OPEN-SOURCE AND CLOSED-SOURCE PROJECTS: A FAIR COMPARISON

Cristian AVATAVULUI[1]

Andrei-Iulian CUCU[2]

Alexandru-Mihai GHERGHESCU[3]

Costin-Anton BOIANGIU[4]

Iulia-Cristina STANICA[5]

Cătălin TUDOSE[6]

Mihai-Lucian VONCILĂ[7]

Daniel ROSNER[8]

Abstract

The objective of this study is to provide a comprehensive comparison between open-source projects and closed-source projects by examining several key aspects, including the number of contributors or employees, the number of features introduced per project, the number of vulnerabilities present in the software, revenue or profit, and project management techniques. By comparing these aspects across a range of open-source and closed-source projects, we aim to evaluate the potential for these distribution models to complement each other and identify the contexts in which one model may be more effective than the other. This analysis seeks to provide valuable insights for stakeholders involved in software development, including developers, project managers, and decision-makers. In conclusion, this study provides a nuanced comparison of open-source and closed-source software projects, addressing key areas like contributor numbers, feature development, vulnerabilities, revenue, and project management. The insights gained are intended to guide

[1] PhD student, Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, cristian.avatavului@stud.acs.pub.ro
[2] PhD student, Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, andrei_iulian.cucu@stud.acs.upb.ro
[3] PhD student, Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, agherghescu2411@stud.acs.upb.ro
[4] Professor, PhD Eng., Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, costin.boiangiu@cs.pub.ro
[5] Lecturer, Faculty of Engineering in Foreign Languages, University Politehnica of Bucharest, Romania, iulia.stanica@upb.ro
[6] Lecturer, Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, florin.tudose2110@upb.ro
[7] PhD student, Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, mihai_lucian.voncila@stud.acs.upb.ro
[8] Lecturer, Faculty of Automatic Control and Computers, University "Politehnica" of Bucharest, Romania, daniel.rosner@upb.ro

stakeholders in software development, helping to discern which model is more effective in various contexts and how they might complement each other.

Keywords: open-source, closed-source, comparison, software projects

JEL Classification: L86, C88

1. Introduction

1.1 Objective

The primary objective of this study is to conduct a comprehensive comparison between open-source software projects and closed-source software projects. The comparison is based on various key aspects, including the number of contributors/employees, the number of features introduced per project, the number of vulnerabilities (bugs or more serious problems) present in the software, revenue, and project management techniques.

One important metric to consider is the number of contributors, which is relevant due to the fundamental differences in the underlying development models of the two types of projects. Open-source projects rely on contributions from a large and diverse pool of contributors, who may participate on a part-time basis or as paid employees of other organizations. In contrast, closed-source projects typically rely on a fixed number of employees working full-time, with specific tasks assigned to them.

The second metric examined is the number of features introduced in the software project. Open-source projects rely on the voluntary contributions of developers, which can lead to a more decentralized and organic approach to feature development. Conversely, closed-source projects often assign teams to specific features, leading to a more streamlined and efficient process.

The third metric is the number of vulnerabilities present in the software. Open-source projects often have many contributors, which can facilitate the rapid detection and resolution of security issues. In contrast, closed-source projects may prioritize feature development over security, resulting in a higher likelihood of vulnerabilities.

The fourth metric is revenue or profit, which differs significantly between the two models. Closed-source projects prioritize profit, often using aggressive marketing strategies, while open-source projects generally develop software as "freeware", focusing on attracting users and contributors to the project, which can lead to support from large tech companies. However, smaller open-source projects may rely on donations or voluntary contributions from developers.

Finally, the study examines project management techniques employed by both models. Closed-source projects typically follow a strict hierarchical structure, with limited opportunities for movement or change. In contrast, open-source projects typically operate with little or no formal hierarchy, relying on a decentralized and self-organizing structure that allows developers to work on tasks that interest them.

Overall, this study provides valuable insights into the similarities and differences between open-source and closed-source software projects, which can inform decision-making by developers, project managers, and other stakeholders involved in software development.

1.2 Importance

Historically, the computer market was dominated by big tech companies that held monopolies over different aspects of the supply chain. This created significant challenges for smaller firms seeking to compete, as they lacked the knowledge and resources necessary to do so effectively. Open-source projects have disrupted this status quo by providing access to the same tools and knowledge to all participants, regardless of their size. This has resulted in increased competition and innovation in the industry, benefiting both large and small firms alike.

Today, open-source and closed-source projects coexist in many computer systems and are widely used by companies and individuals around the world. A notable example is the Android operating system, which utilizes a Linux kernel, an open-source project, in combination with proprietary software packages and applications.

Given the ubiquity of both open-source and closed-source software, it is important to conduct a comparative analysis of their respective development methods and associated advantages and disadvantages. Such an analysis can help inform decision-making by software developers and consumers alike and shed light on the potential implications of using one approach over the other.

1.3 Terminology

Throughout this paper, the terms open-source and closed-source will be used frequently. By open-source, we refer to projects that have a publicly available codebase on a hosting site, accessible and usable by anyone. By closed-source, we refer to projects that do not have a publicly available codebase but only provide the application (end-product) to the users. For this paper, we do not consider other aspects such as licensing, which could further categorize these two types of projects. Our focus is on software development projects of any industry or niche.

GitHub[9], a popular code-hosting platform, is used by many projects to share their code and related comments/issues using the git tool as the underlying versioning system.

[9] www.github.com

In this paper, we define a bug as any fault or failure in the code that results in unexpected behavior but does not affect the security or vulnerability of the system. On the other hand, a vulnerability is any fault or failure in the code that results in unexpected behavior and leads to a significant security vulnerability, as classified in the official CVE dataset [1].

A feature refers to any improvement made to an existing codebase, bringing new changes as noticed by the end-user or improving the existing codebase's performance or other noticeable benefits as dictated by common knowledge.

The term end-user can refer to any person or company using a software product, regardless of their technical background. In the case of open-source projects, end-users may also be direct code contributors to the project.

A code contributor is any person who adds or modifies the code in a meaningful way, fixing bugs or developing new features.

In conclusion, this study meticulously examines the multifaceted dynamics between open-source and closed-source software projects, delving into the nuances of contributor engagement, feature development, software vulnerabilities, financial models, and management methodologies. Through this analytical lens, we aim to elucidate the distinct characteristics and operational paradigms of each model, thereby enriching the understanding of their respective efficacies and potential synergies in various contexts. The insights garnered from this comparative analysis are intended to serve as a robust foundation for informed decision-making and strategic planning among software developers, project managers, and other key stakeholders in the realm of software development. This endeavor not only contributes to the academic discourse but also pragmatically guides the evolving practices in the software industry.

The paper's structure is as follows: Chapter 2 compares the number of contributors to different projects, Chapter 3 compares the features of different projects, Chapter 4 compares the number of vulnerabilities in certain projects, Chapter 5 compares profit margins and funding/revenue, and Chapter 6 discusses project management techniques and their influence on project development. Chapter 7 provides the paper's conclusion.

Key Takeaways:

- Objective of Study: To comprehensively compare open-source and closed-source software projects based on contributors, features, vulnerabilities, revenue, and project management techniques.
- Contributors: Open-source projects typically have a large, diverse pool of part-time or externally employed contributors. Closed-source projects rely on a fixed number of full-time employees.
- Feature Development: Open-source projects feature decentralized, organic development, while closed-source projects have a more streamlined, team-specific approach.

- Vulnerabilities: Open-source projects may have quicker detection and resolution of security issues due to many contributors. Closed-source projects might prioritize features over security, potentially increasing vulnerabilities.
- Revenue Models: Closed-source projects focus on profit, often with aggressive marketing. Open-source projects are usually freeware, attracting users and contributors, and may receive support from large tech companies or rely on donations.
- Project Management: Closed-source projects often have a strict hierarchical structure, whereas open-source projects usually have a decentralized, self-organizing structure with more freedom for developers.
- Market Impact: Open-source projects have disrupted traditional tech monopolies, increasing competition and innovation.
- Coexistence of Models: Both open-source and closed-source software are prevalent, exemplified by the Android OS, which uses both.
- Importance of Comparative Analysis: Analyzing these models aids decision-making for developers and consumers, highlighting the implications of each approach.
- Terminology Definitions:
    - Open-Source: Projects with publicly accessible codebases.
    - Closed-Source: Projects without public codebases, offering only the end-product.
    - Bugs and Vulnerabilities: Defined in relation to unexpected behavior and security impact.
    - Features: Improvements or enhancements in the codebase.
    - End-Users and Code Contributors: Defined in the context of software usage and development.

2. Contributors/employees

A crucial aspect in comparing open-source and closed-source projects is the number of contributors or employees involved in the development of a particular project. Although this metric does not directly indicate the pace of development or the number of features a project has, it is still significant and can potentially affect both aspects. A higher number of contributors can facilitate finding support for new features and enable a more efficient resolution of bugs in the software application.

In this paper, we examine data encompassing both open-source and closed-source projects. For instance, Figure 2 displays the most extensive open-source projects on Github ranked by the number of contributors as of 2022, based on a report published by the platform [3]. Interestingly, despite these projects being open-source, the majority of them are owned by major tech companies that utilize these tools in a business setting. Consequently, many contributors to these projects are likely employees of the tech giants who own the projects, remunerated to work on specific codebases.

To support this assertion, Table 1 and Figures 2 [10] exhibit contributions from companies renowned for closed-source projects rather than open-source ones. Table 1 illustrates the top ten private companies based on the number of active contributors as of January 2023, according to the Open-Source Contributor Index [2], a platform that extracts information about open-source repositories hosted on GitHub. These numbers reflect the contributors directly involved in the codebase for open-source projects.

Rank	**Organization**	**Active Contributors**	**Total Community**
1	Google	7188 +*219*	13819 +*295*
2	Microsoft	6956 +*190*	14417 +*212*
3	Red Hat	4195 +*53*	6085 +*33*
4	Intel	3009 +*96*	5910 +*121*
5	Amazon	2793 +*90*	6571 +*121*
6	IBM	2719 +*72*	6337 +*101*
7	Facebook	1845 +*30*	7134 +*196*
8	GitHub	1845 +*63*	3990 +*37*
9	VMware	1244 +*33*	2271 +*49*
10	SAP	1132 +*50*	2282 +*42*

Table 1. Commercial organizations' contributor numbers as of January 2023, according to the Open-Source Contributor Index (an increase from the previous month).

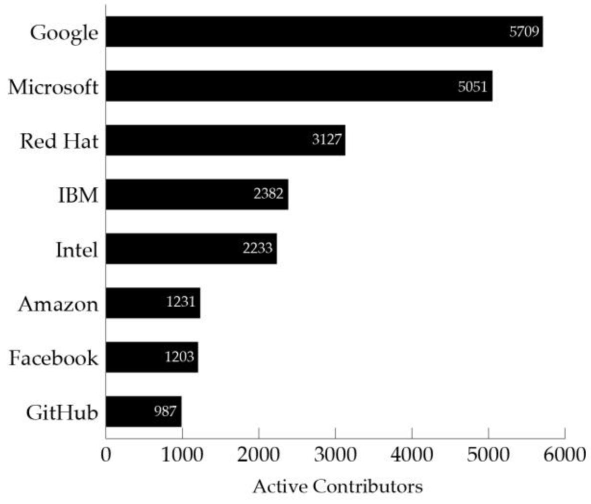

Figure 1. Open-source contributions reported by Github on their platform in December 2020.

Figure 1 illustrates a comparable metric provided by GitHub, dated December 2020, which presents code active contributions on their platform. In this case, active contributors are the users that have more than ten commits. A noteworthy observation is a year-on-year variation. Notably, Google has contributed more than 1,200 new contributors to open-source projects each year [10].

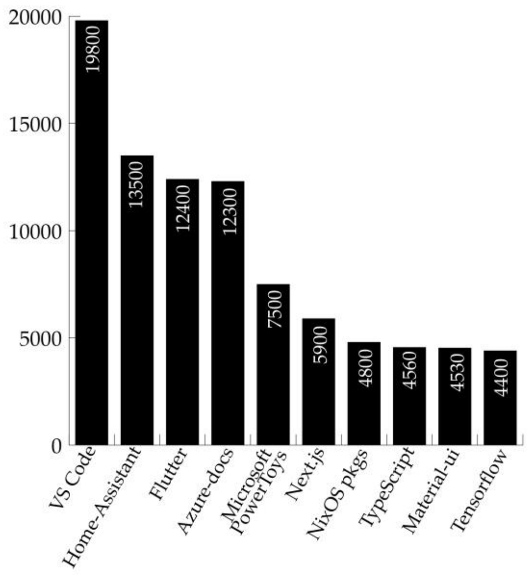

Figure 2. The biggest open-source projects on GitHub, according to the total number of contributors in 2022.

In the context of closed-source projects, the amount of available information is limited. Nonetheless, we were able to gather data regarding the number of employees from four major tech companies, namely [4][5][6][7]. It should be noted, however, that this figure is not indicative of the precise number of software developers, but rather the total number of employees. For instance, some sources estimate that Google has approximately 18,000 employees worldwide that are involved in software creation, development, or related tasks, which is comparable to the number of contributors to major open-source projects. Conversely, Microsoft is reported to have approximately 40,000 employees who work on software development.

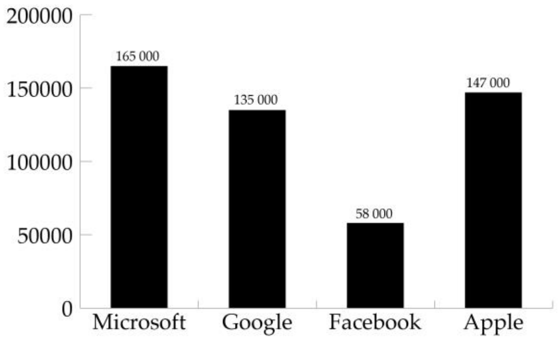

Figure 3. Speculative total number of employees for 4 of the biggest tech companies (all types of employees, not just developers) for 2020.

In conclusion, this chapter's detailed exploration of the number of contributors and employees in both open-source and closed-source projects reveals multifaceted dynamics within the software development landscape. Our analysis, underpinned by data from various reputable sources, indicates a significant presence of major tech companies in the open-source arena, often paralleling their closed-source endeavors. Notably, the juxtaposition of contributor engagement in open-source projects with the employee base in closed-source ventures offers a nuanced understanding of how human resources are allocated and utilized in these distinct yet interconnected domains.

Key Takeaways:

- Contributor/Employee Numbers in Open-Source vs. Closed-Source Projects: The number of contributors or employees is significant in comparing open-source and closed-source projects, though it doesn't directly indicate the development pace or feature count.
- Open-Source Projects Dominated by Major Tech Companies: Most extensive open-source projects are owned by major tech companies, with many contributors being their employees.

- Data from GitHub and Open-Source Contributor Index: Data shows the top open-source projects on GitHub and the top private companies with the most active contributors in open-source projects.
- Tech Giants Leading in Open-Source Contributions: Companies like Google, Microsoft, and Red Hat are leading in terms of active contributors to open-source projects.
- Year-On-Year Variation in Contributions: There is a notable yearly increase in contributors to open-source projects, with Google adding over 1,200 new contributors annually.
- Limited Data on Closed-Source Project Employees: Information about the number of employees in closed-source projects is limited and not necessarily indicative of the number of software developers.
- Comparison of Employees in Major Tech Companies: Estimates show significant numbers of employees in major tech companies involved in software-related tasks, comparable to the number of contributors to major open-source projects.

3. Features comparison

This metric assumes significance in evaluating a software project's success, which directly influences its acceptance among users or investors. The number of features and the development time are critical indicators that determine the speed and efficiency with which the software evolves and is distributed to the end users. In the case of open-source projects, we obtained data from 10 applications, sourced from the TAWOS Dataset [8], which we analyzed and processed as per our research requirements. The outcomes of our analysis are presented in Figure 6, which includes crucial metrics such as the number of bugs or issues, the count of fixed bugs, and the number of features of the respective projects. Additionally, Figure 4 illustrates the number of launched features, in-development features, and the aggregate number of features across all 44 Microsoft applications listed on their official roadmap website [9].

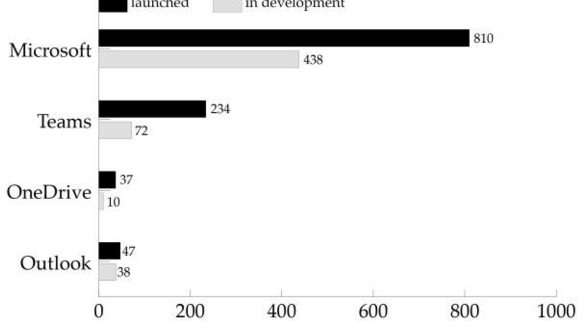

Figure 4. Features numbers and status for Microsoft products, according to the company's official roadmap, between 2021-2023.

Initially, it is crucial to examine Microsoft's closed-source products. Assuming that Microsoft has approximately 40,000 software engineering positions and considering the 1,200 features developed between 2021 and 2023, we can deduce that approximately 33 engineers worked on each feature during this period. It should be noted that not all features were developed simultaneously, and certain features took longer to develop than others. Nonetheless, this approximation offers us a general idea of the engineers' engagement in Microsoft's closed-source projects.

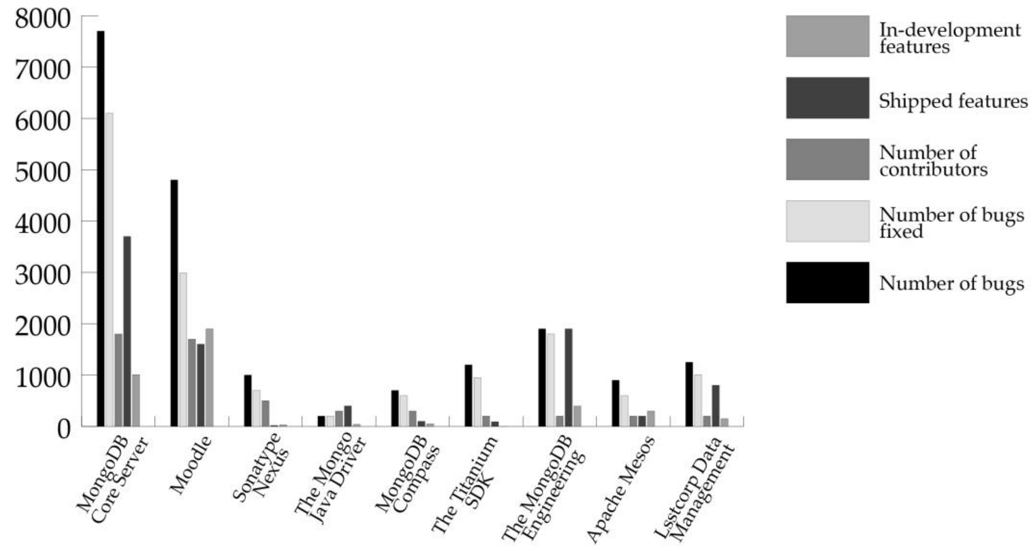

Figure 5. Statistics for 10 open-source projects, sorted by the number of contributors (2018-2020).

Examining several open-source projects presented in Figure 5 (derived from [8]), we have calculated the ratio of contributors to the total number of features for the first three products during a comparable two-year period (2018-2020): MongoDB Core Server - 0.37, Moodle - 0.51, Sonatype Nexus - 0.92. These figures are noticeably different from those of Microsoft's closed-source products. We posit that these differences arise from distinct working methods. On the one hand, Microsoft maintains a clear hierarchy, with numerous personnel working within sub-teams. Conversely, open-source projects lack such a clear hierarchical structure, allowing contributors to move freely between features, codebase issues, or even other projects altogether. Ultimately, these differences reflect each project's objectives. Closed-source projects concentrate on specific features demanded by their clients (either large corporations or a significant number of global users), while open-source projects tend to be more diffuse in their feature objectives, where a particular feature may be introduced by a contributor for personal use or the benefit of a small number of community members, prioritizing practicality over the service of a large number of end-users. This distinction can be summed up by the term impact (referring to the number of end-users reached): closed-source projects concentrate on fewer features with greater impact, while open-source projects spread out over more features with less impact.

In conclusion, our comparative analysis of features in both open-source and closed-source software projects, as delineated in Figures 4 and 5, underscores the fundamental disparities in development methodologies and objectives between these two paradigms. The data-driven approach, utilizing the TAWOS Dataset and Microsoft's official roadmap, has illuminated the intricate dynamics of feature development and distribution. In particular, the significantly divergent ratios of contributors to features in open-source projects, as compared to the more concentrated and impact-focused approach in Microsoft's closed-source ventures, highlight the distinct operational ethos that characterizes each model. Such insights not only deepen our understanding of software development practices but also offer valuable perspective for stakeholders in making strategic decisions that align with their project's specific goals and intended user impact.

Key Takeaways:

- Significance of Features Comparison: This metric is crucial for assessing a software project's success and its acceptance among users or investors.
- Key Indicators: The number of features and development time are essential indicators for determining the software's evolution speed and efficiency.
- Data Source for Open-Source Projects: Analysis based on data from 10 applications from the TAWOS Dataset, focusing on metrics like bug counts and feature numbers.
- Microsoft's Closed-Source Products: Analysis of around 1,200 features developed between 2021 and 2023, with an estimated 33 engineers working on each feature.
- Variation in Development Patterns: Not all features in Microsoft's projects were developed simultaneously, and some took longer than others.
- Open-Source Projects Analysis: Comparison of the ratio of contributors to features in projects like MongoDB Core Server, Moodle, and Sonatype Nexus, showing significant differences from Microsoft's pattern.
- Differences in Organizational Structure: Microsoft has a clear hierarchy and sub-teams, while open-source projects lack such structure, allowing more flexibility for contributors.
- Project Objectives and Impact: Closed-source projects focus on specific features with greater impact for clients, while open-source projects have more diffuse objectives, often prioritizing practicality for smaller user groups.
- Feature Focus: Closed-source projects concentrate on fewer, high-impact features, whereas open-source projects spread across more features with less overall impact.

4. Vulnerabilities

Security is a crucial aspect of software development, and as such, it is essential to consider it when writing and deploying code online. Therefore, our third metric focuses on the security of both closed-source and open-source projects. Figure 7 showcases ten projects, five of which are open-source and five closed-source, that experienced a high number of security issues, according to the official Common Vulnerabilities and Exposures (CVE) dataset [1], during a period spanning from January 2018 to September 2019. The severity of vulnerabilities is measured by the CVSS score, which takes into account multiple factors such as exploitability, impact, complexity, scope, and other relevant metrics. The CVSS score categorizes issues into high, medium, or low severity.

Figure 6 displays the number of vulnerabilities found in popular open-source and closed-source projects between January 2018 and September 2019. The Common Vulnerabilities and Exposures (CVE) dataset was used to identify and categorize the severity of vulnerabilities. The CVE score considers factors such as exploitability, impact, complexity, and scope to rank vulnerabilities as high, medium, or low severity. Android had the highest number of high-severity vulnerabilities with 403 issues, while Debian had the highest number of medium-severity security vulnerabilities at 658. Firefox had the lowest number of vulnerabilities in all three categories among the five open-source projects.

Regarding closed-source projects, all except Acrobat Reader DC had a lower number of low or medium-severity vulnerabilities compared to high-severity breaches. Windows Server 2019 and Edge had 238 and 135 high-severity issues, respectively, whereas Mac OS X performed the best with only 120 vulnerabilities in all three categories. Overall, closed-source projects had almost half as many vulnerabilities as open-source projects, with 1558 and 2731, respectively.

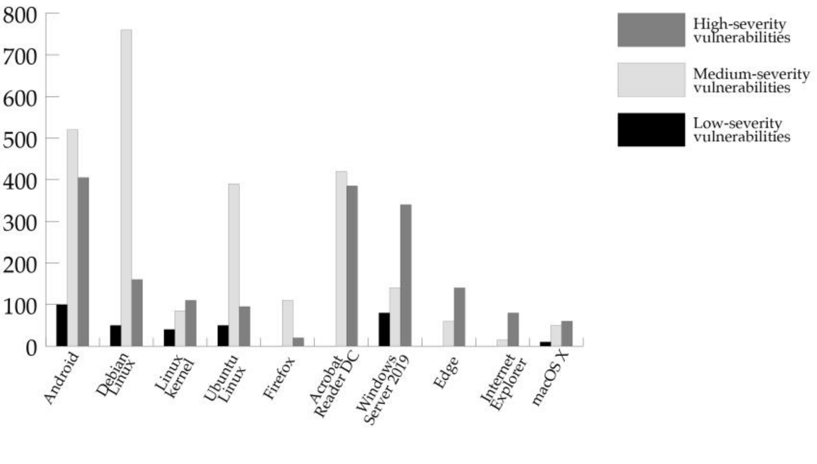

Figure 6. Number of CVE vulnerabilities by severity for 5 open-source projects (first 5 to the left) and 5 closed-source projects (last 5 to the right) between January 2018 and September 2019

However, this figure may be somewhat misleading due to several factors. First, projects that frequently introduce new features may also introduce security issues. Second, the selected period might conceal the number of security issues in the past. Third, differences in project management techniques between open-source and closed-source projects could influence the development and validation process of the product.

Additionally, although the code of open-source projects is publicly available, it does not necessarily make them more vulnerable to attack. There are several examples of severe vulnerabilities in closed-source products that were discovered years after their release. Furthermore, the responsibility of maintaining and securing the code may not be as appealing to contributors in open-source projects as developing new features, whereas closed-source projects often have designated teams for this purpose.

In conclusion, our examination of the security vulnerabilities in open-source and closed-source software projects, as represented in Figure 6 and substantiated by data from the Common Vulnerabilities and Exposures (CVE) dataset, reveals a nuanced landscape of software security. While the initial data suggests a higher incidence of vulnerabilities in open-source projects, this observation must be contextualized within the broader framework of software development practices and the inherent limitations of the study period. The dynamic nature of feature introduction, coupled with the variance in project management methodologies, contributes to the complexity of accurately assessing and comparing the security profiles of these projects. It is also noteworthy that the public availability of open-source code does not axiomatically correlate with increased vulnerability. In contrast, the history of software development contains instances of significant security oversights in closed-source projects, undetected for extended durations. This analysis underscores the multifaceted challenges in maintaining and securing codebases, particularly in open-source projects where the focus may predominantly be on feature development rather than security maintenance, a task often systematically addressed by dedicated teams in closed-source environments.

Key Takeaways:

- Security in Software Development: Security is a critical aspect in software development, requiring careful attention during code writing and deployment.
- Security Metrics: Focuses on assessing the security of both closed-source and open-source projects.
- Use of CVE Dataset: Utilizes the Common Vulnerabilities and Exposures (CVE) dataset to evaluate security issues in ten projects (five open-source, five closed-source) from January 2018 to September 2019.
- Severity Measurement: The severity of vulnerabilities is measured using the CVSS score, which considers factors like exploitability, impact, complexity, and scope.

- Vulnerability Distribution: Android showed the highest number of high-severity vulnerabilities, while Debian had the most medium-severity issues among open-source projects. Firefox had the fewest vulnerabilities.
- Closed-Source Project Vulnerabilities: Closed-source projects, except Acrobat Reader DC, generally had fewer low or medium-severity vulnerabilities compared to high-severity ones. Mac OS X had the best performance in this category.
- Comparison of Open vs. Closed-Source: Closed-source projects had almost half as many vulnerabilities as open-source projects.
- Factors Influencing Vulnerability Figures: Factors like frequent introduction of new features, the selected time period, and differences in project management techniques can affect the number of reported security issues.
- Misconception about Open-Source Security: Open-source projects, despite their public code access, are not inherently more vulnerable to attacks. In contrast, some closed-source products have had severe vulnerabilities undiscovered for years.
- Responsibility in Code Maintenance: In open-source projects, the focus on developing new features may overshadow the responsibility of maintaining and securing code, whereas closed-source projects often have dedicated teams for security.

5. Profit/Funding/Revenue

In the present paper, we examine the metric of profit or funding received by open-source projects or companies. This metric is significant because it serves to attract developers to work on various projects, as well as provide ongoing development and support for the codebase. Established projects that continue to receive funding benefit from increased visibility and greater stability for the future.

Open-source projects can acquire funding through several means, including direct funding from large tech companies that utilize their products. In contrast, smaller open-source projects must rely on donations from end users. Technical support for open-source products is also a way to generate revenue, but it is challenging to implement due to its open-source nature. While this may be a valid way to earn profit for specific projects, it is not the norm and will not be further discussed.

In contrast, closed-source projects generate revenue by selling the rights to access the software to end-users and by selling technical support services. There are other methods for closed-source companies to generate revenue, but they are beyond the scope of this paper. Figure 6 displays a comparison of three open-source projects/companies and three companies that develop closed-source projects.

The significant difference between the two categories is evident from the graph, with closed-source companies aiming to generate profit, while open-source projects do not prioritize profit-making. This difference is further highlighted in the comparison of Alphabet, Google’s parent company, which generated almost 35 billion dollars in 2019, and Linux, one of the largest and most well-funded open-source projects, which received only about 11 million dollars. Apache and Eclipse, other prominent open-source foundations, have even lower profit margins. However, this comparison may be misleading since it compares a project to a company. If we consider just a few of Google’s products, the difference may be more comparable [12][14][15][16][17].

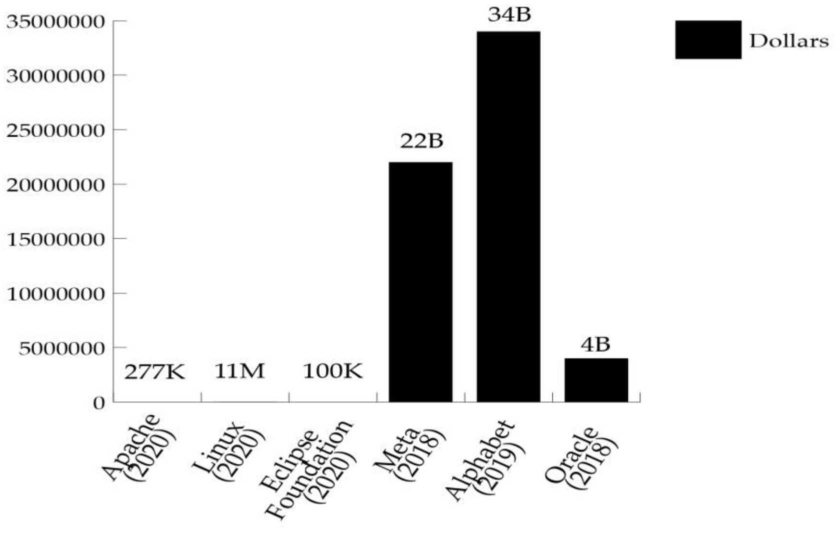

Figure 7. Income for open-source and closed-source projects in recent years.

In conclusion, the exploration of profit, funding, and revenue models in this chapter highlights the distinct financial pathways and priorities that distinguish open-source projects from their closed-source counterparts. Through an analysis of various funding sources and revenue generation mechanisms, it becomes evident that while open-source initiatives often depend on external support such as donations and corporate sponsorship, closed-source entities primarily derive income from direct software sales and associated services. This fundamental divergence in financial strategies underscores the differing objectives and operational paradigms of these two categories. Moreover, the comparison between the financial metrics of leading open-source projects and prominent closed-source companies, as illustrated in Figure 7, offers a revealing snapshot of the economic landscape within the software industry.

Key Takeaways:

- Profit/Funding in Open-Source Projects: The paper discusses the importance of profit or funding for open-source projects and companies, emphasizing its role in attracting developers and supporting ongoing development.
- Funding Sources: Open-source projects acquire funding through various means, including direct funding from large tech companies and donations from end users. Technical support services are a potential but challenging revenue stream.
- Comparison with Closed-Source Projects: Closed-source projects primarily generate revenue by selling software access rights and technical support services. This is in contrast to open-source projects, which don't primarily focus on profit-making.
- Revenue Differences: There's a significant difference in revenue between open and closed-source entities. For instance, Alphabet generated $35 billion in 2019 compared to Linux's $11 million. Apache and Eclipse have even lower profit margins.
- Misleading Comparisons: Comparing open-source projects with large companies like Alphabet can be misleading. A more comparable analysis would be between specific products of such companies and open-source projects.

6. Project management techniques

The final point of discussion in this paper concerns the importance of project organization for the success of both open-source and closed-source projects. While this comparison is based on opinions rather than empirical evidence, we argue that the organization method is a critical factor. Unfortunately, there is a lack of recent data on project management techniques for closed-source projects, as such practices tend to be kept confidential. Nevertheless, it is plausible that closed-source projects rely on Agile or Scrum development techniques with some adaptations to meet their specific needs. For example, Microsoft uses its Microsoft Solutions Framework (MSF) to guide software delivery [11].

In contrast, open-source projects have a distinct organizational structure that relies on the contributions of a community of developers rather than paid employees. Typically, a small team or core group of individuals governs the project, making decisions regarding the project's direction, such as integrating new features, adopting new tools or methodologies, and so on. These decisions can be made either by the governing body of the project or by a vote of the entire community of contributors and developers.

Both closed-source and open-source project management techniques have their advantages and disadvantages. Agile is commonly used in closed-source projects due to its simplicity and effectiveness, but it can also be rigid and inflexible. In contrast, open-source projects allow for greater flexibility in terms of when developers can contribute, but this can create uncertainty and instability in project planning.

In conclusion, project organization is crucial for the success of any project, whether open-source or closed-source. While there are differences in the organizational structures of these two types of projects, both have their strengths and weaknesses that could be improved upon. Our discussion, though grounded in informed suppositions rather than empirical analysis, highlights that the choice of organizational method is not merely procedural but fundamentally integral to project outcomes. The secretive nature of project management practices in closed-source environments, exemplified by the Microsoft Solutions Framework, contrasts sharply with the more transparent and community-driven approaches prevalent in open-source projects. This dichotomy presents distinct sets of advantages and challenges: the structured efficiency of Agile methodologies in closed-source projects versus the flexible yet occasionally erratic nature of open-source project management.

Key Takeaways:

- Project Organization Importance: The organization method is critical for the success of both open-source and closed-source projects.
- Closed-Source Project Management: There's a lack of recent data on closed-source project management techniques, which are often confidential. These projects may use Agile or Scrum with adaptations, like Microsoft's Microsoft Solutions Framework (MSF).
- Open-Source Project Structure: Open-source projects rely on community contributions and are typically governed by a small team. Decision-making can be centralized or based on community votes.
- Comparative Advantages and Disadvantages: Closed-source projects often use Agile for its simplicity and effectiveness, but it can be rigid. Open-source projects offer more flexibility but can lead to uncertainty in project planning.
- Conclusion on Project Organization: Both open-source and closed-source projects have unique strengths and weaknesses in their organizational structures, highlighting the importance of project organization for success.

7. Conclusion

The open-source community has drawn significant interest from large tech companies, as evidenced by the increasing number of contributors to open-source projects from these companies. This trend suggests that big tech is committed to open-source software development. The high number of contributors in an open-source project enhances its ability to create new features, update existing ones, and address bugs and security vulnerabilities.

The contrast between the developer-to-feature ratio in closed-source and open-source projects is stark. Closed-source projects are created to generate profits for their respective companies, and every feature is planned to meet the needs of a customer base. In contrast, open-source projects are often initiated to satisfy the needs of a few individuals, with no intention of bringing them to the market. The discrepancy in purpose partly explains the difference in the ratio of developers to features. Open-source projects are open to any developer who wishes to contribute to meet their own needs, regardless of whether others share those needs.

In terms of funding, open-source organizations operate with budgets that are several orders of magnitude smaller than those of large tech companies. Tech giants increase profit margins by pushing the boundaries of what has already been developed and by having dedicated departments that identify revenue opportunities. Open-source foundations, in contrast, rely on donations and sponsorships for their funding. While the number of contributors helps open-source projects to identify and develop features, the lack of structure within the development team can result in some features being less impactful and useful primarily to the authors or a small number of end-users.

Closed-source projects typically rely on well-defined hierarchies and structured project management methodologies, such as Waterfall or Agile. This approach has advantages, such as clearly defined project requirements and development stages. Open-source projects lack clear boundaries, which may explain why developers are often drawn to these projects despite the lack of financial incentives. However, the larger pool of contributors and the ability of all contributors to participate in project debates enable faster detection and resolution of potential issues.

In conclusion, both closed-source and open-source software development have their own set of advantages and disadvantages. The interest of large tech companies in open-source software can be attributed to several factors, including the number of people involved in open-source projects. The lack of clear boundaries in open-source projects encourages developers to explore new solutions and ideas, which may be harder to achieve within the more rigid structures of a company.

Key Takeaways:

- Large tech companies are increasingly involved in open-source projects, indicating a commitment to open-source software development.
- Open-source projects benefit from a high number of contributors, improving their capacity for feature creation, updates, and addressing bugs and security issues.
- There is a notable contrast in the developer-to-feature ratio between closed-source and open-source projects, due to differences in their objectives and development approaches.

- Closed-source projects aim for profit and are designed to meet specific customer needs, while open-source projects often start to satisfy individual developers' needs without market intentions.
- Open-source organizations have significantly smaller budgets compared to large tech companies and rely on donations and sponsorships.
- The lack of structure in open-source development can lead to features that are less impactful or useful to a limited audience.
- Closed-source projects use structured project management methodologies, offering clear project requirements and development stages.
- Open-source projects, lacking clear boundaries, attract developers due to the freedom to explore new solutions and ideas.
- Both closed-source and open-source software development have unique advantages and disadvantages.
- The interest of large tech companies in open-source software is driven by factors like the involvement of numerous people and the freedom for innovation in open-source projects.

References

[1] Kronser ANDREW - *Common Vulnerabilities and Exposures Dataset,* https://www.kaggle.com/datasets/andrewkronser/cve-common-vulnerabilities-and-exposures 5.01.2023

[2] https://opensourceindex.io - Open-Source Contributor Index. 27.01.2023

[3] Escobar EMILIO - *The state of open source on GitHub,* https://octoverse.github.com/2022/state-of-open-source 27.01.2023

[4] Lionel Sujay VAILSHERY - *Number of employees at the Microsoft Corporation from 2005 to 2022*, https://www.statista.com/statistics/273475/number-of-employees-at-the-microsoft-corporation-since-2005 14.01.2023

[5] Bianchi TIAGO - *Number of full-time Alphabet employees from 2007 to 2021* - https://www.statista.com/statistics/273744/number-of-full-time-google-employees 27.01.2023

[6] Dixon S., *Number of full-time Meta (formerly Facebook Inc.) employees from 2004 to 2021* - https://www.statista.com/statistics/273563/number-of-facebook-employees 27.01.2023

[7] Laricchia FEDERICA, *Apple's number of employees in the fiscal years 2005 to 2022.* https://www.statista.com/statistics/273439/number-of-employees-of-apple-since-2005 27.01.2023

[8] Vali TAWOSI, Afnan AL-SUBAIHIN, Rebecca MOUSSA, Federica SARRO, *A versatile dataset of agile open-source software projects. In Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. Association for Computing Machinery, New York, NY, USA, 707–711. https://doi.org/10.1145/3524842.3528029

[9] https://www.microsoft.com/en-us/microsoft-365/roadmap - Microsoft 365 roadmap. 13.01.2023

[10] Zandt FLORIAN - *How Big Tech Contributes to Open Source*, https://www.statista.com/chart/25795/active-github-contributors-by-employer 13.01.2023

[11] https://www.microsoft.com/en-us/download/details.aspx?id=3214 - Microsoft Solutions Framework Core Whitepapers. 27.01.2023.

[12] https://www.sec.gov/Archives/edgar/data/1341439/000119312518201034/d568983d10k.htm - *Oracle Corporation, Form 10-K, Annual Report 2018*. 13.01.2023

[13] https://projects.propublica.org/nonprofits/organizations/460503801 - The Linux Foundation, Tax Filings by Year. 27.01.2023

[14] https://www.eclipse.org/org/foundation/reports/2020_annual_report.php - Eclipse Foundation, 2020 Annual Eclipse Foundation Community Report. 27.01.2023.

[15] https://www.sec.gov/Archives/edgar/data/1326801/000132680119000009/fb-12312018x10k.htm - Facebook, Inc., Form 10-K, Annual Report 2018. 27.01.2023

[16] https://www.sec.gov/Archives/edgar/data/1652044/000165204420000008/goog10-k2019.htm - Alphabet Inc., Form 10-K, Annual Report 2019. 27.01.2023

[17] https://www.apache.org/foundation/docs/FY2020AnnualReport - The Apache Software Foundation, Annual Report FY2020. 27.01.2023

Bibliography

Bianchi TIAGO - *Number of full-time Alphabet employees from 2007 to 2021* - https://www.statista.com/statistics/273744/number-of-full-time-google-employees 27.01.2023

BONACCORSI, A., & ROSSI, C. (2003). Why Open Source Software Can Succeed. Research Policy, 32(7), 1243-1258.

CAPILUPPI, A., & FERNANDEZ-Ramil, J. (2008). An Empirical Study of Open Source and Closed Source Software Products. IEEE Transactions on Software Engineering, 34(6), 1-17.

CROWSTON, K., & HOWISON, J. (2006). The Social Structure of Free and Open Source Software Development. First Monday, 11(2).

CROWSTON, K., & SCOZZI, B. (2002). A Repository of Open Source Software: Infrastructure, Design, and Behavior. In Proceedings of the 2002 ACM Workshop on Interdisciplinary Software Engineering Research (pp. 37-44).

Dixon S., *Number of full-time Meta (formerly Facebook Inc.) employees from 2004 to 2021* - https://www.statista.com/statistics/273563/number-of-facebook-employees 27.01.2023

Escobar EMILIO - *The state of open source on GitHub,* https://octoverse.github.com/2022/state-of-open-source 27.01.2023

FITZGERALD, B. (2006). The Transformation of Open Source Software. MIS Quarterly, 30(3), 587-598.

Kronser ANDREW - *Common Vulnerabilities and Exposures Dataset,* https://www.kaggle.com/datasets/andrewkronser/cve-common-vulnerabilities-and-exposures 5.01.2023

Laricchia FEDERICA, *Apple's number of employees in the fiscal years 2005 to 2022.* https://www.statista.com/statistics/273439/number-of-employees-of-apple-since-2005 27.01.2023

Lionel Sujay VAILSHERY - *Number of employees at the Microsoft Corporation from 2005 to 2022*, https://www.statista.com/statistics/273475/number-of-employees-at-the-microsoft-corporation-since-2005 14.01.2023

MOCKUS, A., FIELDING, R. T., & HERBSLEB, J. D. (2002). Two Case Studies of Open-Source Software Development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology (TOSEM), 11(3), 309-346.

TIEMANN, M. (2002). Open-Source Software: A (New?) Development Methodology. Economic Review, 87(3), 17-29.

Vali TAWOSI, Afnan AL-SUBAIHIN, Rebecca MOUSSA, Federica SARRO, *A versatile dataset of agile open-source software projects. In Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. Association for Computing Machinery, New York, NY, USA, 707–711. https://doi.org/10.1145/3524842.3528029

WEST, J., & O'MAHONY, S. (2008). The Role of Participation Architecture in Growing Sponsored Open-Source Communities. Industry and Innovation, 15(2), 145-168.

Zandt FLORIAN - *How Big Tech Contributes to Open Source*, https://www.statista.com/chart/25795/active-github-contributors-by-employer 13.01.2023

https://apache.org/foundation/docs/FY2020AnnualReport - The Apache Software Foundation, Annual Report FY2020. 27.01.2023 https://eclipse.org/org/foundation/reports/2020_annual_report.php - Eclipse Foundation, 2020 Annual Eclipse Foundation Community Report. 27.01.2023.

https://microsoft.com/en-us/microsoft-365/roadmap - Microsoft 365 roadmap. 13.01.2023

https://microsoft.com/en-us/download/details.aspx?id=3214 - Microsoft Solutions Framework Core Whitepapers. 27.01.2023.

https://opensourceindex.io - Open Source Contributor Index. 27.01.2023

https://projects.propublica.org/nonprofits/organizations/460503801 - The Linux Foundation, Tax Filings by Year. 27.01.2023

https://sec.gov/Archives/edgar/data/1341439/000119312518201034/d568983d10k.htm - *Oracle Corporation, Form 10-K, Annual Report 2018*. 13.01.2023

https://sec.gov/Archives/edgar/data/1326801/000132680119000009/fb-12312018x10k.htm - Facebook, Inc., Form 10-K, Annual Report 2018. 27.01.2023

https://sec.gov/Archives/edgar/data/1652044/000165204420000008/goog10-k2019.htm - Alphabet Inc., Form 10-K, Annual Report 2019. 27.01.2023