
A Multi-Faceted Analysis of How Organizations Create and Maintain Code Samples

Matheus Albuquerque de Melo

SERVIÇO DE PÓS-GRADUAÇÃO DA FACOM-UFMS

Data de Depósito: 06/10/2023

Assinatura: _____

A Multi-Faceted Analysis of How Organizations Create and Maintain Code Samples¹

Matheus Albuquerque de Melo

Advisor: *Prof. Bruno Barbieri de Pontes Cafeo, Ph.D.*

Dissertation delivered to the Faculty of Computing (FACOM/UFMS) as part of requirements to obtain the title of Master in Computing Science.

UFMS
August/2023

¹This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Acknowledgements

I would like to extend my deepest gratitude and dedicate this dissertation to:

First and foremost, to God for always being by my side and allowing me to reach this moment. I also wish to express my heartfelt thanks to my advisor, Bruno Cafeo, for his support and guidance throughout this journey. My appreciation goes to Awdren Fontão and Hudson Borges for their contributions.

To my parents, Rosângela and José, thank you for your love and unwavering belief in me.

I would also like to express my gratitude to my colleagues who accompanied me during this long master's journey: Gabriel Menezes, Karolina Milano, José Neto, Ana Elisa, and Sidny Molina. Lastly, I want to acknowledge all the members of the Software Engineering Laboratory (LEDES) research group.

Abstract

Code samples, artifacts within the realm of software ecosystems, serve to assist developers by exemplifying the use of APIs, libraries, and other resources. Their exploration in literature began relatively recently, around 2019. Studies proposed to understand the structural characteristics of the code of these artifacts, as well as how they undergo maintenance and evolve over time. In 2020, there was also exploration of the target audience consuming this artifact through questions on StackOverflow. This present study was divided into two parts. Firstly, we investigated code samples repositories on GitHub and analyzed how organizations handles contributions from external developers within this environment. In the second part, we conducted a survey with developers who produce code samples within organizations. This allowed us to better understand their vision and perspectives regarding code samples, as well as analyze their experiences with code samples and dedication to these artifacts. The proposals of both parts of the study had not been previously explored in the context of code samples. Our findings revealed some points, such as the delay in reviewing pull requests, especially those that were rejected, and bottlenecks in the distribution of review activities among maintainers. These findings resulted in a publication. Furthermore, we found that the purposes of code samples go beyond educational purposes as suggested by the organizations' developers. We also identified that experienced developers are involved in the development of code samples in organizations, usually dedicating a few hours per month or week to this activity. Finally, we noted the presence of problems of divergence of opinions among the organizations' developers, in relation to the target audience of the code samples, the development process, and the guarantee of the quality of these artifacts.

Resumo

Code samples, como artefatos presentes no contexto dos ecossistemas de *software*, a qual tem a função de auxiliar os desenvolvedores, exemplificando o uso de *APIs*, bibliotecas e outros recursos começaram a ser explorados na literatura recentemente, por volta de 2019. Estudos propuseram a compreender as características estruturais do código desses artefatos, bem como como eles passam por manutenções e evoluem ao longo do tempo. Em 2020, também houve exploração do público-alvo que consome este artefato por meio de perguntas no StackOverflow. Este presente estudo foi dividido em duas partes. Na primeira, investigamos repositórios de *code samples* no GitHub e analisamos como as organizações lidam com as contribuições de desenvolvedores externos dentro desse ambiente. Na segunda parte, realizamos uma pesquisa com desenvolvedores que produzem *code samples* dentro das organizações. Isso nos permitiu compreender melhor a visão e as perspectivas deles em relação aos *code samples*, além de analisar suas experiências com *code samples* e a dedicação a esses artefatos. As propostas de ambas as partes do estudo não haviam sido exploradas anteriormente no contexto de *code samples*. Nossas descobertas revelaram alguns pontos como a demora na revisão de *pull requests*, especialmente os que foram rejeitados, e gargalos na distribuição das atividades de revisão entre os mantenedores. Essas constatações resultaram em uma publicação. Além disso, descobrimos que os objetivos dos *code samples* vão além dos propósitos educacionais, conforme sugerido pelos desenvolvedores das organizações. Também identificamos que desenvolvedores experientes estão envolvidos no desenvolvimento de *code samples* nas organizações, dedicando geralmente algumas horas mensais ou semanais para essa atividade. Por fim, notamos a presença de problemas de divergência de opiniões entre os desenvolvedores das organizações, em relação ao público-alvo dos *code samples*, ao processo de desenvolvimento e à garantia da qualidade desses artefatos.

Contents

Summary	x
List of Figures	xi
List of Tables	xiii
List of Acronyms	xv
1 Introduction	1
2 Background	4
2.1 Software Ecosystem	4
2.2 Code samples	6
2.3 Code samples, SECO and GitHub	8
3 Exploratory study in code sample repositories	13
3.1 Introduction	13
3.2 Study Design	14
3.2.1 Study Scenario	14
3.2.2 Chosen Platform and Organizations	15
3.2.3 Code Sample Selection	16
3.2.4 Pull Request Selection	16
3.2.5 Maintainers Selection	16
3.2.6 Research questions	16
3.3 Results of exploratory study	18
3.4 Implications	22
3.4.1 The community matters	22
3.4.2 To each organization, its rules	23
3.4.3 Rejected Pull Requests: The 'Time Rejectables' Chronicles	23
3.4.4 Expand to conquer	24
3.5 Threats to validity	24
3.6 Final considerations	25

4	Survey with professionals who produce code samples	26
4.1	Introduction	26
4.2	Study Design	27
4.2.1	Target population of survey	27
4.2.2	Criteria for Participant Selection	28
4.2.3	Pilot survey	29
4.2.4	Final survey	29
4.2.5	Research questions	33
4.3	Results of Survey	35
4.3.1	Roles	35
4.3.2	Experience	37
4.3.3	Number of code samples	38
4.3.4	Frequency of work with code samples	38
4.3.5	Reported Activities	40
4.3.6	Main objective of code samples	43
4.3.7	Target audience	44
4.3.8	Internal perspective within each organization	45
4.3.9	Overview of the similarity between development process and quality assurance with other products of the organi- zation	47
4.3.10	Internal perspective within each organization	48
4.4	Implications	50
4.4.1	Code Samples: Beyond Educational Objectives	50
4.4.2	Veteran Developer, Old code samples	51
4.4.3	Internal Disparities in Code Sample Perspectives	51
4.5	Threats to validity	52
4.6	Final Considerations	53
5	Implications	54
5.1	Implications	54
5.1.1	Fostering Collaborative Synergy: Code Samples as Bridges Between Organizations and External Developers	54
5.1.2	Some practices for providing code samples	55
6	Related work	56
7	Conclusions and Future Work	59
	Bibliography	68

List of Figures

2.1	Example of SECO and Actors	5
2.2	Files of <i>code sample</i> android-custom-lint-rules	8
2.3	Example of interaction: A internal developer (Google employee) responding to an issue from a developer	11
2.4	Example of interaction: A bot and a internal developer (Google employee) responding to a pull request from an external developer.	12
3.1	Time spent to review code sample pull requests.	20
3.2	(Left) Time to review accepted pull requests. (Right) Time to review rejected pull requests.	20
3.3	Lorenz curve between code sample maintainers and pull request reviews.	22
4.1	Participants location map	33
4.2	Role in organizations	36
4.3	Years of code samples vs previous organizations	37
4.4	Number of code samples worked on simultaneously	38
4.5	Frequency Work with code samples	39
4.6	Activities in code samples	40
4.7	Less vs More expienced vs Frequency of work	42
4.8	Main objective of code samples	43
4.9	Variation of the target audience for code samples in organizations from the participants' perspective.	44
4.10	Main purpose of providing code samples vs Organizations	45
4.11	Target audience vs Organizations	46
4.12	Process of development similarity vs Organizations	49
4.13	Process of assurance quality vs Organizations	50

List of Tables

3.1 Unreviewed vs Reviewed Pull Requests	18
3.2 Accepted vs Rejected Pull Requests	19

Acronyms

SECO Software Ecosystem. 4–6, 8, 13, 14, 22, 56

Introduction

In the software development process, various actors, including organizations, developers, suppliers, customers, and users, interact to produce artifacts and services that contribute to the construction of a software product. These actors, along with the artifacts they manage and the shared platform that supports this interaction, they constitute a concept called software ecosystems [25, 27].

A code sample is a type of artifact that organizations can provide within a software ecosystem. Organizations distribute this artifact through official websites/blogs targeting developers or on code hosting platforms, such as GitHub. For instance, Amazon offers more than 5,000 code samples related to its AWS product in GitHub repositories [49], and Microsoft over 2,000 [34], among other organizations like Google, Twitter, Oracle, and Spring.

In the literature, we find some definitions about code samples. They are complete software projects that assist developers in their development efforts, exemplifying the use of APIs, libraries, frameworks, and other resources, and evolve over time to maintain their relevance [32].

While the increasing provision of this artifact by organizations in the software ecosystem landscape, code samples remain underexplored in literature. There are still many facets of this artifact to be examined. When we refer to facets, we address the various perspectives or angles a code sample can have. A code sample can embody facets such as: the perspective of organizations and internal developers, the angle of external contributions, the code's structure itself (previously explored) [32, 33], and the target audience that utilizes these artifacts (previously explored) [6], among others.

In this study, we seek to explore facets that remain underexplored in liter-

ature. Beyond the facet related to the interaction between organizations and contributions from external developers in code sample repositories, we examine others facets like the experience, roles, and dedication of internal developers who create and maintain code samples, as well as the internal perspective of organizations regarding this artifact.

Similarly to how open-source projects can benefit from community contributions [2, 40], code samples on platforms like GitHub have similar potential. However, others research has primarily focused on contributions to general repositories and not specifically on the community's interest in code samples. This raises questions about how organizations manage and incorporate these external contributions.

Considering that the environment of open repositories for code samples might be conducive to collaborations between the organization and the community, we propose our first research question: **RQ1: How do organizations engage with external developers and manage their contributions, especially in the context of code samples?** In response, we conducted an exploratory study centered on external contributions to repositories by developers not affiliated with the code sample providing organization. Investigating these contributions is essential as it reveals organizational interest in the community, the community's intent to contribute to this artifact, and clarifies certain organizational practices in the contribution process.

Motivated by the exploratory study, we decided to further explore practices within organizations related to the examined artifact. We then formulated a second research question: **RQ2: What are the code sample practices within organizations?** To answer, we surveyed professionals directly involved with code samples in organizations, aiming to better understand their experiences, dedication, and the organization's internal perspective. Investigating these facets will help understand more about the process of developing and maintaining code samples within organizations, identify potential issues or best practices, and propose improvements.

We believe this work will enhance understanding from the organization's viewpoint regarding code samples. We hope our findings will spark reflections on improvements in the development, quality, and maintenance processes of code samples, offering insights for organizations seeking to leverage these artifacts in their ecosystems. Moreover, we hope this study will inspire future research to continue investigating organizational processes related to code samples and identify/reveal potential improvements.

The initial phase of our study resulted in a publication, with an article published in the *Workshop on Software Visualization, Evolution, and Maintenance* [31]. In Chapter 2, we provide the background of our work. In Chapter

3, we introduce our exploratory study. In Chapter 4, we present the survey conducted with professionals. Chapter 5 outlines the implications drawn from our findings. Chapter 6 reviews related works. Finally, in Chapter 7, we present our conclusions and directions for future research.

Background

2.1 *Software Ecosystem*

In the context of software engineering, there is a concept called software ecosystem or SECO, which consists of a set of actors capable of interacting with each other with the aim of developing and maintaining one or more software systems around a common technological platform [5, 20, 28, 57]. A set of actors can be divided into individuals and organizations [60], where individuals may or may not belong to an organization. The actors within a SECO, when interacting towards common goals related to a specific software, platform, or technology, form a community (including developers from the organization responsible for the SECO, external developers) [58].

According to definitions by Manikas et al. [28] and Wouters et al. (2019) [60], various actors have been identified in the context of SECOs, such as niche player, orchestrator, technology provider, platform provider, end user, etc. Although there are a variety of actors within a SECO, we highlight some directly related to our context:

- **Organization:** Entities like companies or foundations responsible for the SECO. It is important not to confuse with the individual actor inside the organization. For instance, Google is the organization responsible for the Android SECO.
- **Internal Developer:** Professionals directly associated with the organization that manages the SECO, in charge of the development and maintenance of products or services.

- **External Developers:** Individuals not linked to the central organization of the SECO who collaborate according to their own interests and objectives. They can identify flaws, propose improvements, and promote products.

In this work, we focus on the organization as an abstract entity that governs the SECO and offers artefact within their ecosystems. Moreover, our focus is on the professionals, both internal and external, who develop and maintain the artifact that we explored; we will call these individuals external or internal developers. In Figure 2.1, we illustrate an example of SECO. In this figure, we can see the actors that will be addressed in this study. We can also observe artifacts which may include source code, documentation, files, etc., that can be produced or consumed by these actors. Additionally, there are technological platforms such as blogs, websites, code hosting platforms, etc., which serve as mediums through which an ecosystem can expand, store artifacts, and bring together actors.



Figure 2.1: Example of SECO and Actors

Within the domain of mobile applications, an example of a software ecosystem are Android and iOS. In these ecosystems, there are professionals dedicated to the development and maintenance of these operating systems. Some individuals, known as internal developers, have direct affiliations with corporations such as Google or Apple (organizations) and are responsible for developing features, creating resources, drafting documentation, and providing services, in addition to directly contributing to the operating system. On the other hand, there are developers who, although not directly associated with these companies, depend on the resources, APIs, and functionalities offered by the ecosystem of these platforms to create, for example, games and apps

for these mobile operating systems. These latter are considered external developers, with the potential to engage in collaborative efforts within the SECO.

In some software ecosystem models, external actors can generate values that go beyond monetary compensations, such as knowledge, experience, or fulfilling needs [27]. To attract individuals capable of generating value in their ecosystems, organizations can create and provide artifacts, which are means of exchanging information among those involved in a SECO [57]. In short, artifacts can be the code of a software, components, services, models, documentation, and other elements that can interact with actors or other software [48].

Some developers may encounter difficulties for learning to use a product offered within the ecosystem, such as an API. At this point, some barriers arise, such as lack of motivation to read the documentation [55], difficulties in understanding and using specific functionalities [54], the need for rapid learning [68], and even documentation that tends to become outdated [24]. One of the artifacts that an organization can provide and that can be particularly useful to meet developers' needs, particularly when it comes to aiding in the understanding of an API, libraries, frameworks, and software functionalities in general, are code samples.

2.2 *Code samples*

Developers may, for example, when they have difficulties about a framework, a library, turn to the web. In such cases, some developers, in the absence of official documentation for a library or for other reasons, end up seeking answers on question and answer sites like Stack Overflow or other similar platforms. When seeking help online, they may come across code fragments that are copied from software projects or online sources to Stack Overflow, known as "code snippets" [43].

Many developers not only use these code snippets to learn or seek answers but also sometimes use them with the intention of code reuse. Although code reuse serves to accelerate development [45, 47] and is useful for developers [64], in the case of code snippets, they can bring vulnerabilities, outdated code, or code that violates the original license [43, 45].

Code samples may have similar purposes to code snippets, but they are not the same artifact. In the literature, we can find that code samples are not just code fragments (like code snippets) but are described as complete software projects aimed at assisting developers in understanding software products such as APIs, libraries, and frameworks, and they evolve over time undergoing maintenance to avoid becoming outdated [32, 33]. According to Menezes *et*

al. [32], it was found that code samples, despite usually being small and simple for better understanding and ease of reuse, provide a complete working environment with configuration files and other tools to facilitate the use of the code sample by the client. Studies have also shown that code samples are not only consumed by beginner developers. The range of developers includes novices to the most experienced, making them an important artifact in the community [6].

In the industry, we find some definitions as well. Oracle states that "code sample is provided for educational purposes or to assist your development or administrative efforts" [39]. Guidelines set by Mozilla highlight the didactic purpose, stating that codes should be clean and easy to comprehend, avoiding the inclusion of unnecessary libraries, files, or dependencies [9]. As for the Spring organization, it states that "code samples are designed to make you productive as quickly as possible".[53]. As for Microsoft, they serve to demonstrate things that the developer can build [35].

Figure 2.2 exposes the files of the *android-custom-lint-rules* code sample stored in the GitHub repository¹. Among the project files, the presence of configuration files can be noticed. Gradle files², which are automation tools for compilation and software building, can also be seen. The project also includes the README.md, a file that describes the project's objectives, and the LICENSE file indicating the license used³. In this example, it can be observed that code samples come with a configuration environment, including build files and usage licenses, demonstrating that they are not just code fragments like code snippets. Additionally, there are files like CONTRIBUTING.md⁴ that specify rules for developers to contribute to the project.

Several organizations make code samples available in their ecosystems. These code samples can be found on their official developer blogs/sites, and often the source code is made available on GitHub. We have examples from some organizations such as Google¹, Microsoft², Amazon³, Spring⁴, Oracle⁵, and Twitter⁶.

¹<https://github.com/googlesamples/android-custom-lint-rules>

²https://docs.gradle.org/current/userguide/what_is_gradle.html

³<https://www.apache.org/licenses/LICENSE-2.0>

⁴<https://github.blog/2012-09-17-contributing-guidelines/>

¹[https://developers.google.com/style/](https://developers.google.com/style/code-samples)

code-samples

²[https://docs.microsoft.com/pt-br/](https://docs.microsoft.com/pt-br/samples/browse/)

samples/browse/

³[https://docs.aws.amazon.com/](https://docs.aws.amazon.com/code-samples/latest/catalog/welcome.html)

code-samples/latest/catalog/welcome.html

⁴<https://spring.io/guides>

⁵[https://www.oracle.com/downloads/samplecode/](https://www.oracle.com/downloads/samplecode/developers-admins-samplecode.html)

developers-admins-samplecode.html

⁶[https://blog.twitter.com/developer/en_us/a/2015/](https://blog.twitter.com/developer/en_us/a/2015/twitterdev-code-samples-on-github)

twitterdev-code-samples-on-github

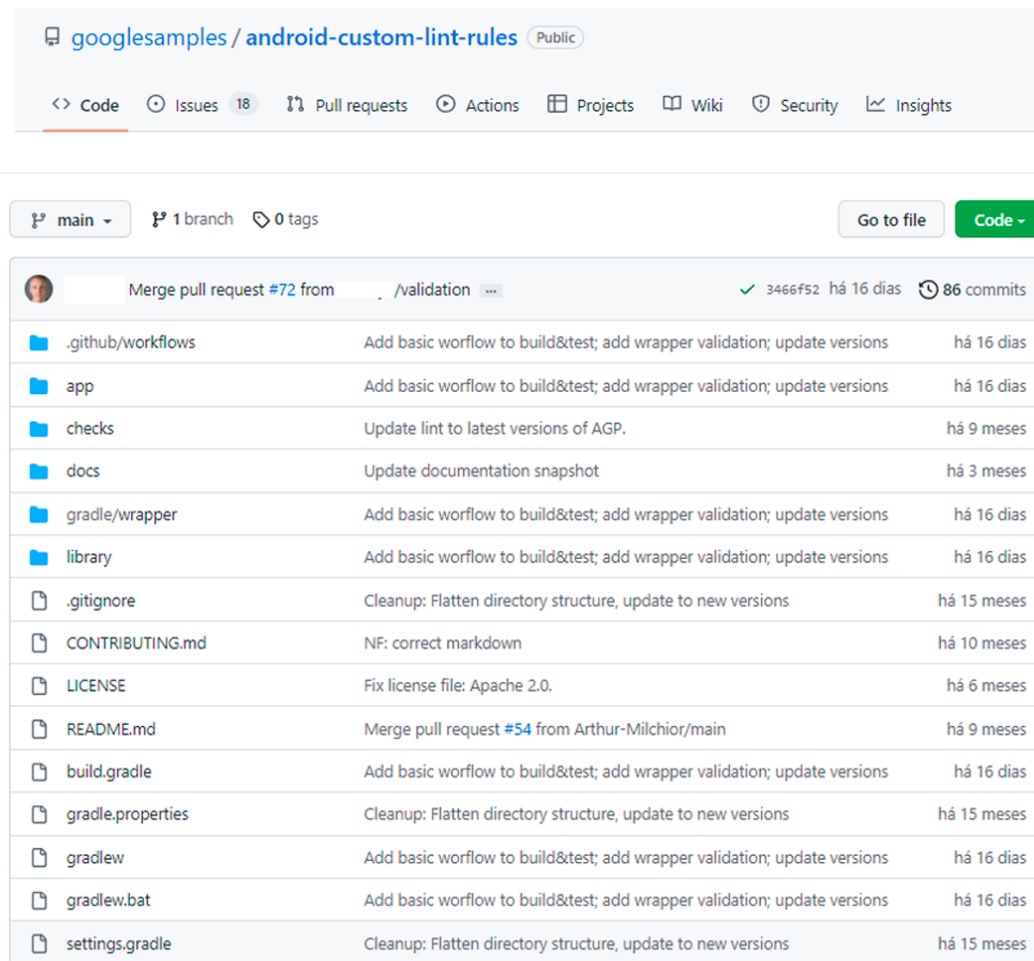


Figure 2.2: Files of *code sample* android-custom-lint-rules

Code samples that are officially provided by organizations with specific guidelines for contributions and code licenses might exhibit consistent quality. This suggests that code samples can be viewed as a more reliable alternative than online code snippets, which often violate license terms or come from untrustworthy sources.

2.3 Code samples, SECO and GitHub

According to Lima *et al.* [25], SECO approaches generally refer to an environment where a repository of components brings together stakeholders, as well as software products and components. The components of a SECO, such as source code, APIs, libraries, and documentation, are typically stored in repositories. These components are considered artifacts in the context of the study.

GitHub is the leading online code hosting platform where developers can

submit their own open-source projects and contribute to others [61]. With the growth and popularity of GitHub, organizations have started storing artifacts of their open-source software products in GitHub repositories. GitHub has become the technological platform that gathers the stakeholders of a SECO, including major tech companies like Google, Microsoft, and Amazon.

Code samples are artifacts that organizations can make available. It has been previously mentioned that various multinational organizations provide the source code of their code samples on GitHub. As code samples evolve over time, undergo maintenance, and are used by developers with various levels of expertise, it is necessary for actors to interact with each other for the benefit of these artifacts. GitHub helps bring together the stakeholders of code samples (organization's developers and external) and facilitates access. Through GitHub, developers can create forks (independent copies) of the source code of a code sample to learn from it, reuse code in their own projects, or contribute to code samples through pull requests or issues (requests for code incorporation or identified problems).

By understanding that one way to make code sample available is by storing them in repositories, such as on GitHub, it is necessary to understand some concepts present in these repositories that enable interaction among different actors within the code hosting platform, such as pull requests, issues, collaborators, contributors, and users.

- **Fork:** On GitHub, a "fork" is a copy of a repository made by a user to their own GitHub space. It allows users to contribute to projects without altering the original repository. Forks encourage collaboration, facilitate proposed modifications, and are useful for personal work;
- **Branch:** Git repositories are organized similarly to a tree. When creating a repository, the initial version is stored in the "master branch," which is equivalent to the trunk of a tree. A branch is a "branching" off the main tree of the project. Created branches can serve for version control, adding new features, separating a part in development from a stable part, etc;
- **Pull requests:** These are used to inform about changes that have been sent to a branch in a GitHub repository. When opening a pull request, it is possible to discuss changes with collaborators who are usually responsible for accepting or rejecting the requests. Pull requests can be made by both organization developers and external developers;
- **Issues:** Issues allow users to report problems or make comments about a project stored in the repository. It is also a means by which users can ask questions related to a project;


- **Merge:** This is the process of combining different branches and integrating them into a single branch. It is typically used to consolidate changes into one branch;
- **Commit:** A commit is a set of changes that add modifications to source code and files in a repository;
- **Contributors:** All individuals who have contributed commits to a repository and had their commits merged into the project's default branch. They can be either organization developers or external developers;
- **Collaborators:** Collaborators can be project maintainers or users who have administrative permissions on the repository. They can be either organization developers or external developers;
- **Member:** An individual who is part of an organization, with access to specific resources and privileges within it. Members can have different levels of permissions, added by owners or administrators of the organization;
- **Users:** Users are all those who have a profile within the platform. They can act as a collaborator, contributor, or both on a project. Typically, user profiles are associated with the number of followers/following and some personal information such as their organization, email, location, etc.


It is possible to observe part of the functioning of a SECO by observing repositories on the GitHub platform. In the following Figure 2.3 and Figure 2.4, we can observe the interaction between actors, where organizations, through their internal developers, interact with external developers in favor of artifacts. In the Figures, we can see the interaction between a internal developer and a external developer who opened an issue and another pull request within the code samples repository called glass-enterprise-samples.

In the issue of Figure 2.3, the external developer opens a question in the context of the repository, and the organization's developer responds three days after the issue is opened, providing an explanation and guidance. Then, the issue is closed by the organization's developer, and the external developer reacts to the response with a heart emoji.

In Figure 2.4, an external developer attempts a pull request to contribute to a code sample, which is addressed by a Google collaborator *bot* that guides the developer in their first contribution to the repository, asking them to sign a Contributor License Agreement (CLA) for their request to be reviewed. Subsequently, the developer confirms that they have completed the indicated steps as instructed by the *bot*, receiving a thank-you message in return. Finally, it

May I use these samples with Glass Explorer Edition? #33

 Closed opened this issue on 8 Jan 2021 · 1 comment


 commented on 8 Jan 2021

I know that you've system images here: <https://developers.google.com/glass-enterprise/downloads/system-images> and EE1 to EE2 update instructions here: <https://developers.google.com/glass-enterprise/guides/get-started>. Would it be okay/possible for me to update my explorer version to the new system images?


I'm also aware of the different specifications:

Google Glass Enterprise Edition	Google Glass Enterprise Edition 2
<ul style="list-style-type: none">• Intel Atom processor• Dual-band 802.11n/ac Wi-Fi• Assisted GPS & GLONASS• Barometer• 32GB storage memory• 780mAh battery• Dynamic driver speaker instead of bone conduction audio transducer	<ul style="list-style-type: none">• Qualcomm Snapdragon XR1. Quad core up to 1.7GHz, 10nm• Android Oreo with Android Enterprise Mobile Device Management• 3GB LPDDR4• Bluetooth 5.x AoA• 8MP 80° FOV camera• 3 beam-forming microphones• USB Type-C port supporting USB 2.0 480Mbps• 820mAh battery with fast charge• 6-axis accelerometer/gyroscope• On head detection sensor and Eye-on screen sensor for power-saving features• Water and dust resistant• ~46g weight

I have also successfully followed the instructions here: <https://support.google.com/glass/answer/9649198?hl=en>. Thank you :)

 commented on 11 Jan 2021 Collaborator

Hi @ , sorry but the explorer and enterprise products are different and using the newer system images for EE2 will not work. Similarly, these samples are intended for EE2 developers, but let us know if you are looking for anything specific and we might be able to help.

 1



  closed this as completed on 11 Jan 2021


Figure 2.3: Example of interaction: A internal developer (Google employee) responding to an issue from a developer

can be seen that an organization's developer closes and merges (*merged*) the pull request, accepting the external contribution.


Wrong variable naming in example #30


Merged merged 1 commit into `googlesamples:master` from `...:master` on 10 Sep 2020

Conversation 4 Commits 1 Checks 0 Files changed 1


 commented on 10 Sep 2020 Contributor ...

Updated variable in example file so the code works without errors.

 wrong variable naming in example Verified ✓ f552017

 googlebot commented on 10 Sep 2020 ...

Thanks for your pull request. It looks like this may be your first contribution to a Google open source project (if not, look below for help). Before we can look at your pull request, you'll need to sign a Contributor License Agreement (CLA).

 Please visit <https://cla.developers.google.com/> to sign.

Once you've signed (or fixed any issues), please reply here with `@googlebot I signed it!` and we'll verify it.


What to do if you already signed the CLA


Individual signers

- It's possible we don't have your GitHub username or you're using a different email address on your commit. Check [your existing CLA data](#) and verify that your [email is set on your git commits](#).


Corporate signers

- Your company has a Point of Contact who decides which employees are authorized to participate. Ask your POC to be added to the group of authorized contributors. If you don't know who your Point of Contact is, direct the Google project maintainer to [go/cla#troubleshoot \(Public version\)](#).
- The email used to register you as an authorized contributor must be the email used for the Git commit. Check [your existing CLA data](#) and verify that your [email is set on your git commits](#).
- The email used to register you as an authorized contributor must also be [attached to your GitHub account](#).


 Googlers: [Go here](#) for more info.


 commented on 10 Sep 2020 Contributor Author ...

`@googlebot I signed it!`


 googlebot commented on 10 Sep 2020 ...

CLAs look good, thanks!

 Googlers: [Go here](#) for more info.

 commented on 10 Sep 2020 Collaborator ...

Thanks for the fix!

 merged commit `87042ee` into `googlesamples:master` on 10 Sep 2020 View details


 1 check passed

Figure 2.4: Example of interaction: A bot and an internal developer (Google employee) responding to a pull request from an external developer.

Exploratory study in code sample repositories

3.1 *Introduction*

As described in Chapter 2, code samples are provided within a context called software ecosystems. Within these ecosystems, there are interactions between actors. In some SECO models, external developers can add value to an organization's product, and the organization's engagement to attract contributions from such actors is essential. Some code samples are made available through repositories on GitHub, and within this platform, interactions between internal and external developers can occur through pull requests and issues. Pull requests need to be reviewed to be approved or rejected [44], and one way organizations engage is by allocating developers to this review activity, thus fostering interaction.

Pull requests can be analyzed through repository mining. Within this domain, several studies have investigated pull requests without specifically focusing on code sample repositories [36, 52, 66, 67]. These studies do not address the interaction between the organization and the community concerning code samples. In our exploratory study, we analyzed pull requests within code sample repositories hosted on GitHub.

Our goal is to determine if organizations are receiving contributions from external developers (assessing their engagement) and understand if they are open to reviewing and accepting such contributions, as well as the time it takes to review and distribute tasks among maintainers. The lack of contribu-

tions from external developers might suggest that organizations might not be receptive, or there's limited community interest or incentives for such contributions. This can affect the maintenance and utilization of this artifact. On the other hand, the presence of external contributions and organizational engagement gives us a evidence of the relevance of the study artifact for both the organization and the community, revealing findings about the organizations' practices in these repositories.

To guide our goal, we defined the following research questions: (RQ1) Do organizations review and accept external contributions in code samples? (RQ2) How long does it take organizations to review contributions? Is there a difference in time between accepted and rejected contributions? (RQ3) How do organizations distribute pull request review tasks among code sample maintainers? To answer these questions, we conducted an exploratory study evaluating around 12,000 pull requests from 2,179 code samples of Android, AWS, Azure, and Spring.

3.2 *Study Design*

3.2.1 *Study Scenario*

In previous studies on code samples, it has been observed that the most common problem faced by the audience consuming this artefact is when they try to modify the code samples [33]. It was also observed that the most recurring demand from audience consuming code samples is related to improving this artefact, whether improving source code, documentation, or supporting tools [33]. Given a scenario where external developers consuming code samples might find issues and attempt to contribute to repositories of this artifact, organizations can leverage these contributions to generate value within the SECO. To maintain a successful SECO, organizations need to meet the ecosystem needs and use business or motivation to encourage actors to contribute to the ecosystem evolution [27, 28]. Pull requests allow external developers to interact and contribute with code samples, requesting changes to their repository [16, 52].

On the one hand, organizations can benefit from receiving contributions from their external developers, this can improve their interactions and engagement, which is good for a healthy SECO. On the other hand, it is essential that these contributions be reviewed since code review has a significant impact on software quality, it is crucial to review as many submitted changes as possible [30]. If changes with low-quality code or defects are incorporated into code sample repositories, it can have a negative impact on those who want to learn from this artifact or copy parts of it into their projects. In this study, we

aim to explore how organizations deal with requests to change code samples repository through pull requests on GitHub. In addition, we aim to explore how are distributed the activities from pull requests management between the code sample maintainers.

3.2.2 Chosen Platform and Organizations

We selected code samples from four platforms: Android, AWS, Azure, and Spring Boot. The following reasons motivated us to select these platforms: (1) they are relevant and have a wide range of clients; (2) they support the creation of different application niches, such as mobile applications, web applications, and cloud computing; (3) their code samples are publicly available on GitHub; (4) some of these platforms have already been explored in previous works [32, 33, 58]. Although there are other large platforms that provide code samples, we have identified 4 that cover the mobile and cloud computing niches.

- **Android:** Android is an open-source mobile operating system owned by Google, serving over 1300 brands and more than 24,000 device models worldwide. There are over 2 million apps developed for this platform ¹. As such, the platform is widely used by people from all over the world, attracting developers from diverse regions. The code samples available on this platform assist developers in creating applications with exemplifications of resource usage, APIs, etc.
- **AWS (Amazon Web Services):** AWS is a cloud computing services platform owned by Amazon, catering to organizations like Lyft, Netflix, Coca-Cola, and Moderna ². Besides some free services, the platform offers a wide range of cloud computing products. The code samples provided in this ecosystem can assist developers from various client organizations.
- **Azure:** Azure is a cloud platform that competes with AWS and is owned by Microsoft. They offer over 200 cloud computing products and services with clients spread worldwide³. The code samples available on this platform, similar to the previous ones, can be useful for developers across various client organizations of the platform.
- **Spring Boot:** Spring Boot is a platform belonging to the Spring ecosystem. It is an open-source framework that facilitates the creation of standalone applications ⁴. Several organizations and developers produce ap-

¹,https://www.android.com/intl/pt-BR_br/everyone/

²<https://aws.amazon.com/pt/products/compute>

³<https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-azure/>

⁴,<https://spring.io/projects/spring-boot>

plications based on this framework, its language is JAVA, which made it even better known.

3.2.3 Code Sample Selection

We select code samples from the official list provided by their organization on GitHub. The code sample selection was performed, as well as the extraction of all other metrics, through Python scripts. We used PyGithub library⁵. PyGithub encapsulates the functionality of the GitHub API⁶ and provides it with Python functions. As a result, we obtained a total of 2,179 code samples, 44 from Android, 1,047 from AWS, 1,007 from Azure, and 81 from Spring Boot. All data and scripts are available publicly.⁷

3.2.4 Pull Request Selection

Our study focuses on contributions from external developers in code samples, which are then reviewed (those that were accepted (merged), those that were rejected (only closed)) by the organization, so we should not select pull requests created by maintainers (considered internal developers in this study) or code samples that have been closed by their creators. For that, in the first step, we select pull requests of the selected code samples. Second, we remove pull requests created by maintainers of the code sample (details about the maintainers are presented below). Finally, we removed pull requests that were closed by their own creators.

3.2.5 Maintainers Selection

We consider a code sample maintainer (organization developer) any GitHub user who *accepted* (performed the merge into the project's branch) at least one pull request (which was not created by himself) to a repository of the studied code samples.

3.2.6 Research questions

(RQ1) Do organizations review and accept external contributions in code samples? To answer this question, we compute four metrics: number of *reviewed* pull requests, number of *unreviewed* pull requests, number of *accepted* pull requests and number of *rejected* pull requests. we consider a pull request to be *reviewed* when its state changes to closed and *unreviewed* when

⁵<https://pypi.org/project/PyGithub/>

⁶<https://docs.github.com/pt/rest>

⁷<https://github.com/MatheusM97/VEM2022>

its state is open. We consider that a pull request has been *accepted* when its state changes to closed and it has been merged [36, 52]. And we consider that a pull request has been *rejected* when its state changes to closed but it has not been merged [36, 52]. *Rationale:* If there is a high proportion of unreviewed pull requests, it may indicate that organizations are not worried about reviewing them or have not been maintainers enough to attend to the needs of the community. This may discourage external developers from contributing to the evolution of the code sample. In addition, if we find a low proportion of accepted pull requests, it may indicate that organizations are not open to receiving contributions, possibly from external developers, in their code samples.

(RQ2) How long does it take for organizations to review contributions? Is there a time difference between accepted and rejected contributions? To answer this question, we selected three metrics: *time to review*, *time to accept* and *time to reject*. *Time to review* is computed by the difference between the date pull requests became *closed* and its creation date. *Time to review* consider both, accepted and rejected pull requests. *Time to accept* is calculated by the difference between the date pull requests became *closed* and its creation date, but only considers accepted pull requests. Similarly, *time to reject* is calculated by the difference between the date pull requests became *closed* and its creation date, but considers only rejected pull requests. *Rationale:* If the *time to close* is too high, this may deter external developers to contribute, as they don't see their contributions being inserted or even reviewed by code sample maintainers. By another view, if the *time to close* and the *time to accept* shows a high value, it could be an indication that maintainers took more time to review the pull requests may be due to a detailed review in order to reduce the insert of low-quality code or even errors in the code sample.

(RQ3) How do organizations distribute the activities of reviewing contributions via pull requests among code sample maintainers? Given the set of code sample maintainers, we compute the number of pull requests *reviewed* performed for each maintainer. Next, we compute the Gini coefficient [15] along with the Lorenz curve [26]. The Gini Coefficient is a statistical measure that represents the inequality of a distribution, a value of 0 expresses total equality (for instance, everyone has the same income), and a value of 1 expresses maximum inequality. The Lorenz Curve is a graphical representation of the proportion of review activities and the proportion of maintainers. The straight line represents perfect equality, while the Lorenz Curve indicates the degree of inequality. The area between the line of equality and the Lorenz Curve is used to calculate the Gini coefficient. The adoption of the Gini Coefficient and the Lorenz Curve allows us to assess the inequality in the distri-

bution of pull request reviews among the code samples’ maintainers. These metrics provide insights on whether reviews are being carried out equitably among the maintainers or if they are concentrated on a few individuals. *Rationale:* In this question, we intend to analyze the distribution of activities among the maintainers. An excessive concentration can indicate a possible workload overload for certain maintainers and, consequently, delays in the review process. By understanding this distribution, we can identify potential bottlenecks and areas for optimization in the review process.

3.3 Results of exploratory study

This section presents the obtained results. First, we present the results of pull request states (RQ1). Next, we present the results of the time to review code sample pull requests (RQ2). Finally, we present the results of the distribution of pull requests review (RQ3).

(RQ1) Do organizations review and accept external contributions in code samples? Table 3.1 presents the number of *reviewed* and *unreviewed* pull requests from code samples throughout the entire sample period. We found 483 pull requests for code samples from Android, 123 (25.5%) *unreviewed* and 360 (74.5%) *reviewed*. For AWS samples, we found 6,194 pull requests, 1,166 (18.9%) *unreviewed* and 5,028 (81.1%) *reviewed*. For Azure samples, we found 4,933 pull requests, 1,579 (32.1%) *unreviewed* and 3,354 (67.9%) *reviewed*. For Spring samples, we found 1,150 pull requests, 150 (13.1%) *unreviewed* and 1,000 (86.9%) *reviewed*. These results show that most pull requests have been reviewed by organizations. In any case, there is a non-negligible percentage of pull requests unreviewed, and organizations must show efforts to review them.

Table 3.1: Unreviewed vs Reviewed Pull Requests

Project	Unreviewed	Reviewed	Total
Android	123 (25.5%)	360 (74.5%)	483
AWS	1,166 (18.9%)	5,028 (81.1%)	6,194
Azure	1,579 (32.1%)	3,354 (67.9%)	4,933
Spring	150 (13.1%)	1,000 (86.9%)	1,150

Table 3.2 presents the number of *accepted* and *rejected* pull requests from code samples throughout the entire sample period. For Android samples, we found 292 (81.1%) *accepted* pull requests and merged to code samples repository and 68 (18.9%) *rejected* pull requests. In AWS samples, we found 4,422 (87.9%) *accepted* pull requests and 606 (12.1%) *rejected* pull requests.

For Azure samples, we found 2,901 (86.4%) *accepted* pull requests and 453 (13.6%) *rejected* pull requests. In Spring samples, we found 705 (70.5%) *accepted* pull requests and 295 (29.5%) *rejected* pull requests. These results seem to show that most *reviewed* pull requests are *accepted*. At least 70% of them, but reaching cases with 87% of *accepted* pull requests. In general projects (which are not necessarily code samples), have approximately 76% acceptability rate [36]. This value is exceeded in the code samples of 3 out of the 4 analyzed platforms. The fact that the majority are reviewed, coupled with the acceptance rate of pull requests, may indicate the organizations' management of these external pull requests and their receptiveness to this type of contribution.

Table 3.2: Accepted vs Rejected Pull Requests

Project	Accepted	Rejected	Reviewed
Android	292 (81.1%)	68 (18.9%)	360
AWS	4,422 (87.9%)	606 (12.1%)	5,028
Azure	2,901 (86.4%)	453 (13.6%)	3,354
Spring	705 (70.5%)	295 (29.5%)	1,000

In general, organizations demonstrate openness to reviewing contributions through pull requests in code sample repositories. A higher number of contributions have been accepted and incorporated into the code samples, regardless of the period analyzed.

We can conclude that organizations interact with external developers through reviews; they review and accept the majority of these pull requests. On the other hand, with the existence of these external contributions, the interest of external developers in contributing to the artifact was demonstrated.

(RQ2) How long does it take for organizations to review contributions? Is there a time difference between accepted and rejected contributions?

Figure 3.1 presents the time taken to pull requests be *reviewed*. We can see that, in the median, Android samples take 7.5 days to be *reviewed*. While pull requests from AWS samples take 3.6 days, in the median. Pull requests from Azure samples take 12.73 days and Spring samples take 32.58 days on the median. This indicate that there is no standard across organizations, while pull requests for code samples from AWS typically take approximately 3 days, Spring pull requests take more than 30 days to be reviewed. However, on all platforms, there are pull requests that take longer than others, and there are even cases that take more than 500 days to review. This may be due to the complexity of the modifications, given that the number of lines added or

modified and the number of commits is an attributes to determine the time to review [65].

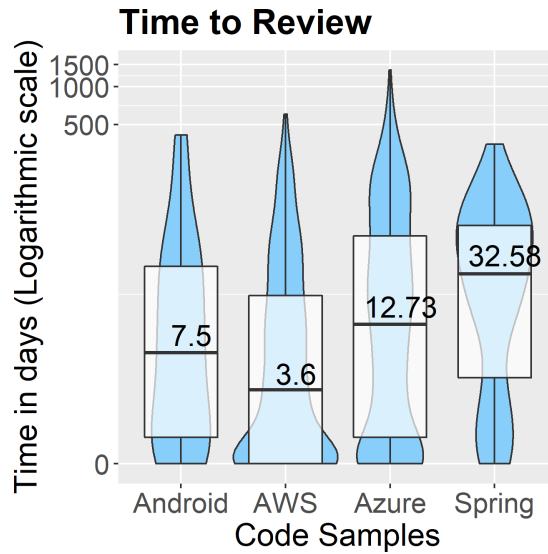


Figure 3.1: Time spent to review code sample pull requests.

Figure 3.2 (left) presents the time taken to review pull requests *accepted* into code sample repositories. We noted that in Android samples, in the median, pull requests took 6.61 days to be *accepted*. In AWS, was 2.67 days, on the median. While in Azure samples were 7.75 days and Spring samples were 7.54 days, in the median. Figure 3.2 (right) shows the time taken to review pull requests *rejected*. For Android samples, we noted that, in the median, was spent 24 days reviewing *rejected* pull requests. In AWS samples, pull requests took 29 days, in the median. While Azure and Spring present 75.5 and 108.38 days respectively, in the median.

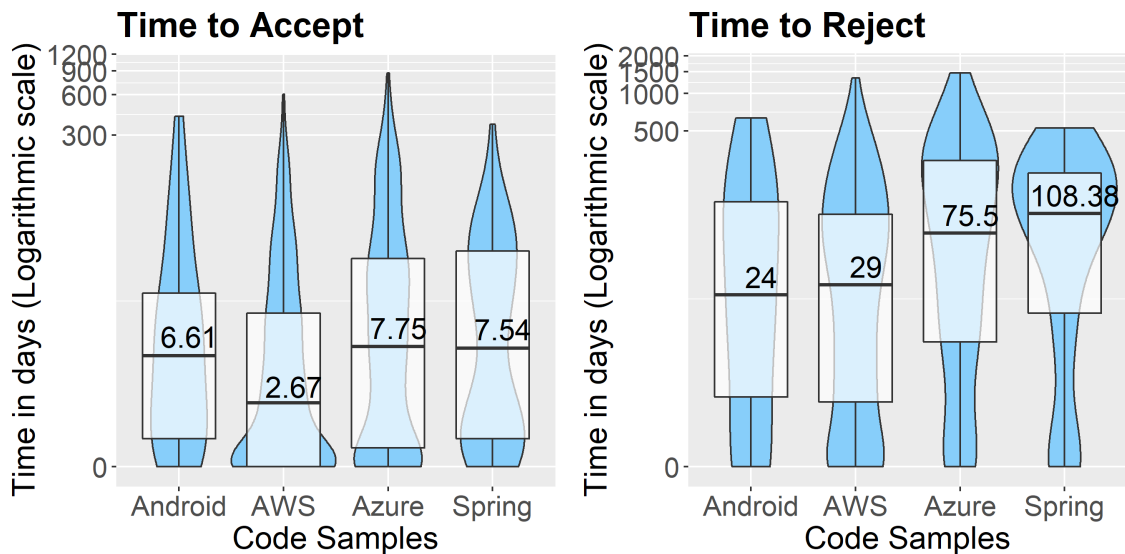


Figure 3.2: (Left) Time to review accepted pull requests. (Right) Time to review rejected pull requests.

We found that, in general, accepted pull requests take less time to be reviewed than rejected pull requests. This behavior is also observed in other projects (projects in general, which are not necessarily code samples) [17]. Pull requests with code that does not conform to the project's coding style tend to be rejected [18], and pull requests that are out of compliance may remain open for a longer time while waiting for the necessary changes, and if the pull request author does not make the required changes, they may be rejected. Other factors that can contribute to longer review times include proposing a new feature that is not accepted by the project (or conflicts with other features) [29], leading to extended evaluation periods and, consequently, rejections.

The time to review varies by organizations and repositories, but the results show that rejected pull requests take longer to be closed.

(RQ3) How do organizations distribute the activities of reviewing contributions via pull requests among code sample maintainers? Figure 3.3 presents the Lorenz curve for the relation between code sample maintainers and the number of pull requests reviewed. In addition to the Lorenz curve, we also compute the Gini coefficient. The values were 0.8 for Android, 0.91 for AWS, 0.91 for Azure, and 0.82 for Spring. These results display an inequality in the distribution of activities among maintainers. Through the coefficients, we can observe that the values are closer to 1 than to 0, which would represent complete equality.

The Lorenz curve shows a deviation from the straight line that would represent an egalitarian distribution, indicating that the pull request review activity is primarily carried out by a small group of maintainers across all analyzed organizations. This behavior can be explained because the pull request review task requires experienced maintainers and organizations are concerned about selecting only competent maintainers for this task. Consequently, having the review power in the hands of a small group of maintainers might be an organizational strategy, where this group is allocated due to their familiarity with the contribution guidelines and standards, and their experience with code samples. This can ensure consistency in the artifact's quality.

However, as observed in the results of RQ2, there are certain reviewed pull requests that take more than 500 days to reach their final review. When there's a large number of pull requests pending review and instances where the reviews are prolonged, an uneven distribution with reviews concentrated in the hands of a small group of maintainers might lead to bottlenecks. Organizations should be mindful of this potential issue.

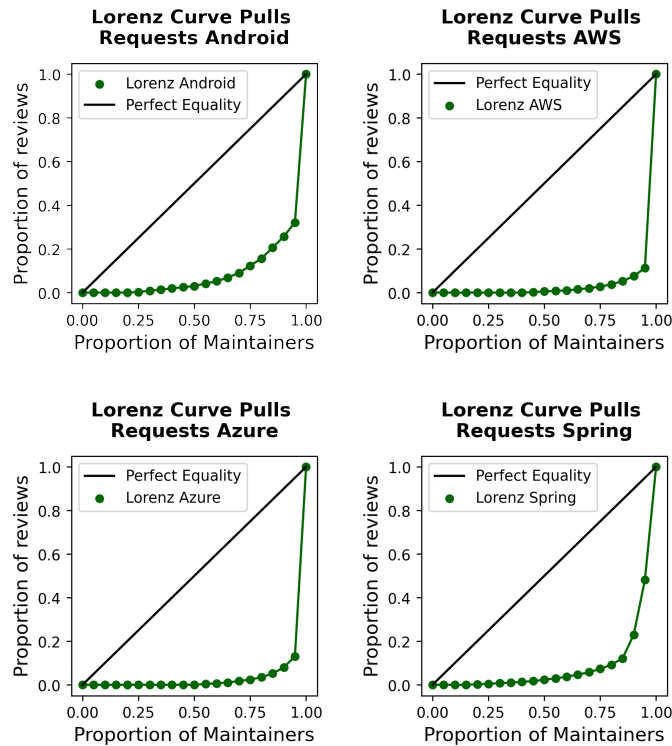


Figure 3.3: Lorenz curve between code sample maintainers and pull request reviews.

We observed that there isn't an equal distribution of review activity among the maintainers in the analyzed organizations. Only a subset of the maintainers is responsible for accepting or rejecting the majority of the pull requests.

3.4 Implications

3.4.1 The community matters

The results show that the majority of the pull requests received are accepted, reaching high values of 70.5% in the case of Android and over 87% in the case of AWS, indicating the interest of organizations in receiving contributions from external developers who are part of their SECO community, interacting with them through this medium.

These interactions among the actors are as important as interactions with the platform and play a crucial role in the survival of SECO [27]. The data shows that organizations recognize the importance of the community and contributions. Therefore, it is essential for organizations to devote efforts to encourage and increase these interactions and contributions.

3.4.2 *To each organization, its rules*

We observed different review times for pull requests among organizations, indicating that each organization may handle contributions differently, with some having a larger number of reviewers than others. Organizations might require adherence to the project's code style, acceptance of 'Contribution Licensing Agreement' (CLA) terms, or demand associated test files. When we analyzed some repositories to gain insight into the rules, we encountered requirements specified in the 'CONTRIBUTING' files, detailing contribution guidelines. In other words, each organization can have its own set of rules; some might be less open, while others might be more accommodating, having fewer requirements.

We recommend external developers who are interested in contributing to always follow the instructions of each organization to avoid having their pull requests rejected. It is important to ensure that their changes have not been previously resolved by other contributors, thus avoiding the insertion of duplicate or obvious pull requests, which could consume the organization's time and result in rejection. Adhering to the code style is crucial, as studies have shown that contributions with a different code style from the project tend to be more likely to be rejected [18]. External developers may test their contributions and including test files to mitigate the chances of their contributions being rejected.

3.4.3 *Rejected Pull Requests: The 'Time Rejectables' Chronicles*

We have observed that, in general, rejected pull requests take more time to be reviewed and closed, not being an exclusive issue of pull requests from code sample repositories, but rather a common problem in repositories overall [17]. We have noticed some situations where rejected pull requests fit into: the first situation was when reviewers suggested changes, but contributors took a long time to respond and did not address the "rule inconformities" or the reviewers' suggestions. The second situation was when contributors did not sign the CLA, leaving the pull request open for an extended period. Another situation was when no one opened a review, and after some time, the author themselves closed the pull request.

In some of these situations, we found automated bots pointing out the need to sign the CLA or informing the remaining time for contributors to comply or fix their pull requests, otherwise, they would be closed. Establishing deadlines for contributors to correct their pull requests or comply with the organization's rules seems to be a way to prevent pull requests from staying open for too long and, consequently, being rejected (rejected for being ignored).

Some practices to mitigate prolonged open pull requests and save developers' time include using bots to identify inconformities and using static code analysis tools to automatically detect problems with code style, formatting, vulnerabilities, and other programming best practices. Additionally, to prevent contributors from closing their pull requests on their own before a review, it is crucial for responsible professionals to provide prompt feedback.

3.4.4 *Expand to conquer*

In our observations, we noticed a pattern among organizations where only a portion, a small group of maintainers, were responsible for the majority of pull request reviews. Even so, having a small group of developers might be a strategic choice by the organization (and not necessarily a problem) due to their knowledge of the code samples or contribution rules. However, in situations where there's a high demand for pull requests to review, or if the reviewer is engaged in other organizational tasks, it can create a bottleneck in the review activities. We observed pull requests that took months or even years to be reviewed. Of course, for this phenomenon, there might be other causes beyond the organization's maintainers. Such delays might discourage external developers from making frequent contributions.

To mitigate this potential bottleneck, we recommend organizations take steps to more equitably distribute review tasks among their group of maintainers. By increasing the number of maintainers (as needed based on pull request demand) and improving the distribution of activities among them in the review process, organizations can benefit in two ways. First, pull requests could be processed more swiftly, ensuring that updates and fixes are integrated without unnecessary delays and avoiding contributor drop-offs. Secondly, a more efficient and responsive review system could act as a motivating factor for external contributors, encouraging them to contribute more frequently to the project.

3.5 *Threats to validity*

We will discuss the validity threats of the study according to the main threats described by Wohlin [59]. We identified these possible threats to our study and allocated them to the most appropriate category.

External Validity: To try to cover the maximum of generalization, we focused on studying only globally recognized ecosystems like Android, Spring, AWS, and Azure, to reach the largest number of contributors and achieve greater diversity in the sample, as people from all over the world can contribute to these projects.

Construct Validity: It is not necessarily certain that the way we filtered external developers from organization developers (maintainers) is fully effective. We cannot validate with certainty that all members of the organization were excluded since there might be organization members who are not defined as members or collaborators on the repository and not have administrative powers to perform merges in the projects. On the other hand, organizations may adopt external members to maintain their repositories.

Conclusion Validity: It is important to consider that the delay in reviewing pull requests can be influenced by various factors, and it is not always directly related to the size of the maintainer group. Although we identified a small group of maintainers responsible for the majority of pull request reviews, this does not necessarily reflect the situation where other developers do little work. They may have different reasons for conducting just a few reviews. For example, over time, some maintainers may have left the organization or are no longer responsible for the code sample in question. To mitigate this and avoid erroneous conclusions, we filtered a shorter time period to see if the concentration of review activities among maintainers would still persist.

3.6 *Final considerations*

In this chapter, we addressed pull requests as a tool to analyze the interaction between organizations and external developers. The results revealed that, in general, organizations not only review but also accept contributions through this medium. This interaction can be crucial for the maintenance of code samples and for the SECO. We identified that organizations have varying review times and rejected pull requests take longer to be reviewed. A third observation was that the majority of reviews are in the hands of a small group of maintainers. For future studies, we might suggest examining external developers to determine if the time identified for review in this study are sufficient to encourage them to contribute. Other studies could also delve deeper into whether the contribution rules specified in each code sample influence the acceptance or rejection rate as well as motivation for new contributions.

The active engagement of organizations with the community through pull requests and the adoption of practices such as reviewing these requests motivates us to better understand the organizations' side in the context of code samples. The motivation led us to try to know more about these internal developers, their experiences, dedication and perspectives regarding this artifact.. In the next chapter, we will present data from a survey conducted with professionals from the organizations studied in this chapter.

Survey with professionals who produce code samples

4.1 *Introduction*

In the previous chapter, our exploratory study with pull requests in code sample repositories showed that code samples have become relevant in recent years, receiving numerous pull requests from external developers. The organizations studied have shown interest in receiving contributions from the community in these repositories, investing time and effort in reviews and allocating professionals for these activities.

As per Chapter 2, we saw that in the literature we find some definitions of code samples, even definitions from organizations, and we also identified the audience that consumes the code samples [6]. Therefore, we did not find studies that showed the practices within organizations and perspectives of internal developers about code samples; for example, we do not know if internal developers perceive differences in the target audience or have aligned objectives regarding code samples.

In this chapter, we conducted a survey with internal developers, whom in this chapter we will refer to as organization professionals or participants to avoid confusion with the developer role within the organization. Professionals from the 4 organizations studied in the previous chapter were invited to the survey. To the best of our knowledge, this will be the first study to apply a survey in the context of code samples among professionals involved with this artifact. Our goal is to investigate the following aspects in this study:

1. What are the objectives of the code examples for the organization's professionals?
2. In the perspective of professionals, to which audiences are the code sample projects directed within the organizations?
3. Are there similarities with other products of the organizations in terms of quality assurance and development process?
4. What is the experience and roles of professionals who work with this artifact?
5. How do they dedicate to the development of code samples?

We hope to understanding practices of how organizations are creating and maintaining code samples, through the responses of these participants, expecting to identify possible issues with the practices and internals of the organizations in order to propose improvements.

To facilitate the conduct of the study, we defined the following research questions:

- (RQ1) What is the role of individuals working with code samples within organizations, and what practices are associated with this work?
- (RQ2) How does the experience of individuals working with code samples influence the frequency at which they engage in this activity?
- (RQ3) What is the underlying purpose of providing code samples within organizations, and how are these purposes perceived by the professionals involved?
- (RQ4) How do the development, quality assurance practices of code samples compare to similar practices applied to other products within organizations?

4.2 *Study Design*

4.2.1 *Target population of survey*

We selected professionals who work or have worked on code sample repositories stored on GitHub belonging to the following platforms: AWS, Android, Azure, and Spring-Boot, as described in Chapter 3. Moreover, our target was solely professionals from the organizations that own the platforms where the code samples were hosted.

4.2.2 Criteria for Participant Selection

We utilized the "Purposeful Sampling" technique to select the sample for the survey. In this type of sampling, we identified and selected individuals or groups with knowledge or experience related to the research's phenomenon of interest [41, 50]. In our case, we sought individuals who had worked (created and/or responded to issues, made pull requests, including creation, response, and/or merging, created tags and/or releases) with code samples and were professionals with the analyzed organizations.

We only included professionals who had engaged in the aforementioned activities within the last two years (from April 10, 2021, to April 10, 2023), with the intention of increasing the probability of reaching individuals who have actively worked with code samples recently. To specifically filter for professionals from the organizations, we sought out users who were associated as either a *MEMBER* or *COLLABORATOR* in at least one code sample repository of the organizations. This was done as a way to help filter out users who had no affiliations with the organization. This was necessary because the parameter on GitHub, which determines if a user is part of an organization via the repository, is private and accessible only to members of that organization. Also, to ensure that we invited only those truly part with the organizations, we not only checked for corporate email addresses but also cross-referenced their GitHub and/or LinkedIn profiles. By analyzing the organization each individual declared in their profile, we could confirm their affiliation. This manual verification was conducted for each candidate before initiating the pilot, and for the rest of the candidates one week before dispatching the final survey.

We chose not to include individuals who did not have an email linked to their GitHub profile in our sample, since the email was not public, we decided not to disturb the users by other means. Additionally, we decided to exclude highly popular users, often referred to in the literature as "rockstars." A common way to identify them is by their follower count [19, 63]. It has been observed that the more followers these users have, the more they actively participate in events and influence their followers [19, 23]. We strategically decided to exclude these popular users since they are often more sought after than less popular ones, and it can be more difficult to obtain responses from them. We focused on less popular users, specifically those with fewer than 323 followers (median + standard deviation of the sample). Our initial sample consisted of 475 candidates.

4.2.3 Pilot survey

We initially conducted a pilot study with two surveys, each being a subset of the other. One had a greater number of questions, and we sent them to 20 random candidates (10 per survey) via email, with a link to a Google Forms. Our aim was to identify questions that might be misinterpreted or from which we wouldn't obtain responses or the expected responses. We received feedback from three participants, who suggested new options for multiple-choice questions and highlighted questions that could potentially violate the N.D.A (Non-Disclosure Agreement) contract [51] according to respondents. Therefore, we improved the phrasing of some questions and removed ones that went unanswered or were flagged as potential N.D.A violations.

4.2.4 Final survey

In the final version of our survey, we included a total of 14 questions. Among them, one demographic question, eight multiple-choice questions related to the role, experiences, and dedication of the participants regarding code samples, and four questions about the participants' views on code samples. The applied form can be found in the footer¹.

We did not consider any response nor did we resend the survey to any candidate chosen in the pilot. Our study invited 455 individuals via e-mail containing a link to the Google Forms with the questions from May 30, 2023, to June 6, 2023.

Survey questions

The first question was demographic in nature: "Which country do you live in?" where we provided an open field for users to respond. Regarding the questions about professional experience, we proposed: "How many years have you been working in software development?", "How many years have you been working with code sample repositories? (If you no longer work with code samples, please consider the number of years you have worked with them.)", and "Have you previously worked with code samples in other organizations? How many organizations?". We utilized the criterion of years of work related to the artifact or software development, which has been employed as a metric of experience in other studies [8, 56, 62]. We drew from scales of professional experience in years documented in other studies [7, 11, 50], using a year-based scale to facilitate and expedite the survey response process. We set the following response options: "up to 1 year", "between 2 and 5 years", "between

¹https://docs.google.com/forms/d/e/1FAIpQLSdzqJUIss-6wU_1amTOAA-c00mAg7IT0ROnNP9EduTuWyUe_w/viewform

5 and 10 years", and "more than 10 years", which were employed for the first two questions about years of work. For the question about how many previous organizations the respondent had worked with code samples, we offered the choices "none", "1", "2", "3", "4", and "5 or more".

To determine their roles, we asked, "What is/was your role within the organization when working with code samples?". We provided participants with options to identify as developers, software architects, quality analysts, program managers, and some roles related to developer relations that were highlighted in other studies [14, 38]. We also included additional options suggested by respondents during the pilot tests. Each participant could select only one role, provide further details, or input a role not listed in the form via the 'other' field.

Regarding dedication, we assessed the number of code samples that professionals typically work on at the same time with the following question, "How many code samples do you typically work on simultaneously?" We provided participants with 5 multiple-choice options ranging from 1 to 5 or more. On another aspect of dedication, we presented a frequency-based question, "How frequently do you work with code samples?" to gauge dedication to code samples. We offered 5 options: "Every day I work a few hours with code samples"; "I don't work every day, but I work a few hours every week with code samples"; "I work with code samples for a few hours per month, but not every week"; "I work with code samples for a few hours per year"; "I go years without working with code samples".

Still on the topic of dedication, we inquired about activities with the following question, "Which of these activities related to the development, communication, and maintenance of code samples do you perform?". We prompted developers to check or write in the "other" field any activities related to the development, maintenance, and communication of code samples. Among the options available for marking, we inferred activities from repositories and communication, as code sample repositories generate these types of activities. Activities in code sample repositories include: responding to issues, reviewing pull requests, and managing code sample repositories (a generalization of administrative activities such as adding members and collaborators to repositories, releasing new versions, among other repository-related tasks that do not involve responding to issues, reviewing pull requests, and coding). We also provided the option for respondents to indicate if they define requirements or scope and code code samples or coding activities. In terms of communication options, we included: publishing code samples on blogs, forums, and other official channels of organizations, communicating with external developers regarding code samples, training other professionals to handle this communica-

tion within code sample repositories, or being responsible for recruiting new members for the organization through code samples.

Regarding the professionals' perspective, we posed 4 questions. We asked, "In your opinion, what is the main objective of providing code samples" For the main objective, we formulated key goals based on the literature about code samples. The following options were provided to the respondents:

- **Facilitate knowledge sharing:** Code samples can serve as a means for developers to share their knowledge about a framework's feature, a library's functionality, etc., by providing examples of usage and sharing them with other developers. This objective can have educational purposes [32, 33, 39], where other developers learn from these code samples. Knowledge sharing can also be mutual, where both professionals within the organizations and external individuals can contribute and collaborate in the maintenance or creation of new code samples.
- **Promote new functionalities, APIs, libraries, etc.:** Since code samples are provided to exemplify the usage of a specific functionality or feature, they can aim to promote the adoption of new resources, such as a newly designed feature for the Android platform that developers may not be familiar with. Developers can learn about these new resources through code samples and start using them.
- **Accelerate development:** According to some definitions, code samples are meant to help developers become more productive [53]. Developers can also partially or entirely copy code from a code sample for their own use [9]. One utility for code samples is to accelerate development through code reuse, either in whole or in part, or at the very least, to increase the speed of learning about a resource.
- **Promote code reuse:** Developers may use code samples as a way to reuse code. In a recent study [32], it was observed that developers forked repositories, but often did not update the project. This could indicate that some developers may be forking repositories solely for code reuse in their projects and not for contributing. We also found guidelines, such as those from Mozilla, encouraging code reuse, and some "CONTRIBUTING" files in GitHub repositories indicating that code samples should be "developer-friendly" to facilitate copying and pasting by other developers.
- **Promote collaboration and continuous learning:** Sharing code samples on collaboration platforms, such as GitHub, can create opportunities for collaboration between external and internal developers, allowing them to learn from each other, review, and improve the shared code.

This creates a cycle of contributions, where new code samples are made available and evolve over time.

- **To attract new developers to the organization’s ecosystem:** Our previous study [31] showed that organizations receive and review contributions from external developers in their code sample repositories. Through this engagement with code sample repositories, new developers could be attracted to the organization’s ecosystem.

Regarding the main objective, we included the "others" field for participants who disagreed with the provided objectives or wanted to add their own perspectives.

Concerning the target audience of code samples, we asked, “Does the target audience of code samples vary depending on the repository or project?” In the literature, it is mentioned that the audience consuming code samples ranges from developers with little experience to seasoned developers [6]. In the survey, we asked each developer about their perspective on whether the target audience for a code sample changes based on the repository. The options provided were: "Yes, in my organization, code samples are crafted for developers with a specific level of expertise (for instance, some might be made targeting beginner developers while others target experienced developers)," "No, they are designed for developers of any expertise level," "I don't know," "I prefer not to say," and "other."

Regarding the perspective of the similarity of code samples to other products within the organization, we provided two questions: “In your opinion, on a scale of 1 to 5, how similar is the development process of a code sample to that of another product within the organization?” and “In your view, on a scale of 1 to 5, how similar is the quality assurance process of a code sample compared to that of another product within the organization?” We provided respondents with a 5-point Likert scale ranging from 1 (not similar) to 5 (very similar) to assess the similarity between the code samples development process and other products within the organization, as well as the similarity of the quality assurance process for code samples compared to other products. Initially, we had planned to extract more details about the development and quality processes of code samples from participants, along with a more detailed comparison with other products. However, due to time constraints, which could extend the survey and decrease the response rate or even lead to participants potentially raising concerns about non-disclosure agreements, we decided to simplify the question using a Likert scale. Any further details will be explored in subsequent studies through interviews.

Participants in final survey

We received responses from 26 professionals, resulting in a margin of error of approximately 15.85% within a 90% confidence interval. We identified participants from 3 organizations: Google (only 1), Amazon (12), and Microsoft (10). Three participants did not provide their respective emails in the form, so it was not possible to identify which organizations they were from. To anonymize the participants, we included ID from R1 to R26, which we will refer to as such during the results.

The map below, Figure 4.1, displays the participants with colors corresponding to the number of participants from each country. We received responses from North and South America, Europe, and the Middle East. We had 14 participants from the USA, 3 from Brazil, 2 from Canada, 2 from the Netherlands, and 1 participant from each of the following countries: Israel, Germany, France, Finland, and the United Kingdom.

All participants answered all the survey questions, except for the option to provide their email address.

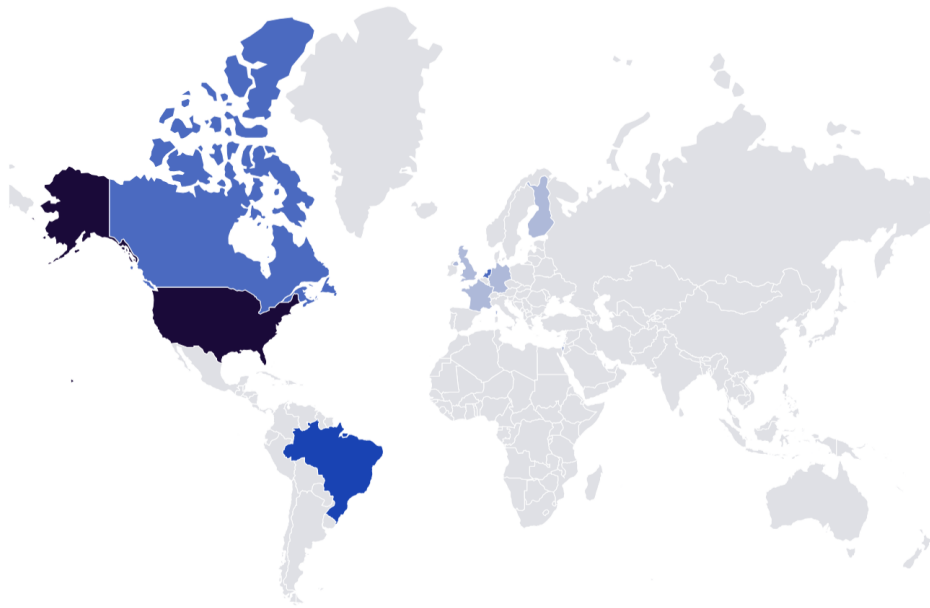


Figure 4.1: Participants location map

4.2.5 Research questions

(RQ1) What is the role of individuals working with code samples within organizations, and what practices are associated with this work? To address this question, we examined the roles reported by the professionals who participated in the survey. Additionally, we considered their experience in terms

of years in software development and with code samples. Lastly, we assessed their reported dedication by analyzing their work frequency, the number of code samples they work on simultaneously, and the activities related to code samples they mentioned.

Rationale: To address this question, we examined the roles reported by the professionals who participated in the survey. Additionally, we considered their experience in terms of years in software development and with code samples. Lastly, by examining the dedication of the professionals, we can understand more practices related to the development and maintenance of code samples within the analyzed organizations.

(RQ2) How does the experience of individuals working with code samples influence the frequency at which they engage in this activity? To answer this question, we considered developers with more than 5 years of experience, who stated that they had worked with code samples in 3 or more organizations, i.e., those who had a number greater than the sum of the median (1) of the professional's number of organizations plus the standard deviation (1.81). On the other hand, less experienced developers were classified as those with less than 5 years of experience and who had worked in fewer organizations previously than the median of the sample, i.e., those who had not worked with code samples in any organization before. We then analyzed the reported frequencies by contrasting them between these more experienced and less experienced professionals.

Rationale: The purpose of this question is to investigate whether individuals with more experience with code samples work with these artifacts more frequently compared to those with less experience. With this inquiry, we aim to understand if there is a relationship between professionals' experience with code samples and the frequency of their dedication to these artifacts.

(RQ3) What is the underlying purpose of providing code samples within organizations, and how are these purposes perceived by the professionals involved? To address this question, we examined two survey questions. The first pertained to the professionals' perspective on the main objective of providing code samples, and the second concerned the target audience for whom the code samples are intended within organizations. Subsequently, we internally analyzed the organizations where the professionals were identified and which had more than one survey participant, in this case, Amazon and Microsoft.

Rationale: Understanding the professionals' perspective regarding the main objective of providing code samples and their target audience will allow us to learn more about what these professionals think and understand their viewpoints. On the other hand, in the internal analysis by organizations, if there

is a misalignment of opinions, it may indicate that within the same organization there are different approaches or strategies for dealing with code samples. This could be due to a lack of a well-structured concept of code samples internally, or it might even be a strategic decision by the organization.

(RQ4) How do the development, quality assurance practices of code samples compare to similar practices applied to other products within organizations? To address this question, we analyzed the two perspective questions obtained through Likert scale feedback from professionals regarding the development process and quality assurance in relation to other products of the organizations. Lastly, we internally analyzed two organizations, specifically Amazon and Microsoft, which had more than one participant, to examine the internal perspective.

Rationale: With this question, we aim to understand how code samples are handled in relation to practices and standards established for the development of other products within the same organization from the professionals' perspective. If they are not classified as similar, it may indicate that code samples might have different strategies concerning their development process and quality within the same organization (misalignment of internal perspective) or between organizations (varying perspectives across organizations).

4.3 Results of Survey

(RQ1) What is the role of individuals working with code samples within organizations, and what practices are associated with this work?

To answer this question, we divided it into subsections, each addressing a specific point.

4.3.1 Roles

In our sample, participants reported being involved in at least 13 different roles while working or having worked with code samples. Figure 4.2 displays the roles declared by the participants. The largest group, comprising 42.31% (11 participants), identified themselves solely as developers. Following that, 30.77% (8 participants) declared themselves as software architects, and 7.69% (2 participants) reported being developer/Technical Advocates. Additionally, we found 1 person in each of the following roles: Content developer, Solutions Architect, CTO, and 2 individuals declared multiple roles.

We can further observe that 2 participants, R21 and R23, reported having played different roles throughout their careers while working with code samples in their respective organizations. Both individuals mentioned working with code samples in 3 or more organizations. Among the roles mentioned

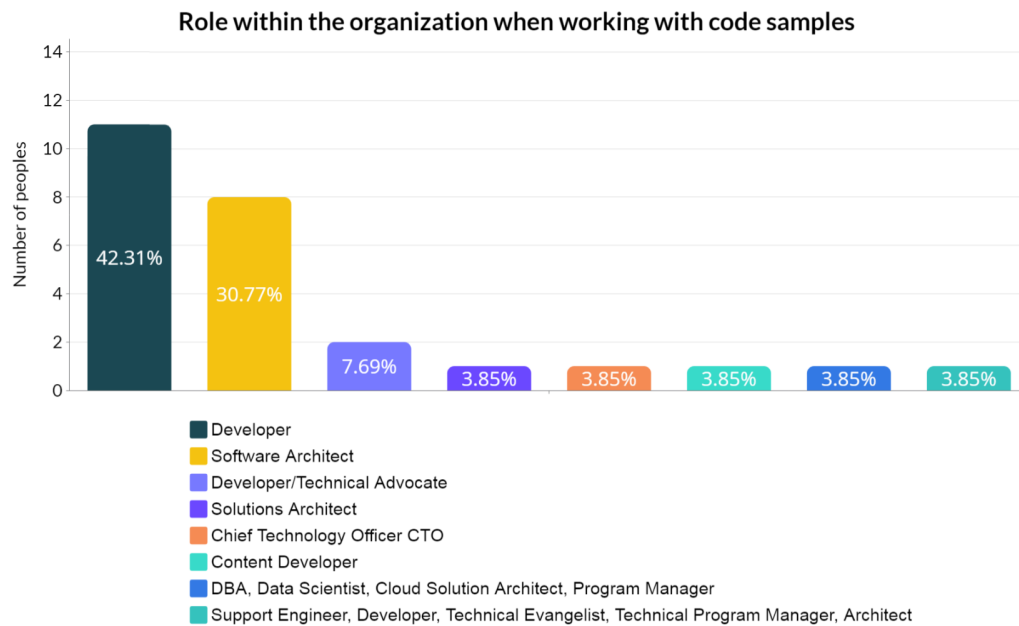


Figure 4.2: Role in organizations

by the participants, we found developers (R2, R3, R6, R13, R17, R18, R25, R26), software architects (R1, R8, R14, R16, R19, R20, R22, R24), developer/Technical Advocate (R9, R15), CTO (R4), Solutions Architect (R11), Content Developer (R12), support engineer (R23), and technical evangelist (R23), program manager (R23).

We believe that the numerous presence of software architects in our sample is related to specific characteristics of the organization in which these professionals are employed. In this case, all of these professionals are from the same organization, Amazon. However, we cannot assert that this concentration of software architects has a direct influence on how code samples are produced, nor that it is a fundamental role in code samples production. Although the presence of software architects in the sample is a relevant data point, it is crucial to explore other organizational aspects to understand how the organization's context may influence the production and usage of code samples.

The results did not demonstrate the existence of a defined professional role for working with code samples. Any software development professional within an organization can be involved in the process of creating, coding, maintaining, and other activities related to code samples. Although a significant portion of the participants declared having only the role of developers, participants from other roles also reported engaging in activities related to code samples similarly to those who identified solely as developers. For example, in our sample, we found that approximately 85% indicated working directly with coding activities of code samples. This specific activity involved individuals from various roles, not being exclusive to a specific role.

We found that working with code samples is not restricted to a specific professional role but is an activity that can involve various roles within a software development organization.

4.3.2 Experience

Regarding software development experience, 76.92% (20 participants) of the sample declared having more than 10 years of experience, and 96.15% (25 participants) have more than 5 years of experience. None of the participants reported having less than 2 years of experience, demonstrating a solid level of experience among the participants. Additionally, none of the participants reported having more years of experience with code samples than with software development.

In Figure 4.3, we compare the number of previous organizations in which each participant worked with code samples with the years of experience working with code samples. We noticed that the majority of participants with more than 5 years of experience with code samples had previously worked with code samples in other organizations.

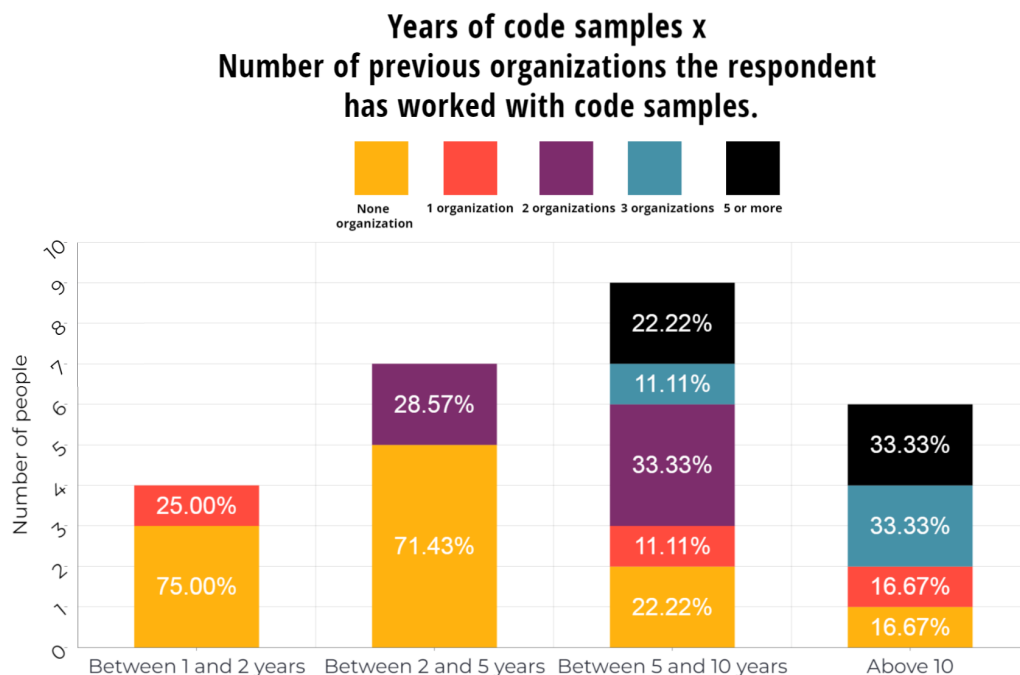


Figure 4.3: Years of code samples vs previous organizations

In general, the results provided an insight into how long the concept of code samples has been present in the industry. Finding professionals who declared more than 10 years of experience with code samples, such as R23, who specified 16 years of work, shows that this concept has been in the industry for over a decade. Moreover, professionals stating that they have worked with

code samples in more than 5 different organizations indicate that there may be many more companies than the ones studied in this research that provide code samples.

Most professionals are very experienced and have already worked with code samples in previous organizations.

4.3.3 Number of code samples

A figure 4.4 displays the values representing the number of code samples participants handle simultaneously. Approximately 50% (13 participants) reported working with one code sample at a time, while 26.92% (7 participants) mentioned dealing with two, 15.38% (4 participants) indicated three, and finally, 7.69% stated they deal with five or more simultaneously.

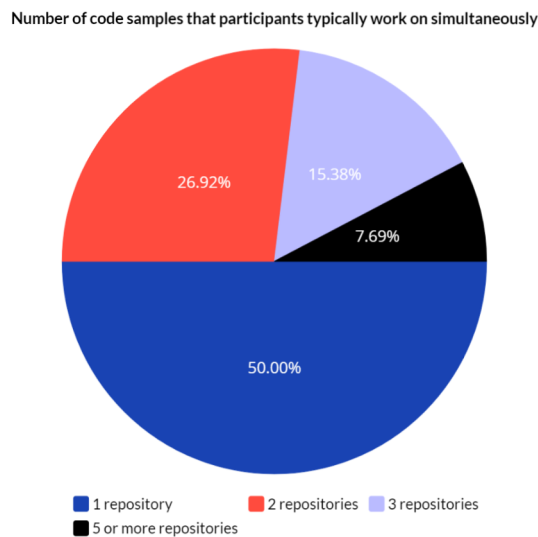


Figure 4.4: Number of code samples worked on simultaneously

The majority of the sample indicated deal 1 or 2 code samples simultaneously (76.92%). This suggests that they allocate time to code samples while also sharing their time with other tasks and responsibilities. As we observed earlier, these professionals hold various roles within the organization and are not solely dedicated to code samples.

Most are not tasked with working with many code samples simultaneously.

4.3.4 Frequency of work with code samples

In Figure 4.5, the distribution of respondents who reported working with code samples at different frequencies is presented. None of the respondents

selected the option "I go years without working with code samples." The majority of respondents (65.38% or 17 participants) do not dedicate themselves to code samples daily but indicated that they work with code samples for a few hours monthly, and some also work with them weekly.

Specifically, 34.62% (9 participants) reported not working with code samples every day but dedicating a few hours per week to them. Another 30.77% (8 participants) stated that they work with code samples for a few hours per month but not weekly. Additionally, 19.23% (5 participants) mentioned working with code samples for a few hours per year. Finally, 15.38% (4 participants) declared working with code samples daily.

Frequency of work with code samples

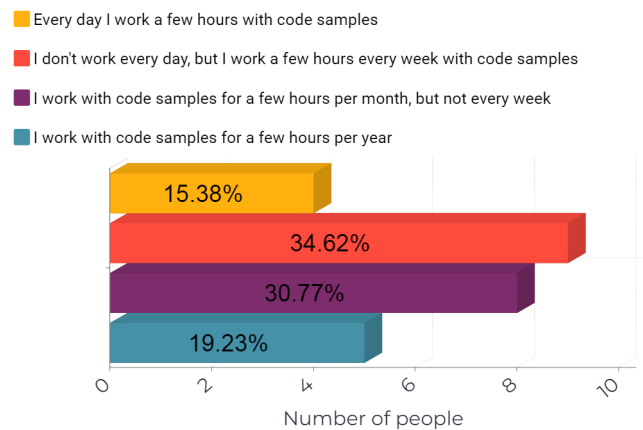


Figure 4.5: Frequency Work with code samples

Among the participants who work more sporadically, contributing a few hours per year, there are some who declared activities such as reviewing pull requests and managing code sample repositories. This raises concerns, as in the previous chapter's study, we observed pull requests taking a long time to be reviewed, contributors giving up on their contributions and closing pull requests without reviews after a while, and a bottleneck in the review process. Professionals who work with code samples sporadically may cause delays in reviews, result in fewer pull requests being reviewed, and discourage community contributions.

The participants in our sample showed to be active in their activities with code samples, with their dedication concentrated monthly. A minority of participants indicated having a more sporadic activity with code samples, mentioning working only a few hours per year. Furthermore, professionals who work sporadically, working only a few hours per year, and yet are responsible for repositories and pull request reviews, may be responsible for potential bottlenecks in the pull request review process.

Most work frequently with code samples, dedicating a few hours each week.

4.3.5 Reported Activities

The figure 4.6 shows the 5 most frequently marked activities, and the results are as follows, as displayed in Figure 4.6: 84.62% (22 participants) reported engaging in coding activities of code samples, 73.08% (19 participants) performed repository management of code samples and their publication through blogs, forums, or other organization platforms. Additionally, 69.23% (18 participants) stated they carry out pull request reviews, and 61.54% (16 participants) mentioned defining the requirements or scope of a code sample. The remaining activities were mentioned by approximately 50% or fewer of the participants.

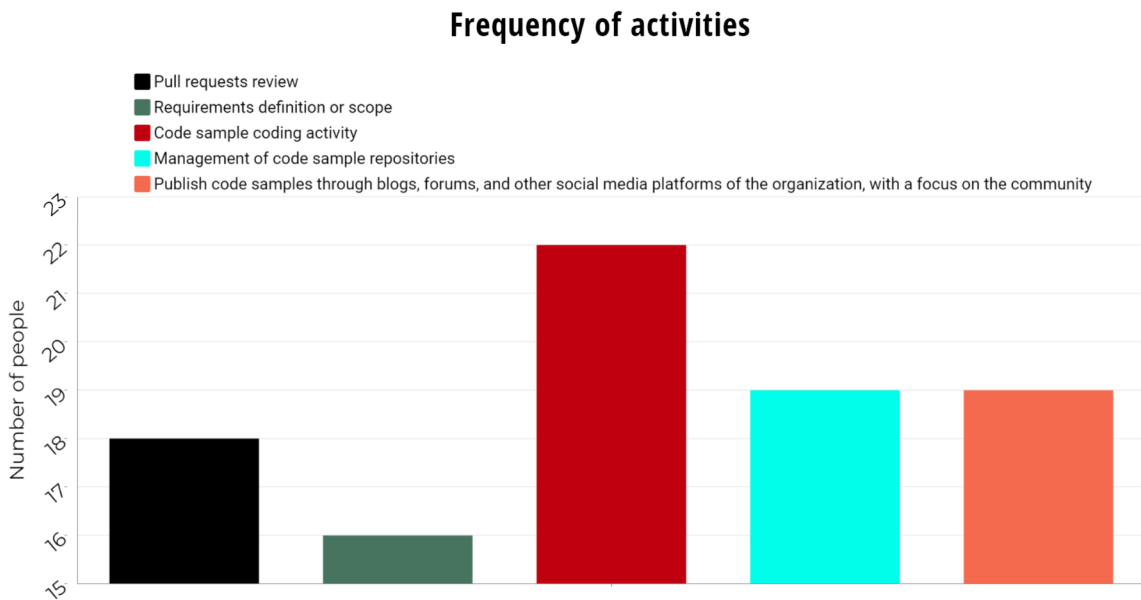


Figure 4.6: Activities in code samples

The results indicate that the majority of participants working with code samples are involved in the coding of the artifacts. Additionally, many of those who code code samples also have the responsibility of managing the repositories (releasing releases, adding members and collaborators, adjusting permissions, among other tasks). They also publish the code samples on official developer websites or blogs of the organization or other platforms, such as the organization's repositories on GitHub. Another frequently mentioned activity was defining requirements or scope, meaning planning and defining the purpose of the code sample to be developed, indicating that a significant portion of the respondents may be responsible for the planning of these code samples, from coding to publication.

Although only the activity of publishing was found among the most reported, we found that 46.15% (12 participants) of the participants engaged in communication activities (include respond to issues) with external developers regarding code samples, such as responding to Issues. Additionally, 19.23% (5 participants) declared being responsible for seeking out skilled developers and members for the organization through code samples, while 23.08% (6 participants) were responsible for training other professionals within the organization to communicate with external developers. The presence of these activities points to the DevRel area, which is significant for code samples, as developers create Issues, pull requests, and attempt to communicate with organizations in code sample repositories.

The occurrence of pull request reviews may indicate a considerable demand in this area, as discovered in the our previous study, which showed that part of the contributions to the repositories occurs through pull requests from external contributors. However, returning to the topic of work frequency with code samples, we identified professionals who reported working only a few hours a year while pointing out important activities in the maintenance of code samples, which could cause a bottleneck in these activities.

The same professionals who work with code sample repositories in organizations are responsible for publishing code samples through other means such as blogs, media, or organization websites. Code samples involve various activities.

In response to RQ1, there isn't a specific role for handling code samples; organizations assign mostly experienced developers to work with code samples. At the same time, they don't overload the tasks, as most reported handling a maximum of 1 or 2 code samples simultaneously. However, the majority work with them frequently. Professionals are typically involved in various activities related to code samples.

(RQ2)How does the experience of individuals working with code samples influence the frequency at which they engage in this activity?

In Figure 4.7, we conducted a comparison between two groups of respondents within our sample: the more experienced and the less experienced. The aim was to investigate whether the experience declared by them influences the frequency at which they work with code samples.

The results revealed that among the more experienced respondents, 85.75% (6 participants) declared dedicating hours of work to code samples weekly, while only 14.29% (1 participant) stated dedicating themselves daily to this type of activity.

On the other hand, among the less experienced respondents, 50% (4 partic-

ipants) declared dedicating a few hours per year to working with code samples, 37.50% (3 participants) dedicate hours monthly, and 12.50% (1 participant) mentioned working with code samples daily.

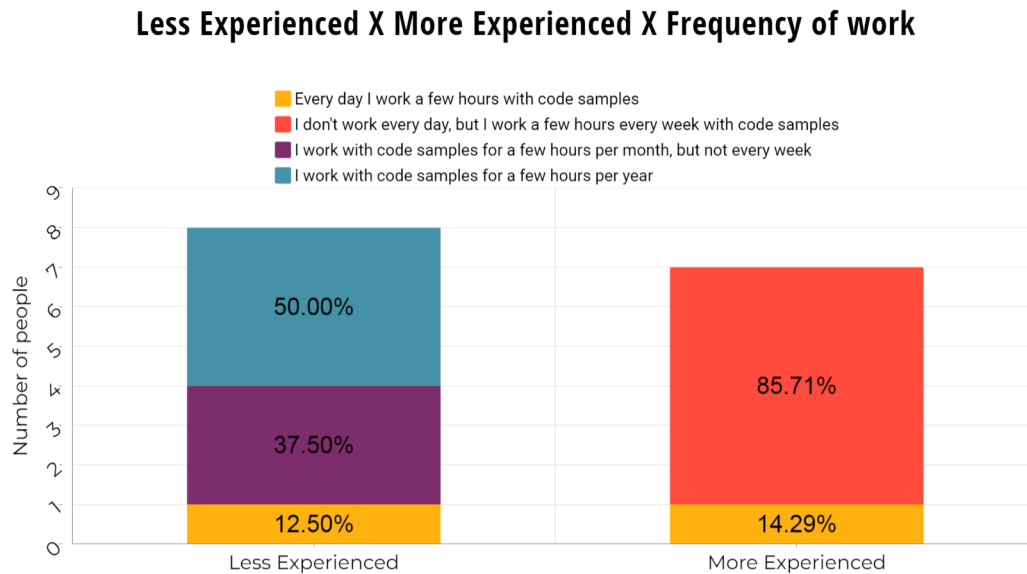


Figure 4.7: Less vs More experienced vs Frequency of work

The results revealed that, in general, more experienced participants work more frequently with code samples. This suggests that organizations may be allocating professionals with greater experience in code samples to work on this artifact, possibly because these individuals are already familiar with code sample development from previous work in other organizations. Another possible reason is that these professionals have a deeper understanding of the resources or technologies they are exemplifying through the code samples. A third motive could be their desire to share their knowledge with less experienced colleagues, prompting them to make code samples available within the organizations.

But less experienced developers may also face more difficulties in producing code samples, as they work less frequently with code sample repositories.

Looking at the roles of the less experienced and more experienced, we found no justification for a greater dedication from the more experienced. Both among the more experienced and the less experienced, there are professionals who state they have other defined roles within the organization.

These results suggest that, in our sample, more experienced participants tend to work with code samples more frequently compared to less experienced ones.

(RQ3) What is the underlying purpose of providing code samples within

organizations, and how are these purposes perceived by the professionals involved? To answer this question, we divided it into subsections, each addressing a specific point.

4.3.6 Main objective of code samples

Regarding the main objective of code samples, we identified four main goals. In Figure 4.8, we observed that 46.15% (12 participants) considered facilitate knowledge sharing as the main objective. Next, 23.08% (6 participants) indicated that the main goal is to accelerate development, while 15.38% (4 participants) stated that the focus is on promoting new functionalities, APIs, libraries, among other resources.

Furthermore, 11.54% (3 participants) highlighted that the main objective of code samples is to attract new developers to the organization's ecosystem. It's worth noting that one of the participant R23 chose to select "others" in the field and considered all available options as main objectives. Additionally, R23 added that, in their view, code samples aim to address questions asked by developers in forums.

On the other hand, the options "Promote code reuse" and "Promote collaboration and continuous learning" were not mentioned directly by any of the participants as the main objective of code samples.

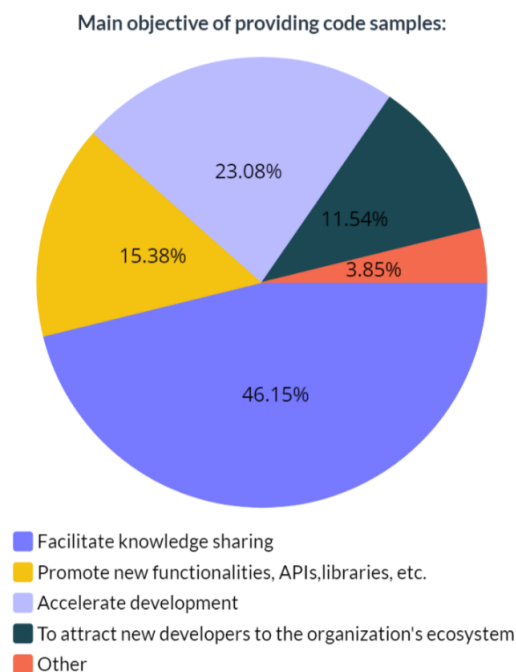


Figure 4.8: Main objective of code samples

The results indicate that although the participants' perspectives may not be aligned, the purposes of code samples can go beyond educational objectives.

They serve to accelerate development, attract developers to the ecosystem, and promote and encourage the use of new features or APIs.

Regardless of the organization, the main objective of providing code samples varied among the participating professionals, but the largest group indicated facilitating knowledge sharing.

4.3.7 Target audience

In Figure 4.9, the depicted outcomes of participant responses illustrate that 73.08% (19 participants) assert variability in target audience across repositories or code sample projects within their organizations. This group affirms that code samples are tailored for developers with specific levels of expertise. Conversely, 26.92% (7 participants) indicated a lack of distinction among created code samples, asserting their universality across expertise levels.

Does the target audience of code samples vary depending on the repository or project?

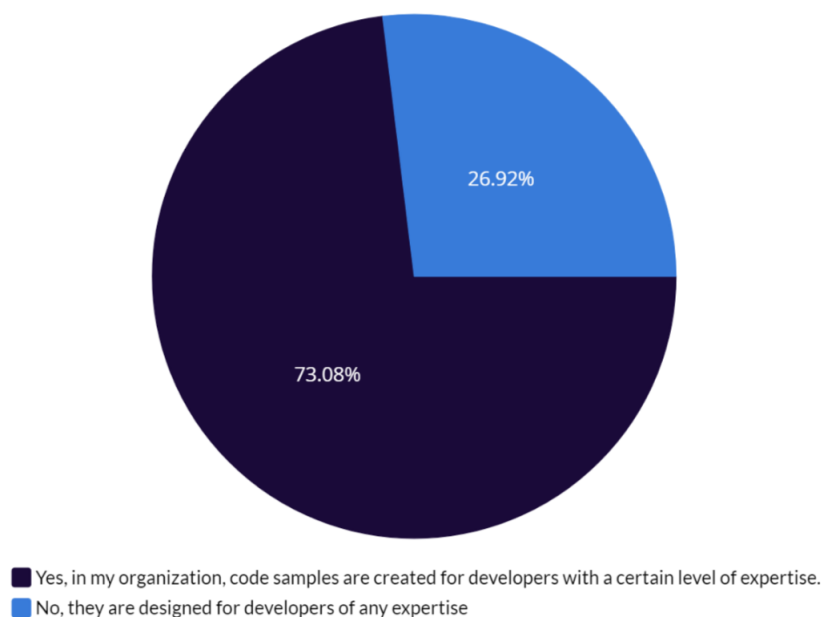


Figure 4.9: Variation of the target audience for code samples in organizations from the participants' perspective.

These findings suggest that most developers working with code samples prioritize levels of knowledge when creating the artifact from their perspective, where some may be crafted for experienced developers, while others are tailored for the less experienced. Meanwhile, a minority show a viewpoint where no distinction is made, indicating that the code samples are made in a more comprehensive manner in terms of target audience.

Regardless of the organization, in the view of the majority, the level of expertise is prioritized when producing a code sample.

4.3.8 Internal perspective within each organization

In Figure 4.10, it can be observed that for Amazon, 33.33% (4 participants) indicated 'Facilitate knowledge sharing' as their main objective. Furthermore, 25% (3 participants) selected 'Accelerate development', another 25% opted for 'Promote new functionalities, APIs, libraries, etc.', and finally, 16.67% (2 participants) indicated 'To attract new developers to the organization's ecosystem'.

In the context of Microsoft, we observed that 60% (6 participants) highlighted 'Facilitate knowledge sharing' as their main objective. Meanwhile, 20% (2 participants) mentioned 'Accelerate development', 10% (1 participant) pointed out 'To attract new developers to the organization's ecosystem', and equally 10% (1 participant) mentioned 'Other', agreeing with all the objectives.

Organization X Main purpose of providing code samples

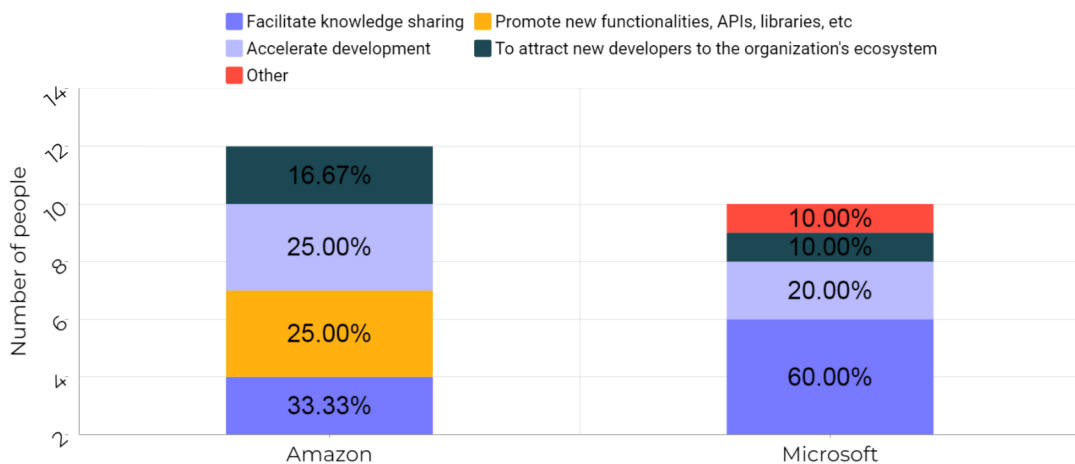


Figure 4.10: Main purpose of providing code samples vs Organizations

These results demonstrate that, although the main objective was not fully aligned between the two organizations, there was a greater degree of convergence within Microsoft, with 60% of participants emphasizing the goal of facilitating knowledge sharing.

This highlights a greater variation in objectives within Amazon compared to Microsoft, where participants exhibited greater consistency toward a collaborative approach, providing code samples to share knowledge. On the other hand, in Amazon, certain perspectives indicated a focus on promoting new functionalities, an approach that did not found among Microsoft participants.

In both organizations, a minority also viewed the code samples as an opportunity to attract new developers to the company's ecosystem. However, the most prevalent objective in both situations was to facilitate knowledge sharing.

In Figure 4.11, we conducted a comparison between employees from the same organizations, specifically Amazon and Microsoft, in order to understand their perspectives regarding code samples. The results showed that there are different viewpoints among employees from the two companies.

At Amazon, 25% (3 participants) of the employees declared that, in their view, code samples are projects intended for developers regardless of their expertise, meaning there is no distinction based on the developers' skill levels. On the other hand, 75% (9 participants) reported that, in their organization, there are indeed distinctions between code samples, with some designed for a certain level of expertise and others for another level.

At Microsoft, 30% (3 participants) stated that code samples are designed for developers of any expertise level, indicating there is no distinction based on developers' skills. However, 70% (7 participants) affirmed that, in their organization, there are indeed distinctions between code samples, and some are created specifically for a certain expertise level.

Organization X View on the variation of the target audience of a code samples

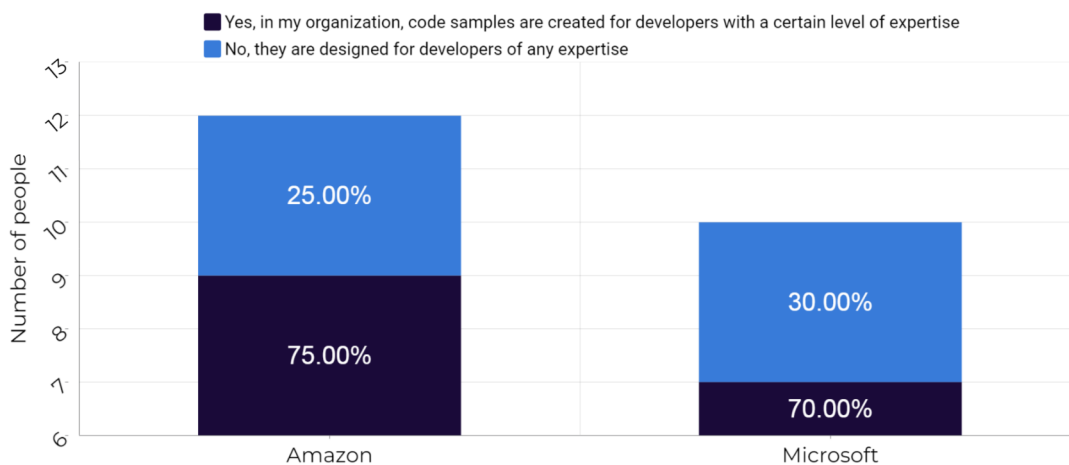


Figure 4.11: Target audience vs Organizations

In an individual analysis of the respondents, we observed that there were people who had been working with code samples for many years on both sides of the responses. Additionally, there were individuals who declared involvement in both coding and defining requirements or scope of code samples in both companies, yet they presented different perspectives regarding code sam-

ples, either at Microsoft or Amazon.

These results suggest that perspectives on code samples can vary within the same organization, even among employees who perform similar roles with code samples. The existence of different viewpoints may be related to factors such as the specific objectives of the code samples these participants worked on and their individual experiences, or it could be simply a matter of personal interpretation.

As a result, we identified various objectives for providing code samples from the participants' perspectives. However, we noticed a misalignment in their views. While some saw a variation in the expertise level of developers targeted by the code samples, others within the same organization did not share the same perspective. A similar discrepancy was observed regarding the main objective of providing code sample.

(RQ4) How do the development, quality assurance practices of code samples compare to similar practices applied to other products within organizations? To answer this question, we divided it into subsections, each addressing a specific point.

4.3.9 Overview of the similarity between development process and quality assurance with other products of the organization

Regarding the development process, the participants voted on a Likert scale from 1 to 5, where the value 1 indicated "not similar" and the value 5 "very similar." Option 3, representing neutrality in similarity, was the most voted by 34.62% (9 participants). However, we noticed that 19.23% (5 participants) voted for option 1, and another 19.23% (5 participants) voted for option 2, forming a majority that considered the process as "not similar" or "not very similar." On the other hand, 15.38% (4 participants) and 11.54% (3 participants) indicated that the process was "similar" or "very similar."

The same scale was applied regarding the quality assurance process. Again, the neutral option was the most voted, totaling 38.46% (10 participants). In this case, 34.62% (9 participants) voted for option 2, and 7.69% (2 participants) voted for option 1 or "not similar." As for similarity, there were fewer votes compared to the previous case, with 11.54% (3 participants) rating it as "similar," and 7.69% (2 participants) voting for "very similar."

In both cases, the majority of votes were for options indicating neutrality or low similarity. The number of participants voting for "not similar" (option 1) or "not very similar" (option 2) was higher than those who chose the neutral

option. If we disregard the neutral option and group those who voted 1 or 2 as negative options regarding similarity, we have in both cases a majority group of 42.31% voting negatively about the similarity of the quality assurance process compared to other products within the organization, and 38.46% in relation to the similarity of the development process. The smallest portion of our sample rated the similarity of code samples positively compared to other products within the organization.

In general, regardless of the organization, the majority of participating professionals indicated from a neutral vote downwards, meaning the largest group rated it as "not very similar" or "not similar", both for the development process and for the quality assurance process compared to other products of the organization.

4.3.10 *Internal perspective within each organization*

Now we are starting an internal comparison of the responses of professionals from the same organization, remembering that in the case of Google there was only 1 participant, and 3 were not identified, so they will be excluded, leaving only Amazon and Microsoft. In Figure 4.12, the Likert scale votes of the participants from Amazon and Microsoft are displayed regarding the similarity of the code sample development process compared to other products within the organization.

At Amazon, 4 participants remained neutral regarding the similarity, while 6 voted negatively, rating between 1 and 2 on the scale, indicating they perceive low similarity. On the other hand, 2 participants voted positively, giving a rating between 4 and 5, suggesting similarity with other products within the organization.

At Microsoft, 4 participants also positioned themselves as neutral, 4 voted negatively with a rating between 1 and 2, indicating low similarity, and 2 voted positively with a rating between 4 and 5, pointing out similarity with other products within the organization.

Regarding the similarity of the quality assurance process, in Figure 4.13, we observed that among the participants from Amazon, 4 remained neutral in the rating scale, 6 voted negatively about the quality (between 1 and 2), and 2 participants rated positively (between 4 and 5). In Microsoft, 5 remained neutral in the rating scale, 3 voted negatively (between 1 and 2), and 2 voted positively (4)

In a more detailed analysis of the participants who rated the process of development and quality assurance, we observed the following results: Among the 4 participants who voted positively (R6, R7, R8, and R12) regarding the

**Working Amazon and Microsoft with code samples
X Similarity of the code sample development
process with other products of the organization**

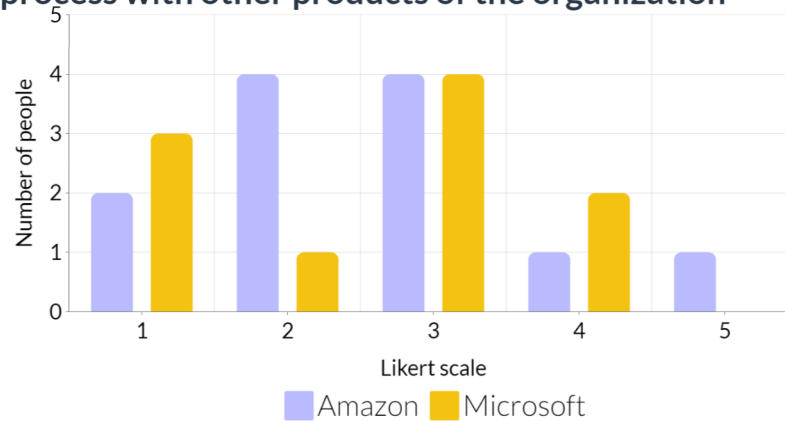


Figure 4.12: Process of development similarity vs Organizations

similarity of the development process, 2 of them (R7 and R12, one from each organization) voted neutral regarding the quality assurance process. On the other hand, among those who voted positively for quality assurance (R6, R8, R10, R16), 2 of them (R10 and R16, one from each organization) voted neutral regarding the development process. Thus, out of the 4 who voted positively for the similarity of the development process and the 4 who voted positively for the similarity of the quality assurance process, only 2 participants maintained their positive votes in both cases.

Among those who rated negatively the similarity of the development process, none of them rated the quality positively, and only 3 participants (1 from Microsoft and 2 from Amazon) changed their votes to neutral regarding the quality. As for the 9 participants who rated negatively the similarity of the quality assurance process, none of them changed their vote to positive regarding the development process, and 2 of them (from Amazon) changed their votes to neutral regarding the development process.

Both at Microsoft and Amazon, there were more experienced participants with code samples and a similar frequency of work on both sides of the rating.

The analysis shows that most participants were consistent in their positive and negative ratings in both questions about similarities. In the manual analysis, we did not identify individuals who oscillated from negative to positive in both organizations. Some participants may have been unsure about the similarity of one aspect (development process or quality assurance), leading them to change their votes to neutral in one of the questions.

Working Amazon and Microsoft with code samples X Similarity of the code sample assurance quality with other products of the organization

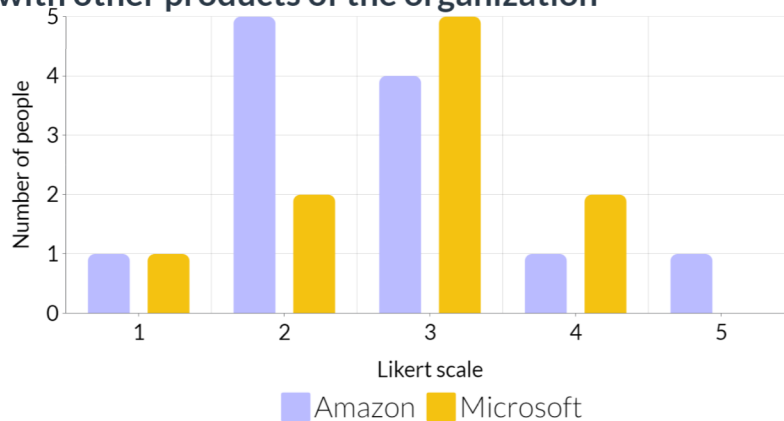


Figure 4.13: Process of assurance quality vs Organizations

The results also indicate that there are internal differences in perspectives within both organizations, and factors such as experience with code samples may not have influenced the decision. However, in both organizations, the results show that the majority, excluding neutral votes, rated negatively in both questions.

4.4 Implications

4.4.1 Code Samples: Beyond Educational Objectives

The main objectives indicated by participants when providing code samples suggest that these artifacts go beyond the educational purposes often mentioned in literature. Although the most frequently mentioned goal was 'Facilitating Knowledge Sharing' through code samples, which can be beneficial for both those wishing to learn from them and those providing them, this objective often aligns with educational purposes. However, some participants pointed out that code samples also serve other purposes in their view.

Code samples are also used to expedite the development process and not just for learning, especially when a code sample is well-structured and of good quality, as it can be incorporated into a project. Another goal identified was to promote new functionalities by providing code samples, thus introducing developers to these new features. Code samples can also be strategically used to attract external developers to collaborate and participate in the organization's ecosystem.

4.4.2 *Veteran Developer, Old code samples*

In our study, The majority of participants had over 10 years of experience in the software industry, and most of them had been working with code samples for more than 5 years, having gained experience across various previous organizations. Furthermore, we observed that more experienced developers tend to engage more frequently with code sample repositories.

Experienced professionals indicated through their years of working with code samples that it is not a new concept in the industry. They highlighted that the concept of code samples has existed for more than 10 years. We had participants who reported producing code samples for as long as 16 years. In our previous chapter's research, we had only observed repositories dating back to 2013.

The presence of professionals experienced in the area of code samples may suggest an organizational strategy to allocate these individuals. Given organizations' interest in providing code samples to exemplify the use of certain APIs, features, or libraries, it makes sense to assign experienced individuals to share knowledge, especially benefiting less experienced developers. Moreover, developers experienced, especially those proficient in producing code samples, might be more familiar with contribution guidelines and the code sample production process within the organization. This can result in higher quality code samples that are better aligned with the organization's objectives.

However, it's important to note that code samples, as a means of sharing knowledge and exemplifying the use of a resource, can also be created by less experienced developers. Thus, more experienced developers may be encouraging and guiding their less experienced peers in relation to this artifact.

4.4.3 *Internal Disparities in Code Sample Perspectives*

We have observed that, even within a single organization, there are differing perspectives regarding the objectives, target audience, and development and quality assurance processes of code samples compared to other organizational products. It's essential to note that we are looking at organizations with teams and branches spread across various regions and countries, which might contribute to these perspective shifts.

Organizations might adopt distinct strategies for creating and maintaining code samples. For instance, one organization might vary the target audience of the code samples, crafting some for more experienced developers and others for beginners. However, different objectives might lead to variations in development and quality assurance processes. Therefore, it's important for the organization to determine whether this variation is a strategic feature or a

concern.

On the other hand, the misalignment in perspectives among members of the same organization suggest that the guidelines and objectives for the code samples might not be consistently understood or applied. If this is the case, there will be a need to enhance communication and training among professionals, ensuring a unified understanding of the purpose of the code samples and the processes involved in their development and maintenance.

4.5 Threats to validity

In this section, we discuss the study limitations based on four categories of threats to validity described by Wohlin *et al.* [59]. For each category, we have a set of possible threats to the validity of an experiment. Here, we identified these possible threats to our study and allocated them into the most fitting category and discussed measures we took to reduce the risk of each threat.

Internal Validity: One of the threats in this case is related to historical changes. To mitigate it, we analyzed the candidates' organizations one week before sending out the final survey. The manual analysis aimed to ensure that these employees belonged to the organizations, by verifying their corporate email, GitHub, and LinkedIn profiles. However, employees could still have outdated profiles or may no longer be part of the organizations.

External Validity: One of the threats to external validity could be the generalizability of the study. To mitigate it, we sought to obtain employees from four multinational organizations, which may have employees spread across various countries, aiming to achieve maximum diversity in the sample. We received responses from individuals in nine different countries. Another threat is related to the sample size, with a margin of error of approximately 18.68% at a 95% confidence interval.

Construct Validity: One of the threats in this regard is related to the formulation of questions, as some questions may allow for multiple interpretations. To mitigate this, the questions were analyzed by three other researchers involved in the project. Still, some respondents may have refrained from providing information truthfully due to non-disclosure agreements (N.D.A). To mitigate this threat, we tried to make the questionnaire as non-intrusive as possible, not forcing respondents to answer some questions and providing the option to disagree or withhold information in others.

Conclusion Validity: The sample size may limit the p-value and statistical significance in some cases. Another threat is related to the term "working with code samples," which can have many interpretations, such as merely using or making use of them in projects (which was not the survey's target

audience). To mitigate this threat, we specifically sought participants involved in some modification activity (e.g., releasing, merging, responding to issues or pull requests) in code sample repositories in their respective organizations.

4.6 *Final Considerations*

In this chapter, we conducted a survey with 26 professionals, representing 3 distinct organizations identified. Through this survey, we explored various aspects, including the roles played by participants, their years of experience in software development and code samples, as well as the number of previous organizations they worked with on code sample projects. Furthermore, we investigated the dedication of these professionals, addressing activities, work frequency, and the number of code samples they work on simultaneously.

Consequently, we delved into participants' perspectives on the target audience for which the code samples are intended, their main objectives, and their views on the similarities and differences between the development and quality assurance processes compared to other products within the organization. The results obtained revealed that the majority of participants have more than 5 years of experience in both software development and code samples, and have worked in more than 1 previous organization within this context.

We observed that professionals' dedication to code samples varies, with most of them investing hours on a weekly or monthly basis in this activity. Regarding the number of code samples worked on concurrently, we found that the majority of participants handle 1 or 2 samples.

Furthermore, we identified a diversity of roles undertaken by professionals involved with code samples. Perceptions regarding the objectives of code samples and their target audience exhibited variations among participants. The prevailing opinion indicated that the development and quality assurance processes for code samples have little to no similarity to those of other products within organizations.

Implications

5.1 *Implications*

5.1.1 *Fostering Collaborative Synergy: Code Samples as Bridges Between Organizations and External Developers*

Code samples are not produced just to be consumed; organizations expect a synergy between internal developers and external developers to help in the development and maintenance of the artifact. In Chapter 3, we analyzed approximately 12,000 pull requests from 2,179 code samples of 4 different organizations. We found that most of the pull requests from external developers were reviewed and accepted. Furthermore, among the professionals from the organizations explored in Chapter 4, 69.23% (18 participants) of them were responsible for reviewing pull requests within the organizations, found also that 46.15% (12 participants) were responsible for responding to issues and communicating with external developers. In addition, among these professionals, we found people who were dedicated daily to the code samples, thus showing an engagement of organizations with the community. Code samples can be a means for developers in general to participate in large ecosystems like Android's and use code samples not just for consumption but also as a way to share knowledge, suggest or send bug fixes, address other issues, and even contribute with new code samples.

5.1.2 *Some practices for providing code samples*

For organizations interested in working with code samples, we identified practices that demonstrate that these projects go beyond sharing in repositories. They are dynamic, evolving as highlighted in the literature, and actively receive contributions from external developers, as noted in Chapter 3. Following the insights from Chapter 4, among the principal practices are: having professionals who code, manage code samples repositories, and publish these samples, whether through the organization's websites or blogs, and who are committed to reviewing pull requests. It's important to have team members ready to communicate with external developers. These activities can align with the practice of 'Developer Relations', which can be understood as an organizational area that works to create and nurture a community, maintaining a collaborative relationship with developers and the proprietary platform [13]. Although we identified only three professionals area who classified themselves in this category, in chapter 4, these individuals are responsible for providing APIs, code samples, and other resources to facilitate community engagement, primarily through communication activities [38]. Another practice observed in participants is that, even if not all dedicate daily to code samples, the vast majority put in some hours, whether weekly or monthly. Furthermore, professionals focus on only a few code samples at a time, which may imply that they also have other roles and tasks beyond working with code samples.

Related work

Code samples in SECO. Code samples within the context of software ecosystems have been addressed in previous studies [6, 32, 33]. Menezes *et al.* assessed the characteristics of the source code, its evolution, and the usage of code samples [32]. Menezes *et al.* observed the primary challenges and the main needs of code sample clients [33]. Braga *et al.* analyzed the target audience for code samples and gauged the experience level of their clients [6]. Although we adhere to the definitions of code samples found in previous articles as being complete software projects that are typically simple and small to facilitate understanding and evolve over time, unlike other studies, our research delves into code samples beyond their structural characteristics and the clients who utilize them. We explore how organizations are handling contributions from external developers in repositories, which can aid in maintaining and evolving the code samples, as well as the vision and dedication of the developers in the studied organizations towards code samples.

SECO and artifacts. Within the SECO context, we found studies that explored other artifacts which may be related to code samples, given that code samples serve to exemplify the use of APIs, frameworks, and libraries [32]. Bogart *et al.* investigate APIs within the software ecosystem context and the impact of community contributions on changes to these APIs [3], while Bavota *et al.* explore the evolution of library dependencies in Apache projects and how updates are crucial for these projects to keep functioning [1]. The code sample artifact is directly related to these other artifacts, like libraries and APIs, since they need to evolve as an API or a library does to maintain their relevance [32]. In some SECO, external developers become crucial for maintenance, and it's in this scenario of code sample evolution through external contributions that

we delve into in this work. Other studies have explored APIs in the ecosystem context and the challenges developers might encounter when trying to learn to use APIs [46, 55]. Even though no paper directly mentions the code sample artifact, it can be one of the means to assist in understanding APIs.

Pull requests and contributions. Regarding repositories, contributions, and pull requests, we found some studies that analyzed contributions through this medium and some implications in a context of projects that aren't specifically code samples. Soares *et al.* explored around 97,000 pull requests from 30 GitHub projects to identify characteristics that have impacted pull requests' lifetimes [36]. Soares *et al.* conducted a study with 22,523 pull requests to understand the factors that influence the assignment of reviewers to pull requests [52]. Gousios *et al.* assessed approximately 166,000 pull requests to discern the characteristics of lifetime, acceptance, and rejection of pull requests in 291 projects [17]. Cosentino *et al.* conducted a systematic mapping with four research questions and evaluated 80 publications from 2009 to 2016. Among the topics they addressed, they identified the main reasons for pull requests being accepted or rejected [10]. Other studies have explored contribution rules in repositories through the "CONTRIBUTING" file. For instance, Kobayakawa *et al.* analyzed "CONTRIBUTING" files from 459 GitHub projects to identify the significance of this file [22]. Elazhary *et al.* utilized the content of "CONTRIBUTING" files from 53 projects to check if the recommendations and guidelines were being followed in practice [12]. As is the case with other projects, repositories of code samples stored on GitHub can also receive contributions via pull requests and have rules for contributions. In our work, we sought to evaluate if organizations were receiving, reviewing, and specifying rules for contributions concerning this artifact.

Survey with developers. One of the strategies for collecting opinions, perspectives, and information from developers who have worked on GitHub projects is to invite them by email to respond to surveys. Several studies have chosen this approach to get opinions from Github users [4, 21, 37, 42, 50]. We found several works that applied surveys to GitHub users in other contexts. In Pham *et al.*'s study, they invited 4,000 GitHub users by email to validate findings on 'how the increased transparency on GitHub influences developers' testing behaviors,' receiving responses from 569 users [42]. Shahin *et al.* empirically investigated how Development (Dev) and Operations (Ops) teams are organized in the software industry for the adoption of Continuous Delivery and Deployment (CD) practices. For data collection, they invited approximately 4,000 GitHub users, resulting in a response rate of less than 10% [50]. Borges *et al.* conducted a survey with developers who worked on GitHub repositories, inviting 4,370 participants and receiving 791 responses to unveil their motiva-

tions for starring projects on GitHub repositories [4]. Jiang *et al.* received 102 responses from GitHub developers regarding participants in repositories who do not contribute to those repositories [21]. Oliveira *et al.* conducted a survey with 110 developers from GitHub to evaluate the applicability of two models for computing programming skills based on the metrics of Changed Files and Changed Lines of Code [37]. Similarly to other studies, we sought GitHub users via email, but in the context of code samples repositories, contacting users involved in the development or maintenance of this artifact.

Conclusions and Future Work

In this study, we explored code examples within the context of ecosystems, which are stored in GitHub repositories. We began an exploratory investigation into code sample repositories across four platforms: Android, AWS, Azure, and Spring. In the initial phase of the study, we found that organizations value the external developer community that contributes code samples and engage to review and accept such contributions. However, we also identified areas of concern, such as delays in review and in closing pull requests, especially those that are rejected. Additionally, we observed that only a small group of maintainers was responsible for the majority of reviews, which could be an organizational strategy or an area needing improvement if necessary.

During the second part of the study, we conducted a survey with developers from the organizations that own the platforms discussed in the first part, involving participation from employees of 3 organizations. Our goal was to focus on developers within these organizations, comprehending their experiences, roles, and dedication to code sample production. Moreover, we sought to grasp these developers' perspectives on the purpose of code samples, their intended audience, and their perceptions of the development and quality assurance processes, in comparison to other products within the organizations.

We identified the presence of developers with over 10 years of experience, both in software development and in creating code samples; these professionals had engaged with code samples across various organizations. The number of years spent working with code samples suggested by the participants indicates that the concept of code samples in the industry spans more than a decade. We observed that code sample developers typically allocate a few hours per week or month to this activity, with more experienced ones often

investing more time. We noted that the objectives of code samples extend beyond their educational aspect, promoting new features, expediting the development process, and even fostering engagement between developers and organizational ecosystems. These code samples also serve as a means for external developers to contribute and showcase their expertise within the community and organizations.

We encountered divergent perspectives among developers regarding the target audience for code samples, as well as differing opinions about the development and quality assurance processes compared to other organizational products. This may indicate an internal misalignment in understanding the development and maintenance process of code samples in relation to the guidelines and objectives set within organizations. This could be a potential area for improvement if these varying perspectives impact the quality or purpose of the code samples.

As this work concentrated on exploratory studies, we suggest as future work the conduct of interviews with developers from organizations to obtain deeper insights into how development and quality assurance processes function. Additionally, more research is needed to comprehend how individuals responsible for code samples are selected within organizations, and how their activities are distributed, aiming to consolidate best practices and establish standards for producing higher quality and more precise code samples for clients.

Bibliography

- [1] Bavota, G., Canfora, G., Di Penta, M., Oliveto, R., e Panichella, S. (2015). How the apache community upgrades dependencies: an evolutionary study. Empirical Software Engineering, 20:1275–1317. Citado na página 56.
- [2] Bettenburg, N., Hassan, A. E., Adams, B., e German, D. M. (2015). Management of community contributions: A case study on the android and linux software ecosystems. Empirical Software Engineering, 20:252–289. Citado na página 2.
- [3] Bogart, C., Kästner, C., Herbsleb, J., e Thung, F. (2016). How to break an api: cost negotiation and community values in three software ecosystems. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, páginas 109–120. Citado na página 56.
- [4] Borges, H. e Valente, M. T. (2018). What’s in a github star? understanding repository starring practices in a social coding platform. Journal of Systems and Software, 146:112–129. Citado nas páginas 57 e 58.
- [5] Bosch, J. (2009). From software product lines to software ecosystems. In SPLC, volume 9, páginas 111–119. Citado na página 4.
- [6] Braga, W. M., Menezes, G., Fontao, A., Hora, A., e Cafeo, B. (2020). Quero lhe usar! uma análise do público alvo de code samples. In Anais do VIII Workshop de Visualização, Evolução e Manutenção de Software, páginas 33–40. Citado nas páginas 1, 7, 26, 32, e 56.
- [7] Bushra, F., Usman, A., e Naveed, A. (2011). Effect of transformational leadership on employees’ job satisfaction and organizational commitment in banking sector of lahore (pakistan). International journal of Business and Social science, 2(18):261–267. Citado na página 29.

- [8] Calikli, G., Bener, A., e Arslan, B. (2010). An analysis of the effects of company culture, education and experience on confirmation bias levels of software developers and testers. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2, páginas 187–190. Citado na página 29.
- [9] Corporation, M. (2023). Code example guidelines. <https://shre.ink/MozillaOrg>. Online; accessed 17 July 2023. Citado nas páginas 7 e 31.
- [10] Cosentino, V., Izquierdo, J. L. C., e Cabot, J. (2017). A systematic mapping study of software development with github. Ieee access, 5:7173–7192. Citado na página 57.
- [11] Dias-Neto, A. C., Matalonga, S., Solari, M., Robiolo, G., e Travassos, G. H. (2017). Toward the characterization of software testing practices in south america: looking at brazil and uruguay. Software Quality Journal, 25:1145–1183. Citado na página 29.
- [12] Elazhary, O., Storey, M.-A., Ernst, N., e Zaidman, A. (2019). Do as i do, not as i say: Do contribution guidelines match the github contribution process? In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), páginas 286–290. IEEE. Citado na página 57.
- [13] Fontão, A., Cleger-Tamayo, S., Wiese, I., Pereira dos Santos, R., e Claudio Dias-Neto, A. (2023). A developer relations (devrel) model to govern developers in software ecosystems. Journal of Software: Evolution and Process, 35(5):e2389. Citado na página 55.
- [14] Fontão, A., Cleger-Tamayo, S., Wiese, I., Santos, R. P. d., e Dias-Neto, A. C. (2020). On value creation in developer relations (devrel) a practitioners' perspective. In Proceedings of the 15th international conference on global software engineering, páginas 33–42. Citado na página 30.
- [15] Gini, C. (1912). Variabilita e mutabilita. Reprinted in Memorie di metodologica statistica (Ed. Pizetti E). Citado na página 17.
- [16] GitHub, I. (2023). About pull requests. Online; accessed 20 July 2023. Citado na página 14.
- [17] Gousios, G., Pinzger, M., e Deursen, A. v. (2014). An exploratory study of the pull-based software development model. In Proceedings of the 36th international conference on software engineering, páginas 345–355. Citado nas páginas 21, 23, e 57.

- [18] Hellendoorn, V. J., Devanbu, P. T., e Bacchelli, A. (2015). Will they like this? evaluating code contributions with language models. In 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, páginas 157–167. IEEE. Citado nas páginas 21 e 23.
- [19] Huang, Y. e Chung, W. (2019). Rockstar effect in distributed project management on github social networks. In Proceedings of the 2019 Pre-ICIS SIGDSA Symposium. Citado na página 28.
- [20] Jansen, S., Cusumano, M. A., e Brinkkemper, S. (2013). Software ecosystems: analyzing and managing business networks in the software industry. Edward Elgar Publishing. Citado na página 4.
- [21] Jiang, J., Lo, D., Ma, X., Feng, F., e Zhang, L. (2017). Understanding inactive yet available assignees in github. Information and Software Technology, 91:44–55. Citado nas páginas 57 e 58.
- [22] Kobayakawa, N. e Yoshida, K. (2017). How github contributing. md contributes to contributors. In 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), volume 1, páginas 694–696. IEEE. Citado na página 57.
- [23] Lee, M. J., Ferwerda, B., Choi, J., Hahn, J., Moon, J. Y., e Kim, J. (2013). Github developers use rockstars to overcome overflow of news. In CHI '13 Extended Abstracts on Human Factors in Computing Systems, CHI EA '13, página 133–138, New York, NY, USA. Association for Computing Machinery. Citado na página 28.
- [24] Lethbridge, T., Singer, J., e Forward, A. (2003). How software engineers use documentation: the state of the practice. IEEE Software, 20(6):35–39. Citado na página 6.
- [25] Lima, T., dos Santos, R. P., Oliveira, J., e Werner, C. (2016). The importance of socio-technical resources for software ecosystems management. Journal of Innovation in Digital Ecosystems, 3(2):98–113. Citado nas páginas 1 e 8.
- [26] Lorenz, M. O. (1905). Methods of measuring the concentration of wealth. Publications of the American statistical association, 9(70):209–219. Citado na página 17.
- [27] Manikas, K. (2016). Revisiting software ecosystems research: A longitudinal literature study. Journal of Systems and Software, 117:84–103. Citado nas páginas 1, 6, 14, e 22.

- [28] Manikas, K. e Hansen, K. M. (2013). Software ecosystems—a systematic literature review. Journal of Systems and Software, 86(5):1294–1306. Citado nas páginas 4 e 14.
- [29] Marlow, J., Dabbish, L., e Herbsleb, J. (2013). Impression formation in online peer production: activity traces and personal profiles in github. In Proceedings of the 2013 conference on Computer supported cooperative work, páginas 117–128. Citado na página 21.
- [30] McIntosh, S., Kamei, Y., Adams, B., e Hassan, A. E. (2014). The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects. In Proceedings of the 11th working conference on mining software repositories, páginas 192–201. Citado na página 14.
- [31] Melo, M., Menezes, G., e Cafeo, B. (2022). Exploring pull requests in code samples. In Anais do X Workshop de Visualização, Evolução e Manutenção de Software, páginas 36–40. SBC. Citado nas páginas 2 e 32.
- [32] Menezes, G., Cafeo, B., e Hora, A. (2019). Framework code samples: How are they maintained and used by developers? In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). Citado nas páginas 1, 6, 7, 15, 31, e 56.
- [33] Menezes, G., Cafeo, B., e Hora, A. (2022). How are framework code samples maintained and used by developers? the case of android and spring boot. Journal of Systems and Software, 185:111146. Citado nas páginas 1, 6, 14, 15, 31, e 56.
- [34] Microsoft (2023a). Azure samples. GitHub repository. Citado na página 1.
- [35] Microsoft (2023b). microsoft samples. Online; accessed 17 July 2023. Citado na página 7.
- [36] Moreira Soares, D., de Lima Júnior, M. L., Murta, L., e Plastino, A. (2021). What factors influence the lifetime of pull requests? Software: Practice and Experience, 51(6):1173–1193. Citado nas páginas 13, 17, 19, e 57.
- [37] Oliveira, J., Souza, M., Flauzino, M., Durelli, R., e Figueiredo, E. (2022). Can source code analysis indicate programming skills? a survey with developers. In International Conference on the Quality of Information and Communications Technology, páginas 156–171. Springer. Citado nas páginas 57 e 58.

- [38] Oliveira, R., Ajala, C., Viana, D., Cafeo, B., e Fontão, A. (2021). Developer relations (devrel) roles: An exploratory study on practitioners' opinions. In Proceedings of the XXXV Brazilian Symposium on Software Engineering, páginas 363–367. Citado nas páginas 30 e 55.
- [39] Oracle (2023). Oracle. Online; accessed 17 July 2023. Citado nas páginas 7 e 31.
- [40] Padhye, R., Mani, S., e Sinha, V. S. (2014). A study of external community contribution to open-source projects on github. In Proceedings of the 11th working conference on mining software repositories, páginas 332–335. Citado na página 2.
- [41] Palinkas, L. A., Horwitz, S. M., Green, C. A., Wisdom, J. P., Duan, N., e Hoagwood, K. (2015). Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. Administration and policy in mental health and mental health services research, 42:533–544. Citado na página 28.
- [42] Pham, R., Singer, L., Liskin, O., Figueira Filho, F., e Schneider, K. (2013). Creating a shared understanding of testing culture on a social coding site. In 2013 35th International Conference on Software Engineering (ICSE), páginas 112–121. IEEE. Citado na página 57.
- [43] Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G., e Oliveto, R. (2021). Toxic code snippets on stack overflow. IEEE Transactions on Software Engineering, 47(3):560–581. Citado na página 6.
- [44] Rahman, M. M. e Roy, C. K. (2014). An insight into the pull requests of github. In Proceedings of the 11th working conference on mining software repositories, páginas 364–367. Citado na página 13.
- [45] Reid, D., Jahanshahi, M., e Mockus, A. (2022). The extent of orphan vulnerabilities from code reuse in open source software. In Proceedings of the 44th International Conference on Software Engineering, ICSE '22, página 2104–2115, New York, NY, USA. Association for Computing Machinery. Citado na página 6.
- [46] Robillard, M. P. (2009). What makes apis hard to learn? answers from developers. IEEE software, 26(6):27–34. Citado na página 57.
- [47] Sarker, I. H. e Apu, K. (2014). Mvc architecture driven design and implementation of java framework for developing desktop application. International Journal of Hybrid Information Technology, 7(5):317–322. Citado na página 6.

- [48] Seichter, D., Dhungana, D., Pleuss, A., e Hauptmann, B. (2010). Knowledge management in software ecosystems: Software artefacts as first-class citizens. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, ECSA '10, pagina 119–126, New York, NY, USA. Association for Computing Machinery. Citado na página 6.
- [49] Services, A. W. (2023). Aws samples. GitHub repository. Citado na página 1.
- [50] Shahin, M., Zahedi, M., Babar, M. A., e Zhu, L. (2017). Adopting continuous delivery and deployment: Impacts on team structures, collaboration and responsibilities. In Proceedings of the 21st international conference on evaluation and assessment in software engineering, páginas 384–393. Citado nas páginas 28, 29, e 57.
- [51] SINOVA, U. (2023). About contributions. Online; accessed 29 July 2023. Citado na página 29.
- [52] Soares, D. M., de Lima Júnior, M. L., Plastino, A., e Murta, L. (2018). What factors influence the reviewer assignment to pull requests? Information and Software Technology, 98:32–43. Citado nas páginas 13, 14, 17, e 57.
- [53] Spring (2023). Spring | guides. Online; accessed 17 July 2023. Citado nas páginas 7 e 31.
- [54] Stylos, J. e Myers, B. A. (2006). Mica: A web-search tool for finding api components and examples. In Visual Languages and Human-Centric Computing (VL/HCC'06), páginas 195–202. IEEE. Citado na página 6.
- [55] Tian, Y., Thung, F., Sharma, A., e Lo, D. (2017). Apibot: question answering bot for api documentation. In 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), páginas 153–158. IEEE. Citado nas páginas 6 e 57.
- [56] Treude, C., Figueira Filho, F., e Kulesza, U. (2015). Summarizing and measuring development activity. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, páginas 625–636. Citado na página 29.
- [57] Van Den Berk, I., Jansen, S., e Luinenburg, L. (2010). Software ecosystems: a software ecosystem strategy assessment model. In Proceedings of the fourth european conference on software architecture: Companion volume, páginas 127–134. Citado nas páginas 4 e 6.

- [58] van Ingen, K., van Ommen, J., e Jansen, S. (2011). Improving activity in communities of practice through software release management. In Proceedings of the International Conference on Management of Emergent Digital EcoSystems, páginas 94–98. Citado nas páginas 4 e 15.
- [59] Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., e Wessln, A. (2012). Experimentation in Software Engineering. Springer Publishing Company, Incorporated. Citado nas páginas 24 e 52.
- [60] Wouters, J., Ritmeester, J., Carlsen, A., Jansen, S., e Wnuk, K. (2019). A seco meta-model: A common vocabulary of the seco research domain. In Software Business: 10th International Conference, ICSOB 2019, Jyväskylä, Finland, November 18–20, 2019, Proceedings 10, páginas 31–45. Springer. Citado na página 4.
- [61] Wu, Y., Kropczynski, J., Shih, P. C., e Carroll, J. M. (2014). Exploring the ecosystem of software developers on github and other platforms. In Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work amp; Social Computing, CSCW Companion '14, pagina 265–268, New York, NY, USA. Association for Computing Machinery. Citado na página 9.
- [62] Wyrich, M., Graziotin, D., e Wagner, S. (2019). A theory on individual characteristics of successful coding challenge solvers. PeerJ Computer Science, 5:e173. Citado na página 29.
- [63] Xavier, J., Macedo, A., e de Almeida Maia, M. (2014). Understanding the popularity of reporters and assignees in the github. In SEKE, páginas 484–489. Citado na página 28.
- [64] Yang, D., Hussain, A., e Lopes, C. V. (2016). From query to usable code: an analysis of stack overflow code snippets. In Proceedings of the 13th International Conference on Mining Software Repositories, páginas 391–402. Citado na página 6.
- [65] Yu, Y., Wang, H., Filkov, V., Devanbu, P., e Vasilescu, B. (2015). Wait for it: Determinants of pull request evaluation latency on github. In 2015 IEEE/ACM 12th working conference on mining software repositories, páginas 367–371. IEEE. Citado na página 20.
- [66] Zampetti, F., Ponzanelli, L., Bavota, G., Mocci, A., Di Penta, M., e Lanza, M. (2017). How developers document pull requests with external references. In 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC), páginas 23–33. IEEE. Citado na página 13.

- [67] Zhang, X., Chen, Y., Gu, Y., Zou, W., Xie, X., Jia, X., e Xuan, J. (2018). How do multiple pull requests change the same code: A study of competing pull requests in github. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), páginas 228–239. IEEE. Citado na página 13.
- [68] Zhu, Z., Hua, C., Zou, Y., Xie, B., e Zhao, J. (2017). Automatically generating task-oriented api learning guide. In Proceedings of the 9th Asia-Pacific Symposium on Internetware, páginas 1–10. Citado na página 6.