratively in platforms such as GitHub and GitLab, following a pull-based development model. In this model, developers actively communicate and share their knowledge through conversations. Pull request conversations are affected by social aspects such as *communication dynamics* among developers, *discussion content*, and *organizational dynamics*. Despite prior studies indicating that social aspects indeed impact software quality, it is still unknown to what extent social aspects influence design decay during software development. Thus, since social aspects are intertwined with design and implementation decisions, there is a need for investigating how social aspects contribute to avoiding, reducing, or accelerating design decay. Aims: To fill this gap, we performed a study aimed at investigating the effects of pull request conversation on design decay. Method: We investigated 10,746 pull request conversations from 11 open-source systems, characterizing in terms of three different social aspects: discussion content, organizational and communication dynamics. We considered 18 social metrics to these three social aspects, and analyzed how they associate with design decay. We used a statistical approach to assess which social metrics are able to discriminate between impactful and unimpactful pull requests. Then, we employed a multiple logistic regression model to evaluate the influence of each social metric per social aspect in the presence of each other on design decay. Finally, we also observed how the combination of all social metrics influences the design decay. Results: Our findings reveal that social metrics related to the size and duration of a discussion, the presence of design-related keywords, the team size, and gender diversity can be used to discriminate between design impactful and unimpactful pull requests. Organizational growth and gender diversity prevent decay. Each software community has its unique aspects that can be used to detect and prevent design decay. Also, design improvements can be accomplished by timely feedback, engaged communication, and design-oriented discussions with the contribution of multiple participants who provide significant comments. Conclusion: The social aspects related to pull request conversations are useful indicators of design decay.

***Index Terms*—pull request, social aspects, design decay**

I. Introduction

A key concern of developers involved in collaborative software development is to prevent or decrease the design decay of software systems [1]–[5]. In this context, some studies assess both technical and social factors related to design decay [6]–[8]. The design of a software system is considered decayed if it is more difficult to maintain and evolve than it should be [9]. Design decay can be measured by multiple symptoms, popularly known as code smells. Code smells are structures in a program that indicate poor design choices [10], leading to structural design decay [11]. For example, a *Long Method* is a smell which consists of a method that is too long and complex to understand and change [10], [11].

The identification and removal of code smells require a deep understanding of multiple parts of a system [6], [12]. Therefore, the lack of communication between developers may negatively affect such activities, and even experienced developers have difficulty in identifying code smells when working in isolation [13], [14]. Thus, an important factor that contributes to avoiding, reducing, or accelerating design decay during collaborative software development consists of developers continually communicating and sharing their knowledge along a code change [15]. In collaboratively platforms, such as GitHub and GitLab, the communication is promoted by the pull-based development [16], in which developers actively communicate and share their knowledge in pull request conversations. In such conversations, developers review each other's code, identify issues, and discuss ways to improve the code.

The effectiveness of communication depends on several social aspects [17],[1] which can be categorized into three main dimensions [18]: *communication dynamics*, the role of participants and the temporal aspects of the messages; *discussion content*, message exchanges and the content of each message; and *organizational dynamics*, characteristic of the team as a whole (*e.g.*, team size and gender diversity). Recent studies found the quality of code is influenced by social aspects related to *communication dynamics* and *discussion contents*, which have been linked to an increase in design decay symptoms [1]–[3], [19]–[21]. For instance, the content of comments around a code change (*e.g.,* the presence of code snippets and the number of words per comment) can indicate the quality of the discussion, and therefore contribute to either improve or deteriorate the structural design of a system [1], [2].

[1]Social aspects in software development refer to the ways in which individuals and teams interact with each other within a community.

To be published in the Proceedings 17th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2023 in New Orleans, Louisiana, United States.

Despite prior studies revealed the influence of certain social aspects on the quality of the source code [1]–[3], [19]–[21], one question is still open: *to what extent social aspects influence the design decay during software development?* Since social aspects are intertwined with design and implementation decisions, there is a need for investigating how social aspects contribute to avoiding, reducing, or accelerating design decay. By analyzing social aspects, individually or together, we can provide valuable insights for community managers and developers to enhance collaboration, communication, and overall software quality. By answering the open question, open-source communities and companies can monitor certain social aspects that may positively or negatively influence design decay.

Some studies have investigated how different social aspects can be used as sources of information and reasoning for the quality of a software's design [1], [19], [22], [23]. Existing work has shown that collaborative software development may not always be beneficial, as code review activities may lead to design decay in certain circumstances [2], [3], [24]–[26]. However, there is a lack of evidence on how different social aspects (*e.g.,* [1], [19], [22], [23]), individually or together, can be used as sources of information for reason about quality of a software design. Furthermore, little is known about how pull requests conversations are related to design decay [1].

The goal of this work is to understand the influence of social aspects surrounding pull request conversations on design decay. For that, we extracted 18 decay symptoms, using the Organic tool [12], from 21,492 commits related to merged pull requests of 11 open-source projects hosted on GitHub. For each commit in the dataset, we generated code smell data for the commit and its previous version, to compare the smells present on the 'before' and 'after' versions. By doing that, we were able to see how many smells were added or removed. We also mined 10,746 pull requests to generate a dataset of 18 social metrics for those systems. Finally, we 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.

Our findings reveal that social metrics related to the size and duration of a discussion, the presence of design-related keywords, the team size, and gender can be used to discriminate between design impactful and unimpactful pull requests. We also observed that organizational growth and gender diversity avoid decay. Furthermore, design improvements can be accomplished by timely feedback, engaged communication, and refactoring-oriented discussions with the contribution of multiple participants who provide significant comments. As a contribution, knowing the influence of social aspects on software design can help to derive guidelines for avoiding or mitigating design decay. This knowledge can also help to improve the current practices and develop a new generation of tools for assisting developers on becoming aware about the design impact during software development activities.

II. Background and Related Work

Social aspects in software development. Multiple studies analyzed the importance of social aspects in the software development process [22]. For instance, they collected social metrics to characterize and compute "social debt", aiming to observe how social problems can directly affect decision-making in software development. They concluded that social debt leads to more error-prone decisions. Tamburri *et al.* [23] performed a study on social aspects in industrial projects, providing the definition of common problems that can occur, together with potential mitigation to these problems. The main difference between these studies and our work is that they focus on how the social aspects of software engineering can cause on external software problems.

Wiese *et al.* [18] reported how social metrics were used in the past by software engineering researchers to build prediction models. They identified nine dimensions and three categories of metrics, some of which are heavily used in this work (see Section III-A). This classification was expanded by Barbosa *et al.* [1]. Our study expands it by investigating a broader set of social aspects, particularly by introducing new metrics related to *communication dynamics* and *discussion content*.

Social aspects versus software quality. When looking at the relationship between social aspects and software quality, Betternburg *et al.* [19] studied how social aspects can affect the quality of released pieces of software. They found a set of aspects with a statistically significant connection to defects: (i) low code churn (which was used as the baseline); (ii) low number of external resources (*e.g.*, links); and (iii) high variance of the time between comments in discussions.

Barbosa *et al.* [1] conducted the first study that focused on social aspects surrounding pull request conversations and design decay. They concluded that the social aspects – communication dynamics and discussion content – are able to differentiate between two types of pull requests (impactful and non-impactful) and that multiple social aspect metrics are related to both the increase and decrease of the design decay symptoms. Our study differs from the previous work by exploring a different set of smells at the method and class levels, that are detected earlier in software development [27], compared to the architectural and design levels considered in the related work. Additionally, we have considered how the social aspect related to organization dynamics impacts software quality. Moreover, different from [1], we evaluate how the social metrics impact the software quality, considering each software community in isolates.

Other studies have investigated the relationship between pull request discussions on GitHub and software quality. Soares *et al.* [28] performed a study to understand the relationship between refactorings and improvements to code quality, considering developer discussions. They identified that developers tend to discuss refactorings when they are performing more complex changes. Coutinho *et al.* [29] analyzed how different social and technical factors interacted to influence design decay. They found that the size of pull request discussions

could be used in a software system to differentiate modules whose design decayed at different levels. Also, they identified a relationship between large changes made by newbies and design decay. Our work differs from these two above due to the difference in scope of the analyses. While both related works both focus on the contents of the discussions, this work extends this analysis to the dynamics of the communication and the dynamics of the organization in which they took place.

III. Study Settings

A. Goal and Research Questions

To define the goal, research questions (RQs), and social aspects of our study, we followed the Goal-Question-Metric (GQM) template [30]. Our goal is: **analyze** social aspects; **for the purpose of** investigate their relationship with software decay; **with respect to** the pull request conversations and code decay symptoms; **from the viewpoint of** software developers when doing code contributions; **in the context** of 11 open-source systems. We have four RQs, as follows:

RQ_1: *What social aspects are more related to design decay?* In this work, we focus on three social aspects related to pull request conversations (see Table I). First, *communication dynamics* (8 metrics) is the aspect that encompasses the roles of participants in the discussion and temporal factors of messages exchanged in a conversation. For instance, it includes factors such as the number of messages sent, the response time of participants, or the role of the developers. Second, *discussion content* (6 metrics) is the aspect covering the content of the messages exchanged and the interactions between developers during the discussion. For instance, it includes factors such as number of words and the presence of technical terms. Third, *organizational dynamics* (4 metrics) covers the characteristics of the team as a whole, such as the team size and turnover.

For $\mathbf{RQ}_1$, we aim to investigate how these three social aspects observed in pull request conversations may reduce or amplify design decay. Only the first two of those three dimensions of social aspects were investigated in related works [1], [18], [19], [22], [23], [28], [29]. The answer for $\mathbf{RQ}_1$ will show us which **social metrics** are most prominent in distinguishing impactful and unimpactful pull requests. An impactful pull request is one that causes an *increase* or *decrease* in design decay symptoms upon merging. Conversely, unimpactful pull requests have no effect on design decay. Moreover, when there is an increase in design decay symptoms, it suggests that there are more smells in comparison to the parent commit, while a decrease in design decay symptoms implies that there are fewer smells. Finally, understanding which **social metrics** are most closely associated with impactful pull requests helps us to assess the relationship between these metrics and decay.

RQ_2: *To what extent do organizational dynamics influence design decay?* After distinguishing impactful and unimpactful pull requests via social metrics, in $\mathbf{RQ}_2$ we aim to understand what is the influence of *organizational dynamics* aspects on design decay. Thus, in $\mathbf{RQ}_2$, we investigate if social metrics from this aspect observed in pull request conversations relate to design decay. Our motivation to further explore this social aspect from pull request conversations, in terms of **social metrics**, is that companies and communities need a more holistic view of the importance of certain aspects of their team in collaborative software development.

RQ_3: *How do social aspects grouped by communities influence design decay?* In real settings, multiple social aspects emerge during activities of collaborative software development, due to the intrinsic nature of these activities [1], [14], [21]–[23]. Thus, $\mathbf{RQ}_3$ aims to investigate the impact of multiple social aspects grouped by communities on design decay. This question implies that social aspects can vary between different software repositories. However, it is likely that social aspects are relatively consistent within a software community, as developers share similar values, goals, and practices. Moreover, by analyzing information grouped by communities, we can gain insights into how these social aspects, represented by **social metrics**, are manifested and their impact on design quality.

RQ_4: *How do social aspects influence design decay?* Social aspects may vary among different companies or communities, influencing design decay. However, we also want to investigate how social aspects may influence software projects individually, since not always a project has a big community/companies surrounding it. In $\mathbf{RQ}_4$ we aim to understand how the social aspects influence design decay in a general way. Thus, we analyze the social aspects of *communication dynamics*, *discussion content*, and *organizational dynamics* aspects on design decay from the 11 projects. This analysis provides insights for future research on social aspects and design decay in software development, and it assists software communities of different sizes and in different stages of life.

The contribution by answering these RQs is to help companies and communities establish guidelines and best practices that promote effective communication, collaboration, and design quality. For instance, social aspects can be monitored along with discussions in pull requests by social metrics. This monitoring can indicate the proneness 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 to decrease the quality of the software being produced.

B. Study Steps and Procedures

In this section, we describe details of the steps of our study.

Step 1: Selecting open-source systems for analysis. For the selection of subject systems for our study, we defined four inclusion 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 in the past year; (iii) systems that are at least 5 years old, and are currently active; and (iv) Java-based systems, since there is a wide availability of static analysis tools and libraries that can automatically identify source code problems [12], [31], [32]. These criteria were selected in order to avoid known perils of

TABLE I
DIMENSIONS OF SOCIAL ASPECTS RELATED TO PULL REQUEST CONVERSATIONS INVESTIGATED IN OUR STUDY

Metrics	Description	Rationale
Communication Dynamics Dimension		
Number of Newbies	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)	
Discussion Size	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 Pull Request Opening 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.
Discussion Content Dimension		
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 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 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 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 Keywords	Mean of design-related keywords per comments	The higher the mean of 'design' comments, the smaller the chances of decay happening
Density of Refactoring Keywords	Mean of refactoring-related keywords per comments	The higher the mean of 'refactoring' comments, the smaller the chances of decay happening
Organization Dynamics Dimension		
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 Diversity	Number 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 180 days	A bigger amount of newcomers can introduce less experience on the changes, leading to design decay
Number of Developers Inactive	Number of developers that previously contributed to the project but did not contribute on the past 180 days but are now participating in a pull request	More developers inactive can decrease the engagement of the community, leading to design decay

*For any GitHub Pull Request, we have three types of comments: (i) comments on the Pull Requests as a whole; (ii) comments on a specific line within the Pull Request (Review comment); and (iii) comments on a specific commit within the Pull Request (Commit comment). We use the first one, as we are only interested in pull request conversations.

TABLE II
SOFTWARE SYSTEMS INVESTIGATED IN THIS STUDY

Owner	System	Domain	# Commits	# PRs	Time span	≈LOC
Google	exoplayer	Library	15057	589	2014 - 2023	2.5m
	guava	Library	6025	562	2009 - 2023	1m
	gson	Library	1814	463	2008 - 2023	50.8k
	dagger	Dependency Injection	3856	1088	2012 - 2023	239.5k
	guice	Dependency Injection	2057	344	2006 - 2023	107.6k
Spring	spring-boot	Development Framework	42075	3345	2012 - 2023	785.4k
	spring-security	Framework Security	13432	1075	2004 - 2023	752.3k
Netflix	zuul	API Gateway	1435	713	2012 - 2023	49.3k
	eureka	Service Registry	1687	745	2001 - 2023	53.6k
	Hystrix	Fault Tolerance	2109	673	2012 - 2022	78.6k
	conductor	Microservices	3131	1149	2016 - 2023	89.7k

TABLE III
TYPES OF DECAY SYMPTOMS INVESTIGATED IN THIS STUDY

Class-level Symptoms
God Class, Class Data Should Be Private, Complex Class, Lazy Class, Refused Bequest, Spaghetti Code, Speculative Generality, Data Class, Brain Class
Method-level Symptoms
Feature Envy, Long Method, Long Parameter List, Message Chain, Dispersed Coupling, Intensive Coupling, Shotgun Surgery, Brain Method, Inflated Exception

mining software repositories [33], and are also used in related work [1]. Table II presents details of the 11 selected systems.

Step 2: Detecting the presence of design decay in commits. In this step, we used the Organic tool [12]. Organic is a static code analyzer that collects software metrics for code smell detection in methods and classes [12]. We detected a total of 18 types of code decay symptoms (*i.e.,* code smells) in the pull requests, which are presented in Table III.

For detecting these code decay symptoms, we only considered the commits that merged the pull requests. More precisely, for each pull request PR_i, we downloaded a snapshot of the commit C_i that merged the PR_i. Next, we calculated the difference between C_i and C_{i-1}, C_{i-1} being the parent commit of C_i, in order to guarantee that the introduced decay belonged only to C_i. By doing this, we avoid the rebase effect [34], [35], 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 [36].

Step 3: Computing design code decay in terms of density and diversity of symptoms. The density and diversity of symptoms are indicators of progressive design decay, as observed in previous studies [2], [3], [6]. The density is measured by calculating the number of smells found within a commit, *e.g.,* if a commit has 3 *Long Methods* and 1 *God Class*, the density of that commit is 4. On the other hand, the diversity is calculated by the number of different smell types found within a commit, *e.g.,* if a commit has 3 *Long Methods* and 1 *God Class*, the diversity is 2, as there are two different smell types. Previous studies observed that 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, this positive value represents a worsening of the software design. Similarly, a negative difference in

the density (or diversity) 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 (or diversity) 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. With the number of smells added or removed, we compute four indicators related to the density and diversity at class and method levels, for around 11,600 merged pull requests. The replication package [36] includes all the computed indicators.

Step 4: Calculating metrics to measure different social aspects. We have grouped each metric into three dimensions, namely *communication dynamics*, *discussion content*, and *organizational dynamics*, as presented in Table I and described in RQ_1. These three dimensions indicate social factors that can either facilitate or hinder structural design change. The computation of the social metrics was performed in three steps: (i) collect the 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 (see the column 'description' in Table I). We collected 18 metrics in total, which are used to measure social aspects affecting and interfering with the code development artifacts [37]. We considered those metrics as independent variables to measure certain social aspects.

Step 5: Assessing the relationship between social aspects and impactful pull requests. To decide whether a social metric is *statistically different* for impactful pull requests, when compared to the unimpactful ones, we use the *Wilcoxon Rank Sum Test* [38]. The test was conducted using the customary 0.05 significance level (*i.e.,* 95% of confidence). We selected this non-parametric test based on the observation that social metrics used in our study are not normally distributed [39].

Step 6: Evaluating the influence of multiple social aspects on design decay. We rely on a *multiple logistic regression* [39] model to evaluate the influence of social aspects on design decay. In this model, we analyze each group of social aspects separately. Moreover, the *multiple logistic regression* model calculates the odds ratio using each metric in the presence of each one other. All the social aspects and their related metrics presented in Table I are used as 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* model 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 from our models to avoid the effects of *multicollinearity* [40].

To measure the relative impact, we analyzed the magnitude of the effect of the metrics over the possibility of a merged pull request on the design decay. To this end, we estimate the relative impact using the odds ratio [41]. *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 that lead to the decay of the system occurring per "unit" value of a predictor (metric). 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 [38]. Moreover, we normalized the continuous predictors in the model to provide normality. As a result, the mean of each predictor is equaled to 0, and the standard deviation to 1. To ensure the statistical significance of the predictors, we employ the customary $p - value < 0.05$ for each predictor in the regression models.

IV. Results and Discussion

A. Social Metrics and Impactful Pull Requests

We address $\mathbf{RQ_1}$ by using the *Wilcoxon Rank Sum Test* to assess the relationship between social metrics, from different social aspects, and impactful and unimpactful pull requests. The analysis was made taking into consideration four different scenarios of design decay: (i) the density and (ii) diversity of class-level design decay symptoms; (iii) density and (iv) diversity of method-level design decay symptoms. Table V presents the results of our analysis. Since the results of density and diversity for both levels of design decay were the same, we are only showing the design level (class-level and method-level) on the table. Moreover, the statistical significant metrics ($p - value < 0.05$) are highlighted with a gray background.

The experience of the past stay in the past. Among all the metrics analyzed for both class-level and method-level in the projects, we highlight that the *Number of Developers Inactive (# Devs Inactive)* metric do not exhibit any statistically significant relationship in any of the analyzed projects. We used a threshold of 180 days of inactivity for developers who have previously committed to the repository as a sign that they may no longer be up-to-date with the project's discussions and changes. In this sense, their continued presence in pull request conversations may contribute to design decay. Our results suggest that despite a developer's past contributions, their presence in the pull requests conversations may not necessarily indicate any change in the design decay symptoms. Thus, the *# Devs Inactive* metric will be eliminated from the forthcoming analysis. On the other hand, the presence of new developers in the community (*# New Devs*) can suggest that their lack of experience is related to changes in the design.

Gender relation with the decay. In Table V we can see that the two metrics related to gender had divergent results. The *# Females* only was significant on 4 of the 11 projects for the method-level smells and 3 of the 11 projects for the class-level smells. Conversely, the *# Males* was significant in all projects for both types of smells. These results show us that gender has a relation with changes in the decay symptoms (either improvement or deterioration). To better understand why the

TABLE IV
BLAU INDEX FOR THE GENDER DIVERSITY

	Gender Diversity	
	Females	Males
ExoPlayer	0.114	0.612
guava	0.045	0.532
gson	0.404	0.632
dagger	0.010	0.647
guice	0.022	0.689
spring-boot	0.130	0.504
spring-security	0.282	0.573
zuul	0.027	0.521
eureka	0.077	0.619
Hystrix	0.020	0.583
conductor	0.109	0.672

 # Females were significant in less than half of the projects, we applied the *Blau index* [42] in our data as a measurement of diversity. A higher Blau index value indicates greater diversity, while a lower value indicates less diversity. The results, in Table IV, indicated that in projects where *# Females* do not show significance, the Blau index is below 0.3, emphasizing the need for further investigation into this matter.

(Key)words are efficient to determine impactful and unimpactful pull requests apart. Our results from $\mathbf{RQ}_1$ showed us that the metrics from the *discussion content* aspect, namely *# of Words, WPCD (Words Per Comments in Discussion), # Design Keywords, # Refact Keywords, DDK (Density of Design Keywords)*, and *DRK* (Density of Refactoring Keywords) have a statistical significant relation with changes in the decay symptoms. Some of these metrics (*# Words* and *WPCD*) do not perform as well on the class-level smells (7 out of 11 projects) as the keyword metrics (*DDK*, *DRK*, *# Design Keywords*, and *# Refact Keywords*). The results of our study indicate a strong correlation between *discussion content* and changes in design symptoms. This is expected, based on our rationale, but is particularly significant for future research, as these textual metrics can be easily processed by text analyzers or NLP algorithms. Also, bots that monitor these metrics could analyze pull requests conversations in real time to detect potentially impactful design changes.

> **Finding 1**: The most prominent metrics for differentiating impactful and unimpactful pull requests are the size and duration of a discussion, the presence of design-related keywords, the team size, and gender diversity.

B. Organizational Dynamics and Decay Decay

In contrast to $\mathbf{RQ}_1$, which solely investigated the occurrence of a design changes (whether there were an addition or removal of design decay symptoms), we address the $\mathbf{RQ}_2$ by analyzing the impact of *organizational dynamics* on design decay. To this end, we examine the influence, namely the odds ratio of a social aspect increasing or decreasing the design decay symptoms, of individual metrics within this aspect and their interactions with each other. We used a *multiple logistic regression* to perform this analysis. Table VI overviews the analysis results, where each row represents the metrics for a specific project, divided by class-level smells and method-level smells. The gray cells indicate the metrics that are statistically significant with a $p - value$ less than 0.05. Arrows indicate their behavior, whether it increases the odd of the degradation happening (upward arrow) or decreases it (downward arrow). Moreover, an odds of 1 means no effect, while greater than 1 indicates increased odds and less than 1 indicates decreased odds. Finally, these results provide us with insights into which actions or activities, related to the *organization dynamics* aspect, within a pull request conversation have a relationship with the improvement or deterioration of design decay symptoms. We discuss the results in detail as follows.

Organizational growth avoids decay. Based on our analysis of the *organizational dynamics* aspect related to design decay, we found that all metrics, when statistically significant, have a positive impact (decrease of the design decay symptoms) on the design of the software system. Specifically, we have observed that having a larger team size and the presence of new developers in the team is associated with a lower likelihood of design decay. Based on our rationale, the outcome of the metric *Team Size* is expected, as a larger team means that the assignments are distributed among more members, leading to a more balanced workload. Regarding team size indicated that, on average, there were three active developers involved in the 90-day period leading up to the closure of the pull request. The maximum team size is in the spring community with 27 developers. In conclusion, our findings are corroborated by prior research [43], which demonstrates a positive association between larger team participation and improved code quality.

Surprisingly, we also observed that the metric *# New Devs* was associated with a decrease in design decay symptoms. This was unexpected, as new developers are typically assumed to lack sufficient experience to contribute to complex tasks such as those related to design changes. In summary, it may be beneficial for software companies to focus on responsively growing their teams as a means of improving software quality.

> **Finding 2**: A larger team is associated with a lower likelihood of design decay. Our understanding is that more developers involved in the daily tasks help to decrease the workload, allowing the developers to have more time to be concerned about code quality.

Gender diversity prevents design decay. Interestingly, we found that having a higher number of *male developers (# Males)* is associated with a lower likelihood of design decay. The presence of male developers might be beneficial to the design process. In contrast, we observed that the impact of the *(# Females)* developers on the design decay symptoms is not statistically significant in most of the analyzed projects, except for spring-security and eureka, where it exhibits a similar

TABLE V
RESULTS OF THE STATISTICAL SIGNIFICANCE (P-VALUE) OF THE WILCOXON RANK SUM TEST

	exoplayer		guava		gson		dagger		guice		spring-boot		spring-security		zuul		eureka		Hystrix		conductor	
	CL	ML	CL	ML	CL	ML	CL	ML	CL	ML	CL	ML	CL	ML	CL	ML	CL	ML	CL	ML	CL	ML
Communication Dynamics																						
# Newbies	0.765	0.672	0.818	0.474	0.19	0.046	0.0	0.0	0.0	0.0	0.674	0.881	0.507	0.978	0.584	0.813	0.0	0.0	0.0	0.0	0.0	0.0
# Contributors	0.0	0.0	0.0	0.0	0.0	0.0	0.478	0.894	0.401	0.124	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
# Core Devs	0.983	0.981	0.0	0.0	0.0	0.0	0.0	0.002	0.039	0.0	0.0	0.0	0.0	0.0	0.987	0.874	0.965	0.715	1.0	1.0	0.988	0.985
Discussion Size	0.0	0.0	0.0	0.0	0.0	0.0	0.035	0.22	0.335	0.331	0.0	0.0	0.0	0.0	0.005	0.003	0.0	0.0	0.0	0.0	0.0	0.0
MTBC	0.0	0.0	0.259	0.221	0.0	0.0	0.764	0.964	0.798	0.771	0.007	0.003	0.0	0.0	0.488	0.27	0.0	0.0	0.0	0.0	0.0	0.0
Discussion Duration	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TBOF	0.0	0.0	0.425	0.713	0.0	0.0	0.6	0.984	0.988	0.927	0.369	0.285	0.0	0.0	0.704	0.295	0.0	0.0	0.0	0.0	0.0	0.0
TBLM	0.0	0.0	0.046	0.051	0.0	0.0	0.292	0.511	0.62	0.698	0.0	0.0	0.0	0.0	0.088	0.094	0.0	0.0	0.0	0.0	0.0	0.0
Discussion Content																						
# Words	0.0	0.0	0.063	0.018	0.0	0.0	0.144	0.68	0.841	0.894	0.001	0.0	0.0	0.0	0.054	0.041	0.0	0.0	0.0	0.0	0.0	0.0
WPCD	0.0	0.0	0.052	0.016	0.0	0.0	0.146	0.684	0.862	0.914	0.003	0.001	0.0	0.0	0.056	0.046	0.0	0.0	0.0	0.0	0.0	0.0
# Design Keywords	0.004	0.0	0.022	0.111	0.0	0.0	0.0	0.0	0.02	0.005	0.028	0.039	0.004	0.039	0.685	0.623	0.007	0.002	0.0	0.001	0.0	0.0
# Refact Keywords	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
DDK	0.005	0.0	0.019	0.101	0.001	0.0	0.0	0.0	0.017	0.004	0.062	0.098	0.005	0.041	0.705	0.659	0.009	0.003	0.0	0.001	0.0	0.0
DRK	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Organizational Dynamics																						
Team Size	0.001	0.0	0.0	0.0	0.0	0.0	0.002	0.001	0.0	0.0	0.0	0.0	0.0	0.0	0.013	0.046	0.134	0.001	0.077	0.106	0.014	0.002
# Females	0.2	0.091	0.852	0.71	0.001	0.0	0.982	0.935	0.98	0.977	0.001	0.0	0.0	0.0	0.969	0.757	0.73	0.198	0.979	0.977	0.172	0.011
# Males	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
# New Devs	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.329	0.095	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
# Devs Inactive	0.835	0.678	0.831	0.985	0.374	0.538	0.985	0.981	0.873	0.859	0.535	0.55	0.974	0.848	0.975	0.972	0.977	0.973	0.988	0.987	0.991	0.932

Gray cells represent statistically significant differences, with (*p-value* < 0.05) between impactful and unimpactful pull requests.

TABLE VI
MULTIPLE LOGISTICS REGRESSION RESULTS FOR THE ORGANIZATIONAL DYNAMICS ASPECT

	Team Size		**# Females**		**# Males**		**# New Devs**	
	CL	ML	CL	ML	CL	ML	CL	ML
exoplayer	0.673 ↓	0.531 ↓	0.927	0.941	0.253 ↓	0.367 ↓	0.311 ↓	0.213 ↓
guava	0.433 ↓	0.52 ↓	0.951	0.278	0.464 ↓	0.343 ↓	0.311 ↓	0.378 ↓
gson	0.175 ↓	0.162 ↓	0.95	1.143	1.012	1.501	0.03 ↓	0.019 ↓
dagger	1.126	1.09	0.527	0.524	0.237 ↓	0.25 ↓	1.155	1.146
guice	0.291 ↓	0.401 ↓	3.81	3.536	1.131	0.792	0.245 ↓	0.343 ↓
spring-boot	0.335 ↓	0.275 ↓	0.961	0.703	0.01 ↓	0.011 ↓	0.795	0.79
spring-security	0.773 ↓	0.796 ↓	0.561 ↓	0.565 ↓	0.152 ↓	0.15 ↓	0.561 ↓	0.548 ↓
zuul	0.559 ↓	0.51 ↓	1.061	1.066	1.038	1.062	0.095 ↓	0.09 ↓
eureka	0.263 ↓	0.317 ↓	0.684 ↓	0.74 ↓	0.387 ↓	0.366 ↓	0.057 ↓	0.079 ↓
Hystrix	0.933	0.93	0.409	0.408	0.006 ↓	0.008 ↓	0.773	0.845
conductor	0.846	0.858	0.93	0.955	0.596 ↓	0.629 ↓	0.082 ↓	0.077 ↓

TABLE VII
RESULTS OF MULTIPLE LOGISTIC REGRESSION FOR THE DATA COMBINED BY THE SOFTWARE COMMUNITIES: GOOGLE, SPRING, AND NETFLIX

	Google		Spring		Netflix	
	CL	ML	CL	ML	CL	ML
Communication Dynamics						
# Newbies	0.259 ↓	0.265 ↓	1.278 ↑	1.282 ↑	0.963	0.967
# Contributors	0.299 ↓	0.281 ↓				
# Core Devs	0.523 ↓	0.476 ↓	0.119 ↓	0.119 ↓	0.955	0.97
Discussion Size	0.185 ↓	0.162 ↓				
MTBC	1.164	1.36	1.308 ↑	1.336 ↑	1.056	1.04
TBOF	1.105	1.039	1.403 ↑	1.397 ↑	1.446 ↑	1.46 ↑
TBLM	0.842 ↓	0.777 ↓	0.073 ↓	0.071 ↓	0.684 ↓	0.677 ↓
Discussion Content						
# Words	26.319	22.847	0.0 ↓	0.0 ↓	0.0 ↓	0.0 ↓
WPCD	0.734	0.985	5.964 ↑	4.709 ↑	7.95 ↑	3.823 ↑
# Design Keywords	1.578	0.861	5.884 ↑	4.527	5.176 ↑	3.713
# Refact Keywords			2.609 ↑	1.443 ↑	3.873 ↑	4.149 ↑
DDK	0.558	0.981	0.182 ↓	0.233	0.188 ↓	0.263
DRK	0.771 ↓	0.793 ↓	0.003 ↓	0.004 ↓	0.0 ↓	0.0 ↓
Organizational Dynamics						
Team Size	0.632 ↓	0.614 ↓			0.766 ↓	0.758 ↓
# Females	0.895 ↓	0.91	0.446 ↓	0.436 ↓	0.856 ↓	0.873 ↓
# Males	0.812	0.922				
# New Devs	0.476 ↓	0.459 ↓				

behavior as the *# Males*. This finding is interesting because it suggests that gender diversity can have a positive influence on reducing design decay symptoms. We also investigated the reason behind the non-significance of *# Females* in the other projects and found that many pull request conversations had no female participation, which could affect the regression analysis results. Finally, since our analysis above indicates that gender diversity is a potential factor that could impact the design, we are going to keep investigating these metrics in the next RQs in order to fully understand their behavior in an environment with a larger sample of data, *e.g.,* grouped by communities and the projects altogether. Overall, our results highlight the importance of considering the *organizational dynamics* when attempting to mitigate design decay.

Finding 3: Male developers play a significant role in the design process, while the presence of female developers can also have a positive impact on design quality, suggesting that gender diversity may contribute to better design outcomes in software development projects.

Turnover and decay. We identified that the presence of new developers in a team *(# New Devs)* is strongly associated with a decrease in the design decay symptoms. Previous studies [44], [45] support our finding that newcomers typically do not introduce new code smells, as more experienced developers handle more complex and critical tasks. Our analysis of this metric revealed statistical significance in nine out of 11 projects, with a strong relationship indicated by odds ratios very close to 0 in four out of nine projects. Moreover, previous studies [46] noted that newcomers tend to avoid handling complex tasks and are often mentored or accompanied by experienced developers. Finally, our results suggest that this dynamic may lead to positive outcomes for software design.

C. Social Aspects by Community

In this RQ we explored the behavior of social aspects in pull request conversations across various software projects within the same software community. To achieve this, we

utilized the *Multiple Logistic Regression* to analyze all the metrics across projects belonging to three communities: namely Google (exoplayer, guava, gson, and guice); Spring (spring-boot and spring-security); and Netflix (zuul, eureka, hystrix, and conductor). The goal is to investigate how these social aspects interact with each other in the presence of multiple factors. We can see the results of the analysis in the Table VII. In this table, we can observe that some metrics have blank cells in their rows. Those blank cells are metrics which have a pair-wise correlation above 0.7 [40]. They were removed to avoid the effects of *multicollinearity*.

The Google Community. Our findings suggest that all three social aspects – *communication dynamics, discussion contents*, and *organizational dynamics* – have a positive impact on design. Specifically, metrics related to *communication dynamics*, namely *# Newbies, # Contributors, # Core Devs, Discussion Size*, and *TBLM* (Time Between Last Commend and Merge), are associated with a decrease in design decay symptoms. Interestingly, the odds ratio indicates that the influence of *# Newbies* is stronger than *# Contributors*, which in turn is stronger than *# Core Devs*. This aligns with previous studies [44], [46], which suggest that less experienced developers tend to handle less complex tasks, leading to better software quality. The *DRK* (Density of Refactoring Keywords) metric, related to refactoring discussions, is also linked to a decrease in decay symptoms, implying that discussions focused on refactoring have a positive impact on design quality.

The *organizational dynamics* metrics, such as *Team Size, # Females*, and *# New Devs*, also indicate an improvement in design quality. This finding supports the results of RQ_2, which concludes that a larger team size and greater gender diversity are associated with a decreased risk of design decay. In summary, we can say that *communication* (5 metrics) and *organizational dynamics* (3 metrics) are the two aspects that are more useful for the Google community.

Finding 4: For the Google community, the two most beneficial aspects were communication dynamics and organizational dynamics, both being relevant in increasing the design quality.

The Spring Community. Unlike the Google community, which had only one statistically significant metric related to *discussion content*, here, many of its metrics were significant. In terms of *communication dynamics: # Newbies, MTBC* (Mean Time Between Comments), and *TBOF* (Time Between Pull Request Opening and First Comment) showed a tendency to increase the likelihood of design decay. This tendency for *# Newbies* contrasts with the results for Google, suggesting that, here, new developers may be involved in more complex tasks, leading to design decay. Additionally, the results for *MTBC* and *TBOF* align with our reasoning, since the developers involved in the discussion were not engaged, they may have missed design issues that could end up in the merged code. On the other hand, *# Core Devs* and *TBLM* (Time Between Last Comment and Merge) showed a risk-decreasing tendency for decay, demonstrating that the presence of more experienced developers in discussions was helpful to maintain the design. Moreover, the final feedback of the pull requests was crucial in keeping the developer engaged and focused on the task, ensuring it was merged without compromising design.

As previously mentioned, several metrics related to the *discussion content* aspect appeared as statistically significant: *# Words, DDK* (Density of Design Keywords), and *DRK* (Density of Refactoring Keywords) presented a risk-decreasing tendency, while their counterparts *WPCD* (Words Per Comment in Discussion), *# Design Keywords*, and *# Refactoring Keywords* presented a risk-increasing tendency. This finding is noteworthy because it suggests that pull request conversations with a high number of words and a high proportion of design-related keywords can help preserve the design. This result will be further discussed in the next RQ. Finally, the *organizational dynamics* aspect only had one statistically significant metric, with a risk-decreasing tendency, which was the number of female developers. This result was also observed in the Google community, providing further support for the importance of gender diversity in improving the design quality.

Finding 5: For the Spring community, the two most beneficial aspects were communication dynamics and discussion content, both aspects presented metrics related to the increasing and decreasing of design decay symptoms.

The Netflix Community. Upon analyzing the results for this community, we observed that it differs from the other communities in terms of relevant social aspects. Table VII shows that only two metrics from the *communication dynamics aspect*, two from the *organizational dynamics*, and all from the *discussion contents* showed statistical significance. Furthermore, we can see both increasing and decreasing tendencies in the metrics regarding the design decay symptoms. In the *communication dynamics*, the metrics *TBOF* (Time Between Pull Request Opening and First Comment) and *TBLM* (Time Between Last Comment and Merge) showed contrasting results. While *TBOF* was related to the increasing of design decay symptoms, consistent with the findings from the Spring community, *TBLM* showed the opposite behavior.

This finding is consistent with our rationale, and also highlights the significance of providing prompt feedback within the community, as it helps to maintain high engagement, and, consequently, the design quality. The *discussion content* aspect have the same behavior of the Spring community, with *# Words, DDK* (Density of Design Keywords), and *DRK* (Density of Refactoring Keywords), presenting a relation with design improvement, and *WPCD* (Words Per Comment in Discussion), *# Design Keywords*, and *# Refactoring Keywords* a relation with design decay. These findings will be discussed in more detail in the results of $\mathbf{RQ}_4$. Finally, the *organizational dynamics* aspect for the Netflix community indicates that both *Team Size* and *# Females* developers were associated with a decrease in design decay symptoms, which supports the results of $\mathbf{RQ}_2$ and strengthens these findings.

TABLE VIII
RESULTS OF MULTIPLE LOGISTIC REGRESSION FOR THE DATA COMBINED OF ALL PROJECTS.

	method-level	class-level
Communication Dynamics		
# Newbies	0.823 ↓	0.829 ↓
Discussion Size	0.253 ↓	0.257 ↓
MTBC	1.061	1.086
TBOF	1.322 ↑	1.318 ↑
TBLM	0.946	0.932
Discussion Content		
# Words	1.358 ↑	1.397 ↑
WPCD	0.799 ↓	0.766 ↓
# Design Keywords	1.08	1.015
# Refact Keywords	2.828 ↑	3.1 ↑
DDK	0.877	0.917
DRK	0.359 ↓	0.336 ↓
Organizational Dynamics		
Team Size	0.776 ↓	0.766 ↓
# Females	0.836 ↓	0.85 ↓

Finding 6: For the Netflix community, the most relevant aspect was the discussion content. However, the results for the communication dynamics and organizational dynamics aspects also provide valuable insights on how to manage design quality.

In conclusion, while various social aspects are consistently relevant across different projects and communities, each community has its own set of aspects that can be utilized to ensure the preservation of design quality.

D. Social Aspects and Design Decay

To address the $\mathbf{RQ}_4$, we applied the same methodology of the $\mathbf{RQ}_2$ and $\mathbf{RQ}_3$. However, in this RQ, we applied the *Multiple Logistic Regression* in all metrics of the 11 projects combined. Table VIII shows the results of this analysis. Similar to the previous RQ, the gray cells and the arrows have the same meaning. Moreover, we can observe that some of the metrics are missing, because those metrics had a pair-wise correlation above 0.7 [40]; and they were removed to avoid the effects of *multicollinearity*. Finally, we can also observe a consistency between class-level and method-level design decay symptoms, as the metrics were significant and maintained the same behavior (arrow) in both levels.

Communication Dynamics improves design quality. Our study revealed important factors of *communication dynamics* that are related to design decay. In particular, we found that pull request conversations with a high volume of comments *(Discussion Size)* and a higher number of newbies *(# Newbies)* are associated with a lower likelihood of design decay. We observed an average of 10 comments and two newbies per pull request. The maximum number of comments in a single pull request is 60 within the Netflix community, while the Spring community has a maximum of six newbies per pull request. These results suggest that open and collaborative discussions among team members help to identify and address potential design issues early on in the development process.

Delayed feedback deteriorating design quality. We also observed that longer times between the opening of a pull request and the first comment, metric *TBOF* (Time Between Pull Request Opening and First Comment), are associated with an increase in the design decay symptoms. We observed that the average duration between the opening of a pull request and the first comment is 40 days. The Spring community exhibited the longest duration, with 2600 days between these events. This finding indicates that timely feedback and response to pull requests are critical in order to maintain the quality of the software design. Our results highlight the importance of effective communication and collaboration within software development teams, and emphasize the need for prompt and regular feedback in order to avoid design decay. Our findings underscore the significance of efficient communication and collaboration among team members, offering valuable insights to guide best practices for software development teams.

Finding 7: Faster feedback, high engagement in the communication, and participation of stakeholders can mitigate the risk of design decay.

Discussion Content needs substantial engagement in the topic to improve the design. While analyzing the results, we found that a higher number of words (*# Words*) and refactoring keywords (*# Refactor Keywords*) in pull request conversations were associated with a greater risk of design decay. In contrast, the density of design keywords (*DDK*) and words per comment in discussions (*WPDD*) were found to be positively related to the improvement of the design. Moreover, we found similar results for this set of metrics in $\mathbf{RQ}_3$ across two different communities, which only strengthens these findings. Furthermore, maintaining a balance between the amount of discussion and the inclusion of relevant keywords is essential for mitigating design decay. The reason is that simply having a high number of words and refactoring keywords in a discussion does not necessarily contribute to design improvement. Instead, we found that a high number of words and keywords in multiple comments is more likely to lead to design improvement. This means that a design-focused discussion with multiple participants contributing substantial comments is the key for the improvement of the design symptoms.

Finding 8: Discussions focused on refactoring with multiple participants and large comments improve design, while a high number of words and refactoring keywords alone do not.

Gender Diversity and Team Size. We found that the number of females (*# Females*) in a development team and the team size were both related to a decrease in design decay. These findings suggest that a diverse range of perspectives and experiences within a team may contribute to more effective design decision-making and better outcomes in the long term. These results are also interesting when we look into our findings in $\mathbf{RQ}_2$, where we found that the number of males (*# Males*) in the team is also related to the improvement of

the design symptoms. With that in hand, we can state that gender diversity may play an important role in mitigating design decay, and that team size (*Team size*) can also have a significant impact on design outcomes. These results highlight the importance of creating teams that are both diverse and well-resourced, with enough members to effectively manage and maintain design decisions over time.

Finding 9: The presence of more female developers and larger team sizes were found to be associated with a decreased risk of design decay in the study. The findings suggest that gender diversity and teamwork are important factors in preventing design decay in software development teams.

V. Threats to Validity

Construct and Internal Validity. The results of this study may have been influenced by factors such as the precision and recall of degradation symptoms. To address this potential threat, we utilized Organic tool [12] to detect the class and method-level degradation symptoms analyzed in this paper. Thus, the detection strategies and thresholds utilized by Organic may introduce bias to the results, but this tool has been previously used and validated by several papers [6], [7], [13], [47], and the detection strategies it uses also come from previous studies on smell detection [48].

This study may not represent a holistic view of all social aspects and interactions that may affect software development. This threat was taken into consideration when we defined the set of metrics and software projects analyzed in this work. Due to this threat, we also defined a strict methodology for the execution of the statistical analysis executed in this work. However, we are aware that some metrics chosen may not paint an entirely accurate depiction of reality. The gender metric, for example, was estimated based on an optional field (name) mined from GitHub profiles.

Conclusion and External Validity. In order to mitigate potential problems with external validity, we performed our analyses carefully. In the descriptive analysis, multiple authors contributed to reviewing and 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. Instead, we used the Wilcoxon Rank Sum Test [38], which is non-parametric. For the regression analysis, we performed additional transformations ($\log_2$ and x^3) to reduce the skewness of the metrics. 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 [40]. Furthermore, in order to ensure normality, we normalized the continuous predictors in the model.

Due to aforementioned constraints, we also limited our investigation of design symptoms to systems developed in Java. Due to this, our results might have a bias towards the 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.

VI. Conclusion and Future Work

This study explored the relationship between social aspects found within pull requests conversations and design decay symptoms. To achieve this, we first identified 18 types of design decay symptoms, which were categorized as class-level and method-level smells. Then, we analyzed social metrics to determine which ones were the most prominent to differentiate between decayed and non-decayed merged pull requests. In addition, the study investigated the extent to which organizational dynamics influenced the occurrence of design decay. This involved analyzing factors such as the gender diversity, team size and the turnover of the team. Furthermore, we analyzed the influence of three social aspects: communication dynamics, discussion content, and organizational dynamics on the design decay within three software communities: Google, Spring, and Netflix. Finally, we studied how the social aspects influenced the design decay when considering the projects altogether. Overall, the study provides valuable insights into the factors that contribute to design decay in software development teams, and highlights the importance of effective communication among team members to mitigate the risk of design decay.

From that analysis, we have the following findings: (i) the size and duration of the discussion, the use of design and refactoring keywords, and the team size and gender diversity are the key metrics in distinguishing impactful from unimpactful pull requests; (ii) a larger team and the gender diversity are crucial indicators of improvement in the design quality; (iii) each software community has its own set of social aspects that are most useful in identifying design decay; (iv) improvements in design can be achieved through prompt feedback, active participation in communication, and discussions that are focused on refactoring with the involvement of multiple participants who make substantial comments. These results are important for communities/companies, since they can be able to identify behaviors that are beneficial to the design before the code being merged in the codebase.

Regarding future work, we plan to develop a machine learning model to automatically detect pull requests that may be at risk of experiencing an increase in design decay symptoms. This model will be integrated into a GitHub Bot that can be installed in the repository to monitor pull requests and alert developers responsible for them.

Acknowledgment

This work is partially funded by CNPq (140770/2021-6 (140771/2021-2, 140185/2020-8, 434969/2018-4, 312149/2016-6, 409536/2017-2, and 427787/2018-1), CAPES/Procad (175956), CAPES/Proex (88887.373933/2019-00), FAPERJ (22520-7/2016), FAPERJ/PDR-10 (202073/2020), FUNCAP (BP5-00197-00042.01.00/22), and by the Instituto Estadual de Engenharia e Arquitetura (IEEA), Secretaria Estadual de Infraestrutura do Estado do Rio de Janeiro (001/2021).

REFERENCES

[1] C. Barbosa, A. Uchôa, D. Coutinho, F. Falcão, H. Brito, G. Amaral, V. Soares, A. Garcia, B. Fonseca, M. Ribeiro *et al.*, "Revealing the social aspects of design decay: A retrospective study of pull requests," in *34th Brazilian Symposium on Software Engineering (SBES)*, 2020, pp. 364–373.

[2] A. Uchôa, C. Barbosa, W. Oizumi, P. Blenilio, R. Lima, A. Garcia, and C. Bezerra, "How Does Modern Code Review Impact Software Design Degradation? An In-depth Empirical Study," in *36th International Conference on Software Maintenance and Evolution(ICSME)*, 2020, pp. 1–12.

[3] A. Uchôa, C. Barbosa, D. Coutinho, W. Oizumi, W. K. Assunçao, S. R. Vergilio, J. A. Pereira, A. Oliveira, and A. Garcia, "Predicting Design Impactful Changes in Modern Code Review: A Large-Scale Empirical Study," in *IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 2021, pp. 1–12.

[4] R. N. Taylor and A. Van der Hoek, "Software design and architecture the once and future focus of software engineering," in *Future of Software Engineering (FOSE)*, 2007, pp. 226–243.

[5] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, 2014, pp. 356–366.

[6] L. Sousa, A. Oliveira, W. Oizumi, S. Barbosa, A. Garcia, J. Lee, M. Kalinowski, R. de Mello, B. Fonseca, R. Oliveira *et al.*, "Identifying design problems in the source code: A grounded theory," in *Proceedings of the 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 921–931.

[7] W. Oizumi, A. Garcia, L. Sousa, B. Cafeo, and Y. Zhao, "Code anomalies flock together: Exploring code anomaly agglomerations for locating design problems," in *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, 2016.

[8] A. Yamashita, M. Zanoni, F. A. Fontana, and B. Walter, "Inter-smell relations in industrial and open source systems: A replication and comparative analysis," in *31st International Conference on Software Maintenance and Evolution(ICSME)*, 2015, pp. 121–130.

[9] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus, "Does code decay? assessing the evidence from change management data," *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 1–12, 2001.

[10] M. Fowler, *Refactoring*. Addison-Wesley Professional, 1999.

[11] T. Sharma and D. Spinellis, "A survey on software smells," *J. Syst. Softw. (JSS)*, vol. 138, pp. 158–173, 2018.

[12] W. Oizumi, L. Sousa, A. Oliveira, A. Garcia, A. B. Agbachi, R. Oliveira, and C. Lucena, "On the identification of design problems in stinky code: experiences and tool support," *Journal of the Brazilian Computer Society*, vol. 24, pp. 1–30, 2018.

[13] R. Oliveira, R. de Mello, E. Fernandes, A. Garcia, and C. Lucena, "Collaborative or individual identification of code smells? on the effectiveness of novice and professional developers," *Information and Software Technology*, vol. 120, p. 106242, 2020.

[14] R. de Mello, R. Oliveira, A. Uchôa, W. Oizumi, A. Garcia, B. Fonseca, and F. de Mello, "Recommendations for developers identifying code smells," *IEEE Software*, 2022.

[15] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, 2014, pp. 345–355.

[16] G. Gousios, A. Zaidman, M.-A. Storey, and A. Van Deursen, "Work practices and challenges in pull-based development: the integrator's perspective," in *37th ICSE*, vol. 1, 2015, pp. 358–368.

[17] C. Treude and M.-A. Storey, "Effective communication of software development knowledge through community portals," in *19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ser. ESEC/FSE. ACM, 2011, pp. 91–101.

[18] I. S. Wiese, F. R. Côgo, R. Ré, I. Steinmacher, and M. A. Gerosa, "Social metrics included in prediction models on software engineering: a mapping study," in *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*, 2014, pp. 72–81.

[19] N. Bettenburg and A. E. Hassan, "Studying the impact of social interactions on software quality," *Emp. Softw. Eng. (ESE)*, vol. 18, no. 2, pp. 375–431, 2013.

[20] A. E. Hassan, "Predicting faults using the complexity of code changes," in *Proceedings of the 31th International Conference on Software Engineering (ICSE)*, 2009, pp. 78–88.

[21] F. Falcão, C. Barbosa, B. Fonseca, A. Garcia, M. Ribeiro, and R. Gheyi, "On Relating Technical, Social Factors, and the Introduction of Bugs," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 378–388.

[22] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "What is social debt in software engineering?" in *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2013, pp. 93–96.

[23] D. A. Tamburri, P. Kruchten, P. Lago, and H. Van Vliet, "Social debt in software engineering: insights from industry," *Journal of Internet Services and Applications*, vol. 6, no. 1, pp. 1–17, 2015.

[24] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *Proceedings of the 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 712–721.

[25] R. Morales, S. McIntosh, and F. Khomh, "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)*. IEEE, 2015, pp. 171–180.

[26] L. Pascarella, D. Spadini, F. Palomba, and A. Bacchelli, "On the effect of code review on code smells," in *Proceedings of the 27th International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2019, pp. 1–12.

[27] D. Sas, P. Avgeriou, I. Pigazzini, and F. Arcelli Fontana, "On the relation between architectural smells and source code changes," *Journal of Software: Evolution and Process*, vol. 34, no. 1, p. e2398, 2022.

[28] V. Soares, A. Oliveira, P. Farah, A. Bibiano, D. Coutinho, A. Garcia, S. Vergilio, M. Schots, D. Oliveira, and A. Uchôa, "On the Relation between Complexity, Explicitness, Effectiveness of Refactorings and Non-Functional Concerns," in *34th Brazilian Symposium on Software Engineering (SBES)*, 2020, pp. 788–797.

[29] D. Coutinho, A. Uchôa, C. Barbosa, V. Soares, A. Garcia, M. Schots, J. Pereira, and W. K. Assunçao, "On the influential interactive factors on degrees of design decay: A multi-project study," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 753–764.

[30] V. R. B.-G. Caldiera and H. D. Rombach, "Goal question metric paradigm," *Encyclopedia of software engineering*, vol. 1, no. 528-532, p. 6, 1994.

[31] "Pmd," https://pmd.github.io/, (Accessed on 01/19/2023).

[32] "Sonarqube," https://www.sonarsource.com/products/sonarqube/, (Accessed on 01/19/2023).

[33] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "An in-depth study of the promises and perils of mining GitHub," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, 2016.

[34] M. Paixao and P. H. Maia, "Rebasing in code review considered harmful: A large-scale empirical investigation," in *19th SCAM*, 2020, pp. 45–55.

[35] M. Paixão, A. Uchôa, A. C. Bibiano, D. Oliveira, A. Garcia, J. Krinke, and E. Arvonio, "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, pp. 125–136.

[36] "Replication Package," https://opus-research.github.io/beyond_the_code_pull_requests_design_decay/, 2023.

[37] M. S. Wessel, M. F. Aniche, G. A. Oliva, M. A. Gerosa, and I. S. Wiese, "Tweaking Association Rules to Optimize Software Change Recommendations," in *31st Brazilian Symposium on Software Engineering (SBES)*, 2017, pp. 94–103.

[38] E. Whitley and J. Ball, "Statistics review 6: Nonparametric methods," *Critical care*, vol. 6, no. 6, p. 509, 2002.

[39] J. H. McDonald, *Handbook of biological statistics*. sparky house publishing Baltimore, MD, 2009, vol. 2.

[40] C. F. Dormann, J. Elith, S. Bacher, C. Buchmann, G. Carl, G. Carré, J. R. G. Marquéz, B. Gruber, B. Lafourcade, P. J. Leitão *et al.*, "Collinearity: a review of methods to deal with it and a simulation study evaluating their performance," *Ecography*, vol. 36, no. 1, pp. 27–46, 2013.

[41] A. W. Edwards, "The measure of association in a $2\times$ 2 table," *J. Royal Stat. Soc.*, vol. 126, no. 1, pp. 109–114, 1963.

[42] P. M. Blau, *Inequality and heterogeneity: A primitive theory of social structure*. Free Press New York, 1977, vol. 7.

[43] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Emp. Softw. Eng. (ESE)*, vol. 21, no. 5, pp. 2146–2189, 2016.

[44] M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, and D. Poshyvanyk, "When and why your code starts to smell bad (and whether the smells go away)," *IEEE Transactions on Software Engineering*, vol. 43, no. 11, pp. 1063–1088, 2017.

[45] S. Habchi, N. Moha, and R. Rouvoy, "The rise of android code smells: Who is to blame?" in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 445–456.

[46] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles, "Social barriers faced by newcomers placing their first contribution in open source software projects," in *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*, 2015, pp. 1379–1392.

[47] D. Cedrim, A. Garcia, M. Mongiovi, R. Gheyi, L. Sousa, R. de Mello, B. Fonseca, M. Ribeiro, and A. Chávez, "Understanding the impact of refactoring on smells: A longitudinal study of 23 software projects," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 465–475.

[48] M. Lanza and R. Marinescu, *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media, 2007.