AUGMENTING BUS FACTOR ANALYSIS WITH VISUALIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Muhammad Umair Ahmed
January 2024

Augmenting Bus Factor Analysis with Visualization
By Muhammad Umair Ahmed
January 2024

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Eray Tüzün(Advisor)

Uğur Doğrusöz

Engin Demir

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

AUGMENTING BUS FACTOR ANALYSIS WITH VISUALIZATION

Muhammad Umair Ahmed
M.S. in Computer Engineering
Advisor: Eray Tüzün
January 2024

'Bus factor', also known as 'truck factor', is a measure of how vulnerable a software project is based on the minimum number of people who would have to leave the project (be 'hit by a bus') for it to stall. There is existing research on how to calculate bus factor for software projects but limited work on visualizing the bus factor. We believe providing visualization along with conventionally provided numerical bus factor results will help decision-makers manage the workload and knowledge distribution across the project and also help in planning and hiring decisions.

This thesis proposes, implements, and evaluates a tool named BFViz to visualize bus factor and contributions for software projects from pre-processed Git history. It is a web application that provides a file-browser-like interface with an interactively navigable treemap. Additionally, it has filename-based filtering, individual contribution data for files and folders, and simulation of contributor departure.

The tool is validated with a round of four user evaluations where users, ranging from project owners and engineering managers to developers, complete tasks using the tool on an open-source project that they are involved in and provide feedback with a semi-structured interview and a feature ranking activity. The overall task completion rate for the tasks was 79.55%. All case study participants preferred BFViz over text reports to understand bus factor data. The top three features, by mean ranking, were the contributors' list, the files and folders' visualization, and the simulation mode.

Keywords: bus factor, knowledge distribution, truck factor, pony factor, risk management, visualization.

ÖZET

OTOBÜS FAKTÖRÜ ANALİZİNİN GÖRSELLEŞTİRME İLE GÜÇLENDİRİLMESİ

Muhammad Umair Ahmed
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Eray Tüzün
Ocak 2024

'Otobüs faktörü', 'kamyon faktörü' olarak da bilinir, bir yazılım projesinin ne kadar savunmasız olduğunu ölçen bir metriktir ve projenin durması için kaç kişinin projeden ayrılması gerektiğini belirler. Yazılım projeleri için otobüs faktörünü hesaplama üzerine mevcut araştırmalar bulunsa da, otobüs faktörünü görselleştirmek üzerine sınırlı çalışma vardır. Görsel olarak sunulan sayısal otobüs faktörü, karar vericilerin projedeki iş yükünü ve bilgi dağılımını yönetmelerine, ayrıca planlama ve işe alım kararlarında yardımcı olabileceğine inanıyoruz.

Bu tezde, Git geçmişinden yararlanarak, yazılım projeleri için otobüs faktörü ve projeye katkılarını görselleştirmek için BFViz adlı bir araç geliştirilmiş ve değerlendirilmiştir. Bu araç, etkileşimli gezilebilir bir ağaç haritası, filtreleme, bireysel katkı verileri ve katkıda bulunanın ayrılışı simülasyonu ile bir dosya tarayıcısı benzeri bir arayüz sunan bir web uygulamasıdır.

Bu uygulama, dört kullanıcı değerlendirme turu ile doğrulanmıştır. Proje sahiplerinden ve mühendislik yöneticilerinden geliştiricilere kadar çeşitli kullanıcılar, üzerinde çalıştıkları açık kaynaklı bir projede aracı kullanarak görevleri tamamlamıştır. Kullanıcılardan, yarı yapılandırılmış bir mülakat ve özellik sıralama etkinliği kullanılarak geri bildirim alınmıştır. Görevlerin genel tamamlama oranı %79.55 olarak hesaplanmıştır. Tüm vaka çalışması katılımcıları, metin raporları yerine BFViz'i kullanılarak otobüs faktörü verilerini anlamayı tercih etmiştir. Ortalama sıralama ile en üst üç özellik, katkıda bulunanların listesi, dosya ve klasörlerin görselleştirmesi ve simülasyon modu olmuştur.

Anahtar sözcükler: Otobüs faktörü, risk yönetimi, veri dağıtımı, tır faktörü, görselleştirme.

Acknowledgement

First and foremost, I would like to acknowledge that this work would be impossible if it were not for the unwavering belief and backing I had of my supervisor Dr. Eray Tüzün. I hope I can repay the faith and patience he has displayed in me through the highs and lows during my time as a student and as a researcher, the freedom and confidence he afforded me at every step, and pass on the values he instilled in me and which I hold so dear.

This thesis would also not be possible without the help of our friends at the Intelligent Collaboration Tools Lab at JetBrains N.V.: Vladimir, Misha, Nikolai, Egor, and everyone else, but most of all, Pouria who remained undeterred with his efforts despite serious difficulties at just about every turn.

I'm also eternally grateful to Bilkent University, the administrators, and the admissions committee(s) who took a gamble on me not once, but twice, paving the way for me to receive a quality education that would otherwise have been simply out of reach.

This would also not have been possible without the support of my family, especially my mother, who fought against all odds to afford me the best possible education we could afford, and my father who spent his life's earnings on the same, far beyond his means.

This study was supported by The Scientific and Technological Research Council of Turkey (TUBITAK) 3501 program (Project Number: 121E584).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software development is an activity that often takes place in a team setting. Generally, the number of engineers working on a piece of software increases in line with the number and complexity of features it has. Enterprise software projects are sometimes so complex that there are hundreds of engineers working on them at the same time. The Kotlin project [1], for example, has 600 contributors listed on its GitHub repository. The IntelliJ IDEA Community Edition project [2] has 934 contributors listed on its GitHub repository.

Whenever multiple engineers are working on developing software, there is a certain knowledge distribution that forms according to the individual members' involvement in the development process. If one or a few team members hold crucial information about a software project, then such a software project is at risk of stalling if these members suddenly depart the team. To identify how vulnerable a software project is to this loss, a metric called 'Bus Factor' [1, 2] was introduced by researchers. A project's 'Bus Factor' is the minimum number of team members that need to be 'hit' by the proverbial bus for the project to stall.

There have been multiple studies on calculating the bus factor for software

[1] https://github.com/JetBrains/kotlin
[2] https://github.com/JetBrains/intellij-community

projects. One of the first algorithms to be published was by Zazworka et al. [3]. They suggested a combinatorics-based method to visualize the truck factor characteristics of a software project from code repository data. Cosentino et al. [1] developed a tool to assess the bus factor for software projects held in Git repositories. Their tool calculates the bus factor by ascertaining developer knowledge and generating a list of key developers from this data. They also provided a visual simulation of what would happen in case one or more authors left the project. Avelino et al. [2] also developed a heuristics-based algorithm to estimate the truck factor for software projects and then validated it with surveys addressed to the developers and maintainers of the projects. Jabrayilzade et al. [4] built upon previous work to better estimate the bus factor in software projects by including non-code artifacts such as meetings and documentation in their analysis. Haratian et al. [5] incorporate file importance metrics into existing bus factor algorithms to improve its output. These algorithms are discussed and compared in detail in Chapter 2.2.

Most of the work available in this research area presents bus factor in numbers. It is straightforward to consume data in the form of numbers or text in small amounts but visualization helps when the data scales up. Data visualization also makes it easier to spot patterns, outliers, abnormalities, and missing values in the data that might otherwise require meticulous analysis to identify [6].

Having estimates of the bus factor for software projects and their modules helps in identifying potential knowledge distribution problems, they do not indicate the scale of the problem. We hypothesize that visualizing bus factor information is more effective in conveying the scale of the problem and pointing out the areas where the problem originates from compared to only providing bus factor as numbers. We also propose that an interface to explore the effect of contributors leaving their project on the bus factor and health of the project will be appealing to certain team members working on software projects. In particular, such an interface is expected to appeal to team members in management positions or responsible for the overall health and risk management of projects. The area of visualizing bus factor is not comprehensively covered when writing this thesis. There are a few tools that help visualize bus factor in different ways but we could

not find any tool that explores the effect of contributors leaving projects on their bus factor.

In this thesis, we develop an application that can visualize the bus factor for a project directory-by-directory using illustrative methods such as treemaps, including the ability to scope down into the project directory tree and check the status of different parts of the project in terms of bus factor. Our tool differs from Cosentino et al.[1] in that it provides a detailed, navigable view of files and folders. It also allows to simulate any contributor's departure to predict the effect on the bus actor. We utilize the Jabrayilzade [4] algorithm in our tool. We then validate the usefulness of this application by performing case studies with four participants working on three different software projects. The research questions we are looking to answer are:

RQ1 What specific features of the BFViz did users find useful to measure the risk and dependence of a team on individual team members?

RQ2 Are the features implemented in BFViz are easy to use?

RQ3 Do decision-makers find BFViz a helpful tool to measure the risks and dependence of a team on individual team members?

In Chapter 2, we discuss the origin and significance of the bus factor problem. We also examine existing literature on calculating and visualizing bus factor. In Chapter 3, we describe a web-based tool we developed that enables consuming bus factor information in a visual form. Chapter 4 describes the evaluation study for our tool. In Chapter 5, we present the results of our evaluation. In Chapter 6, we discuss the results obtained, in Chapter 7, we discuss the threats to the validity of this thesis, and in Chapter 8, we draw some high-level conclusions from the case studies.

Chapter 2

Background

2.1 The Bus Factor Problem

"Bus factor", also referred to as "bus number", "truck factor", "lottery number" and other names, can loosely be defined as a measure of estimating risk in software projects from employee turnover. The first known discussion of the concept can be identified in a Python mailing list discussion from 1994 titled "If Guido was hit by a bus?" [1] in which Michael McLay questions the feasibility of adopting Python due to its dependency on a single person: its creator Guido van Rossum.

The first concrete definition of the bus factor is provided by Coplien [7] as the minimal number of developers that would have to be hit by a bus before the project stalls. A higher bus factor indicates that the software project is more resilient to developers leaving. A low bus factor on the other hand indicates that the project is in a vulnerable position and may stall if the developers making up this number were to leave. This definition is carried forward in the work appearing later in this area.

The provided definitions of the bus factor also align with the investigation of

[1] `https://legacy.python.org/search/hypermail/python-1994q2/1040.html`

similar concepts in other areas. Software engineering is classified as a knowledge-intensive pursuit [8], hence knowledge loss by employee turnover applies to software engineering too. Rashid et al. [9] found during their investigation that employee turnover is one of the leading factors contributing to knowledge loss in open-source software projects.

There is also recognition from the practical side of this concept. Some strongly recommended practices in the software engineering community, such as code reviews and pair programming help even out the distribution of knowledge [10, 11, 12], hence also mitigating the risk of having a low bus factor.

The bus factor problem itself occurs when the bus factor of a software project goes down to zero when one or more team members leave. When the bus factor decreases to zero, it means that the project is 'stalled' or incapacitated.

2.2 Bus Factor Algorithms

Researchers have proposed multiple algorithms based on different aspects of software projects to calculate its bus factor. Most such algorithms, such as the ones by Zazworka et al. [3], Rigby et al. [13], Cosentino et al. [1], and Avelino et al. [2] base their results on the version control history of the software projects as it provides an accurate picture of members' contribution to the codebase. Jabrayilzade et al. [4] also use supplementary data such as meeting and code review data to augment their results. We will discuss these algorithms and their strengths and weaknesses briefly in chronological order.

Zazworka et al. [3] present one of the earliest mathematical definitions of bus factor for a software project based on its version control history. In their definition of the truck factor, they first identify the set of files that each developer is associated with. The association is binary: if a developer has edited a file, they are considered to be associated with it. They then calculate a coverage metric $cov_x(n)$ that is defined as the percentage of the code that would still be

covered if n developers left. They specifically target three types of coverage: minimum, average, and maximum. The worst case is minimum coverage ($x = min$), implying that the set of n developers leaving are the ones with the most exclusive knowledge. The best case is maximum coverage, implying that the set of n developers leaving are the ones with the least exclusive knowledge.

The bus factor is presented as a plot of maximum, average, and minimum coverage against the size of the set of leaving developers (n). This way, decision-makers would be able to get an idea of the best and worst-case truck factor of their project at the coverage level they would like to maintain.

The first drawback of this algorithm is that it is not validated by the authors [14]. The second drawback is that it does not take into account the weight of the contribution of a developer to the files they edit. It treats all contributions equally. This is seldom the case in real-life projects as Ricca et al. [14] noticed during their implementation and application of it to real-life projects.

The third problem with this algorithm is its runtime complexity. Ricca et al. [14] identify that this approach runs into scalability problems, especially as $n > 30$. The calculation for some such projects took multiple days to complete. Hannebauer et al. [15] further prove that Ricca et al. [14] approach of calculating the bus factor using Zazworka's metric [3] is NP-hard.

Rigby et al. [13] improved upon Zazworka et al. [3] by using Monte Carlo simulation to estimate the truck factor for large software projects such as Google Chrome. They use git blame data as input for their algorithm. They simulate developers leaving with set size g ranging from one to up to 200. They estimate the likelihood for each value of g as well as the range of the number of files that will be abandoned for each value of g.

Cosentino et al. [1] describe an algorithm and provide a tool to visualize bus factor based on their algorithm. They calculate file ownership for each file in a project from its Git commit data, including per-line contribution data for text (source code, configuration, and documentation) files. They then assign a

knowledge percentage for every file, folder, branch, and the project itself to each developer in the project. This knowledge percentage is then utilized further in bus factor calculation. The knowledge percentage for each line is based on one of four contribution metrics: *Last Change Takes All, Multiple Changes Equally Considered, Non-Consecutive Changes*, and *Weighted Non-Consecutive Changes*. The first two are self-explanatory. The Non-Consecutive Changes metric squashes multiple consecutive changes to a line by the same developer and counts it as one. Weighted Non-Consecutive Changes is the same metric as Non-Consecutive Changes except weights are applied to the changes based on recency. Using each of these metrics, the authors then set minimum modification percentage thresholds to decide on sets of Primary and Secondary developers. The threshold for primary developers, X, for a given artifact is set as $X = 1/N$ where N is the number of overall developers that have modified that artifact. The same threshold for the set of secondary developers is set as half of that of primary developers $X_{secondary} = X_{primary}/2$. The size of the combined set of these developers is then deemed to be the bus factor.

Avelino et al. [2] propose an algorithm based on the Degree of Authorship (DOA) metric from Fritz et al. [16] on developers' knowledge of code. After some preprocessing of the Git data, namely trimming files to target, and tackling developer identifier aliases (multiple IDs for the same developer), they calculate the DOA for each developer and file. Here is the formula Avelino et al. [2] use to calculate the degree of authorship.

$$\begin{aligned} DOA(e,f) = 3.293 &+ 1.098FA \\ &+ 0.164DL \\ &- 0.321\log(1+AC) \end{aligned} \tag{2.1}$$

This formula calculates the Degree of Authorship of engineer e to file f. FA is a boolean indicating whether engineer e created the file f: 1 if yes, 0 if not. DL is the number of commits engineer e has made to file f. AC is the number of commits engineers other than e have made to file f. They then normalize the

Degree of Authorship metric to [0, 1]. To account for anomalous outliers, they limit the minimum normalized and absolute DOA to be greater than specific minimum values for a developer to be counted as an author of a file. These minimums are $k = 0.75$ for the normalized DOA and $m = 3.293$ for the absolute one respectively. A file is considered abandoned if all authors of the file leave the project. To obtain the bus factor of the project or any of its files, they iteratively remove the authors with the highest DOA scores across the project until the remaining authors cover less than 50% of the artifacts.

The main advantage of the usage of this DOA in Avelino's algorithm is that it does not treat changes to every file with the same weight. Instead, it weighs the changes based on how many changes a developer e has applied to that file, increasing DOA, and how many changes other developers have applied to that file, decreasing DOA. This is unlike the other algorithms discussed so far, which don't take contribution distribution into account.

Ferreira et al. [17] evaluated three of the algorithms discussed here, Rigby's algorithm, Cosentino's algorithm, and Avelino's algorithm. They found that both Cosentino and Avelino algorithms perform much better than Rigby's algorithm in accuracy. Avelino's algorithm edges out Cosentino's algorithm in most cases. They also found out that the higher the expected truck factor of a project, the lower the accuracy across the board for these algorithms.

Jabrayilzade et al. [4] build upon previous works, specifically those of Fritz et al. [16] and Avelino et al. [2]. They introduce a modified version of the Degree of Authorship formula. Their formula also includes recency as a factor in determining the Degree of Authorship between an engineer and a file. They also add meetings and code reviews as factors in the calculation of DOA.

$$
\begin{aligned}
DOA_i = {} & 3 \cdot FA + \sum_j DL_i^j + \frac{1}{2} \sum_j RV_i^j \\
& + \sum_l \min(1, \sum_j \frac{MT_i^{l,j}}{MTE}) \\
& + 2.4 \log(1 + \sum_k \sum_j DL_k^j) \\
& + 1.2 \log(1 + \sum_k \sum_j RV_k^j) \\
& - 2.4 \log(1 + \sum_{k \neq i} \sum_j DL_k^j) \\
& - 1.2 \log(1 + \sum_{k \neq i} \sum_j RV_k^j) \\
FA = \exp & \left(\frac{-t_{FA}}{S} \right), DL_i^j = \exp \left(\frac{-t_{DL^j}}{S} \right) \\
RV_i^j = \exp & \left(\frac{-t_{RV^j}}{S} \right), MT_i^j = \exp \left(\frac{-t_{MT^j}}{S} \right)
\end{aligned}
\tag{2.2}
$$

In this equation, DOA_i is the contribution of an engineer to a specific file. FA is a boolean variable representing first authorship. DL_i^j is the contribution of engineer i to file j. RV_i^j is the number of reviews done by engineer i on file j. $MT_i^{l,j}$ is the time spent by engineer i on meeting j on commit l. MTE is the maximum effective time, that Jabrayilzade et al. [4] set at 240 minutes.

To calculate the bus factor from their version of the DOA metric, they reuse Avelino's approach. They assign file authorship scores to developers using the DOA metric and iteratively move the authors with the highest DOAs to a separate list until the remaining authors cover less than 50% of the files. At this point, the authors in the separate list are declared to be the *key developers*. The count of developers in this list is declared to be the bus factor.

Haratian et al. [5] approach the bus factor estimation problem from a file importance perspective. The bus factor algorithms that have been discussed up till this point do not take this factor into account. Instead, a majority of them operate on commit counts, valuing each commit equally. They employ network analysis metrics such as PageRank [18], and centrality measures such as in-degree,

out-degree, and betweenness to analyze file significance. They then integrate the obtained file significance scores into the Avelino [2] and Jabrayilzade [4] bus factor estimation algorithms, reporting a decrease of 17% and 18% in Normalized Mean Absolute Error against the base Avelino and Jabrayilzade algorithms respectively.

Lisan et al. [19] propose a version of the Cosentino [1] algorithm which uses line-based instead of commits as the original algorithm does. They modify the algorithm to use two other metrics, lines of code changes (LOCC) and the cosine difference of lines of code over five different GitHub projects. They compare the results of the modified algorithm with the original one, with their implementation of Rigby's *git-blame*-based algorithm[13], as well information about key developers and estimated bus factors that they request from the maintainers of the project. They find that their implementation with LOCC and cosine difference of lines of code are better than the original implementation which counts commits.

Orrei et al [20] attempt to improve another similar informal metric called the 'pony factor'. Their aim is slightly different in identifying the key developers. The novelty in their approach to existing bus factor and truck factor estimation algorithms is that they use lines of code instead of the number of commits and also try to qualify their findings by targeting their analysis on file types.

2.3 Visualization of Bus Factor

There have been several algorithms to calculate bus factor in literature as listed before including by Cosentino et al. [1], Avelino et al. [2], and Jabrayilzade et al. [4] which have been proposed. Works describing how to visualize bus factor are few, however, according to the best of authors' knowledge.

The one tool that focuses solely on calculating and visualizing the bus factor of Git repositories is the tool built by Cosentino et al. [1]. The visualization represents branches, directories, files, and extensions, as square blocks and assigns

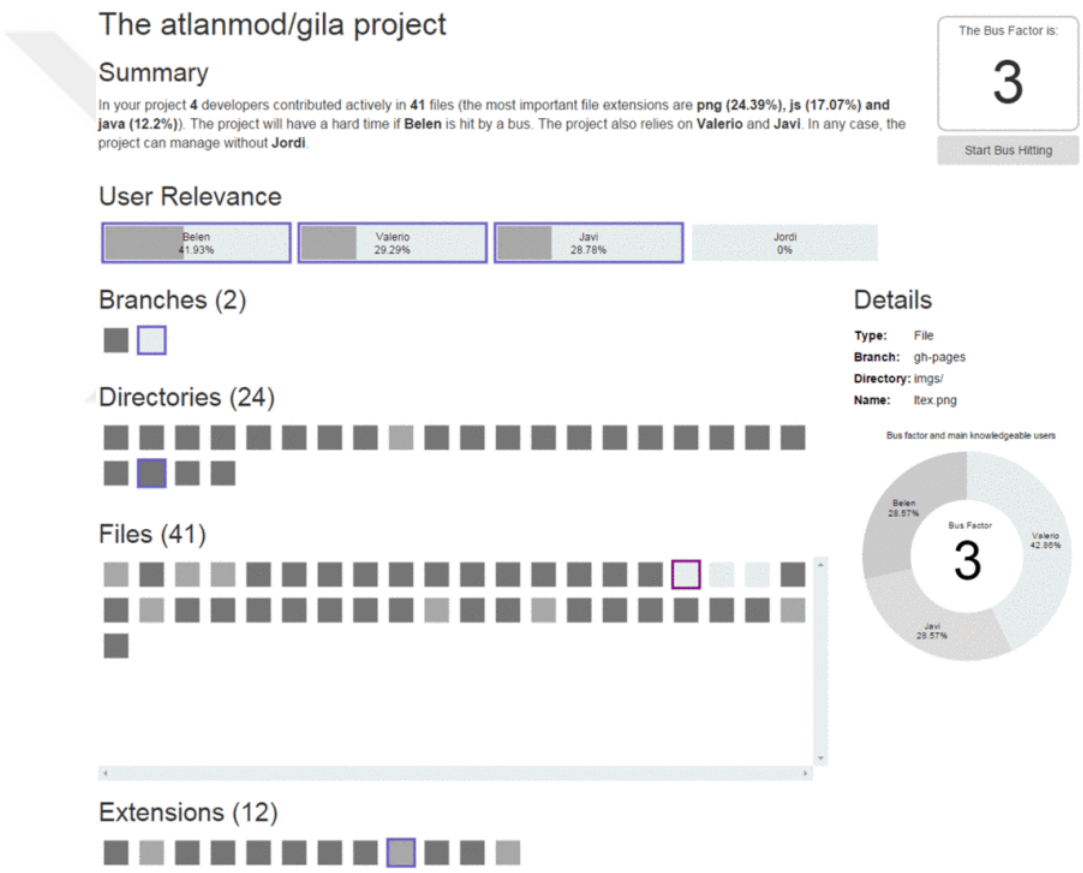

Figure 2.1: A screen capture of the visualization of the bus factor from the tool by Cosentino et al. [1]

colors to them based on which developer is the main contributor. The blocks also change color on developer departure simulation. A screengrab of the tool can be seen in Figure 2.1.

There are a few works that visualize software repositories and include bus factor in their visualizations but they often include it as a side feature. These include free and open-source offerings like Git-Truck [21]. Git-Truck is an open-source project that focuses on analyzing the evolution of Git-based repositories using circle-packing or treemap visualization. It provides coloring schemes based on a few metrics such as top contributor and bus factor but its primary purpose is to show how a Git repository has evolved.

Other tools that mine software repository data often fall short in how they visualize bus factor data such as the GrimoireLab[2] suite which only visualizes it using a pie chart.

There are also some paid options for visualizing software repositories such as CodeScene[3]. We do not include them in our discussion as they are commercial products and bus factor visualization is an additional feature in their software suite.

We propose, implement, and perform user evaluation case studies on a tool to visualize the bus factor on software projects. The tool offers navigable visualizations of files and folders with bus factor information, a simulation of contributor departure and its effect on files and folders, as well as contribution details on files and folders. This tool is called BFViz.

At the time of writing this thesis, the Bus Factor Explorer tool [22], is the most complete one that allows for bus factor visualization as well as for importing projects and exporting bus factor data. This tool is essentially a wrapper around BFViz with extra data customization and handling features that can be used as a complete bus factor visualization and analysis solution.

[2]https://chaoss.community/kb/metric-bus-factor/
[3]https://codescene.com/

Chapter 3

BFViz

What we propose is an approach that can be used to visualize the bus factor of a software project using its Git repository data, complete with navigation features across the folders and files contained in it. Additionally, we also propose a feature, which we call *simulation mode* to allow us to view the effect of one or more contributors leaving the repository on the bus factor. A more complete list of features is described below.

3.1 Features

3.1.1 Treemap Visualization

We visualize the structure of the repository in the form of a treemap. The users can view the name and bus factor of the files and folders which are shown as tiles. Each tile represents a file or folder. The size of the tile is determined by its size in bytes using a logarithmic scale. Each tile is assigned a color marking the range its bus factor initially falls in. There are four categories of bus factor ranges, Not Applicable, Dangerous, Low, and OK. Their initial ranges are shown in Figure 3.2. The boundaries of these ranges are customizable by the user. The

Figure 3.1: BFViz main page

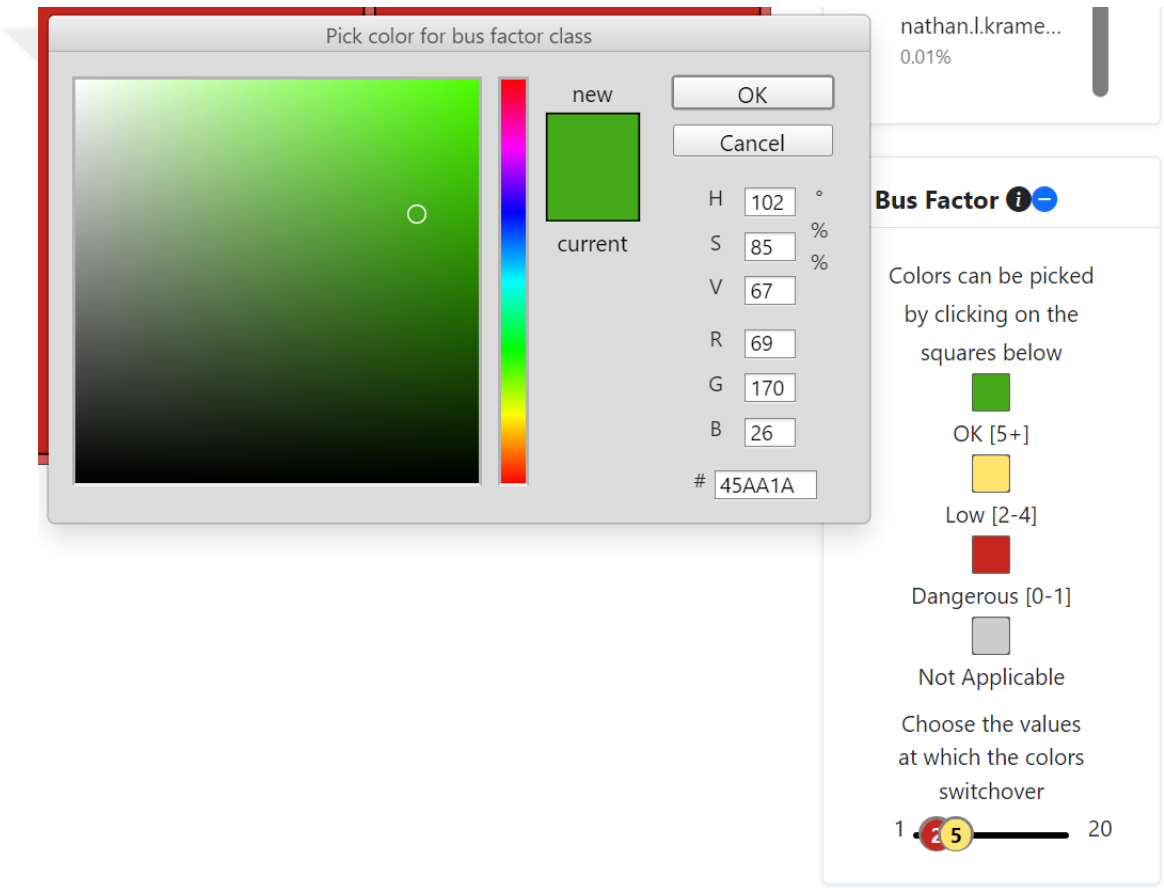

Figure 3.2: BFViz color selector panel

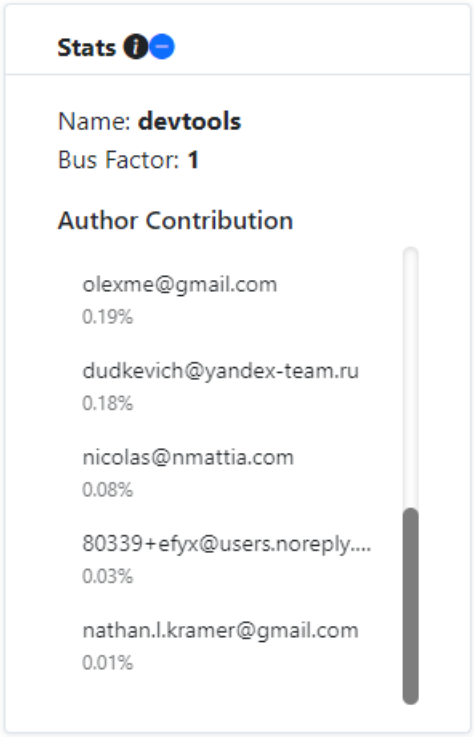

Figure 3.3: BFViz contribution statistics panel

colors assigned to them are also customizable using a color picker.

3.1.2 Contribution Statistics

The contribution statistics section shows the top contributing members to the currently selected folder or file. It is shown as a list of contributor usernames with their relative contribution percentages in descending order. Figure 3.3 shows the panel closely.

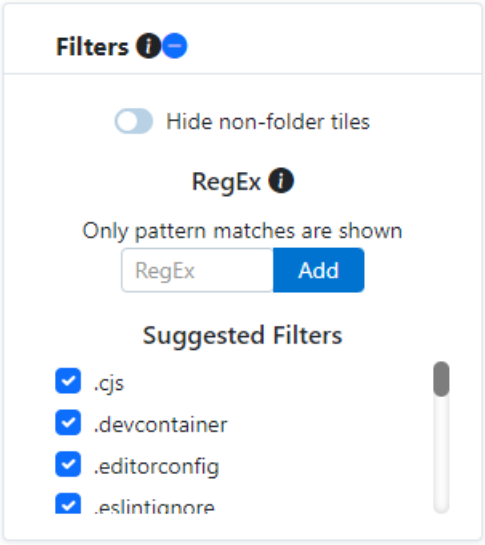

Figure 3.4: BFViz filter panel

3.1.3 Navigation

Navigation across the directory tree of the project can be performed in three main ways in BFViz: By clicking on folder tiles on the treemap itself, which will navigate into the clicked folder, or by using the navigation island. The **Up** button on the navigation island allows the user to go to the parent directory of the current location. The **Home** button takes the user to the root directory of the project. Additionally, the **Current Location** panel shows the current path. The intermediate folders in this path can be clicked to navigate to those folders. Figure 3.10 provides a closer look at the current path panel. Figure 3.9 shows the navigation island.

3.2 Layout

Other than navigating to different folders, users can interact and modify the layout of the treemap visualization in two ways:

3.2.1 Configuration Panel

Figure 3.8 shows some options for configuring the layout of the treemap visualization. The users can select the sorting key, the sorting order, and the D3 layout algorithm to use for laying out the treemap.

3.2.2 Filtering

Users can filter out files and folders that they might not be interested in using the Filtering panel as shown by Figure 3.4. There are two options to filter: Either users can provide a regular expression pattern and only files and folders with matching names will be shown on the treemap. The other option is to use the list of *Suggested Filters* and uncheck extensions and filenames from there that the user wants to be filtered out. These options can be used in tandem as well. The filtering is only visual: It does not cause any changes to the bus factors of the files and folders.

3.2.3 Simulation Mode

This is the feature of the application that allows the user to simulate the effect of one or more contributors leaving the project. The interface with a separate treemap, from the main visualization, can be seen in Figure 3.5. A closeup of just the treemap is shown in Figure 3.6. The contributors list on the right has checkboxes next to their names. A closeup of it is shown in Figure3.7. Unchecking the checkboxes leads to their simulated removal from the project and the bus factor of the project is recalculated without this contributor. The treemap then shows the difference in bus factor in the simulation mode treemap. If there is just a decrease in the bus factor, the tile is highlighted in yellow. If there is a decrease such that the bus factor of the tile overall has dropped to zero, it is highlighted in red. The number on the tiles indicates the change in the bus factor according to the current selection of contributors removed.

3.3 Implementation

The implementation of BFViz is divided into two parts: the analysis, and the visualization. The *backend* is responsible for ingesting pre-analyzed project folder and file data with bus factor statistics and performing further calculations on them. The BFViz application itself does not calculate the bus factor from the Git repository of the software projects. Instead, we use the publicly available Kotlin implementation[1] of Jabrayilzade's [4] algorithm to calculate the bus factor of software projects. The calculations explicitly performed by BFViz are only needed to simulate the impact of developer departure on the bus factor.

The front end is responsible which is the interactive portion that the users will interact with. Git repository data is fed into a separate implementation of Jabrayilzade's algorithm to produce output data that is then consumed by BFViz to provide visualization and related features.

3.3.1 Analysis

The analysis or the calculation is performed using a modified version of the algorithm specified by Jabrayilzade et al. [4] excluding meetings and reviews data. The algorithm relies on the Degree of Authorship concept and is an improvement over Avelino et al. [2]. We chose it because it had the best evaluation metric results for bus factor calculation at the time of writing this thesis.

The categorization of files ignored in the bus factor calculation (classified as 'Not Applicable' as shown earlier) is performed as part of the bus factor calculation. This is done by manually specifying a list of file extensions to ignore in the bus factor calculations.

Bot filtering and merging users are both performed before the bus factor calculation algorithm is executed. Bot filtering is done by checking the user names

[1] https://github.com/JetBrains-Research/bf-core

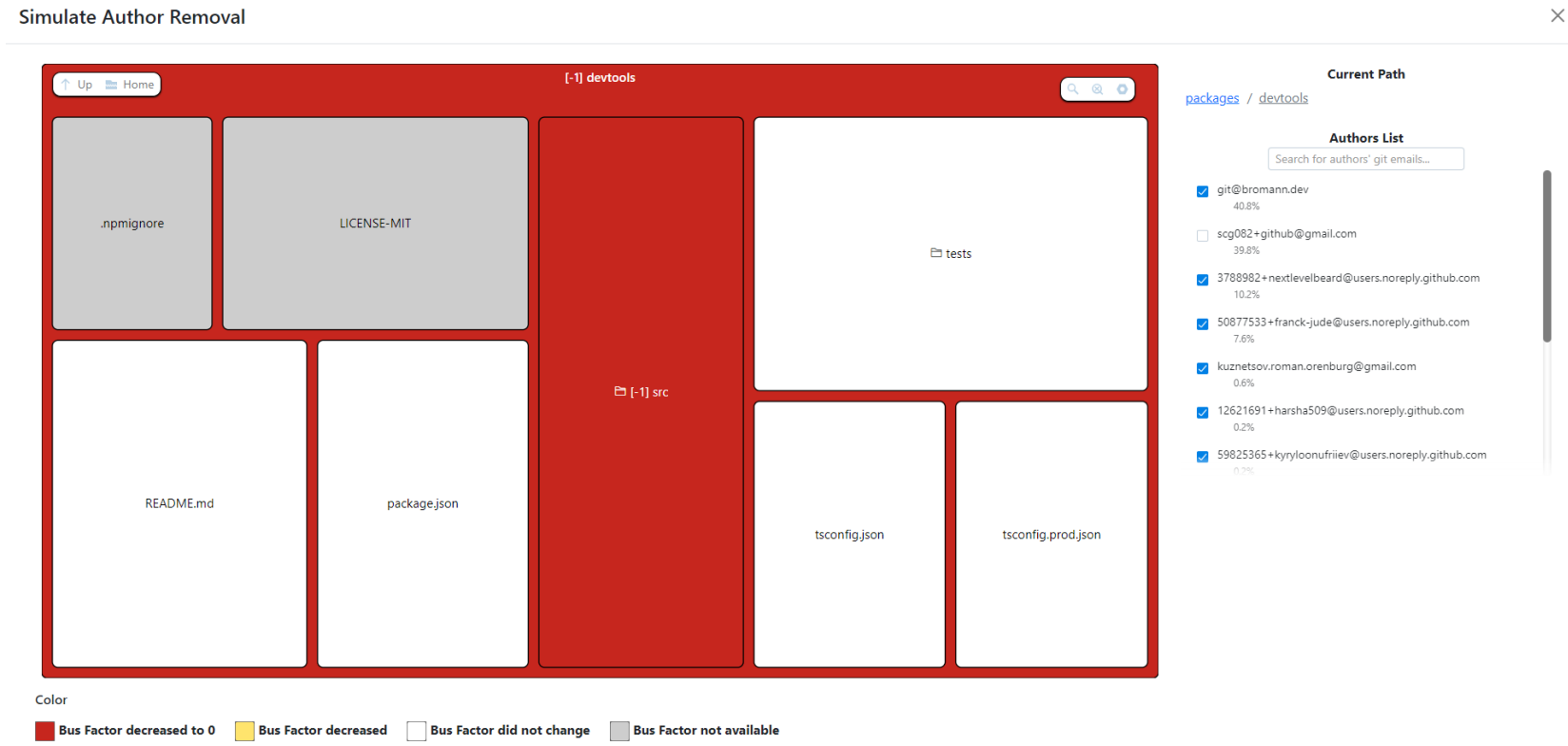

Figure 3.5: BFViz simulation mode panel

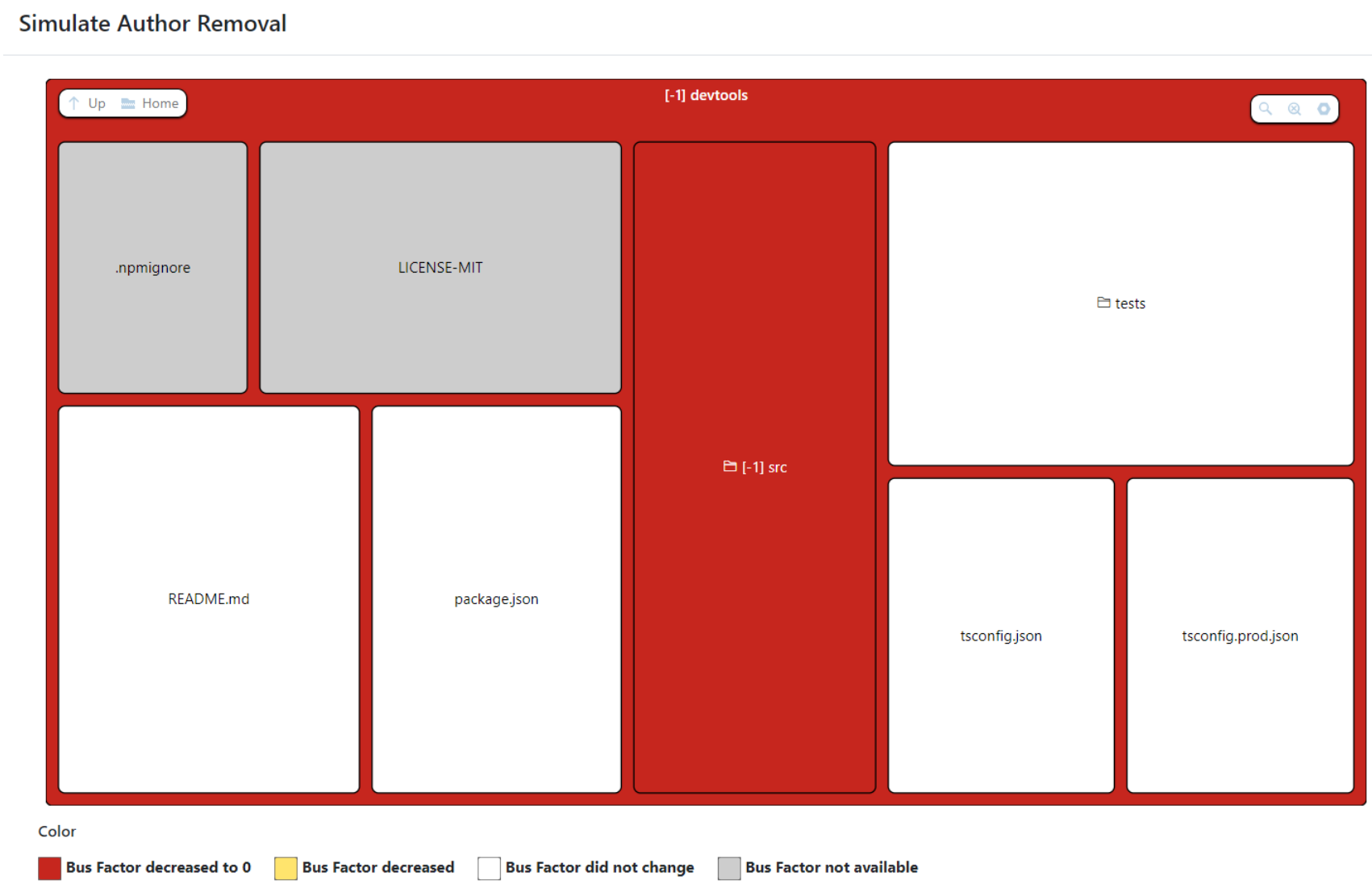

Figure 3.6: BFViz simulation mode treemap closeup

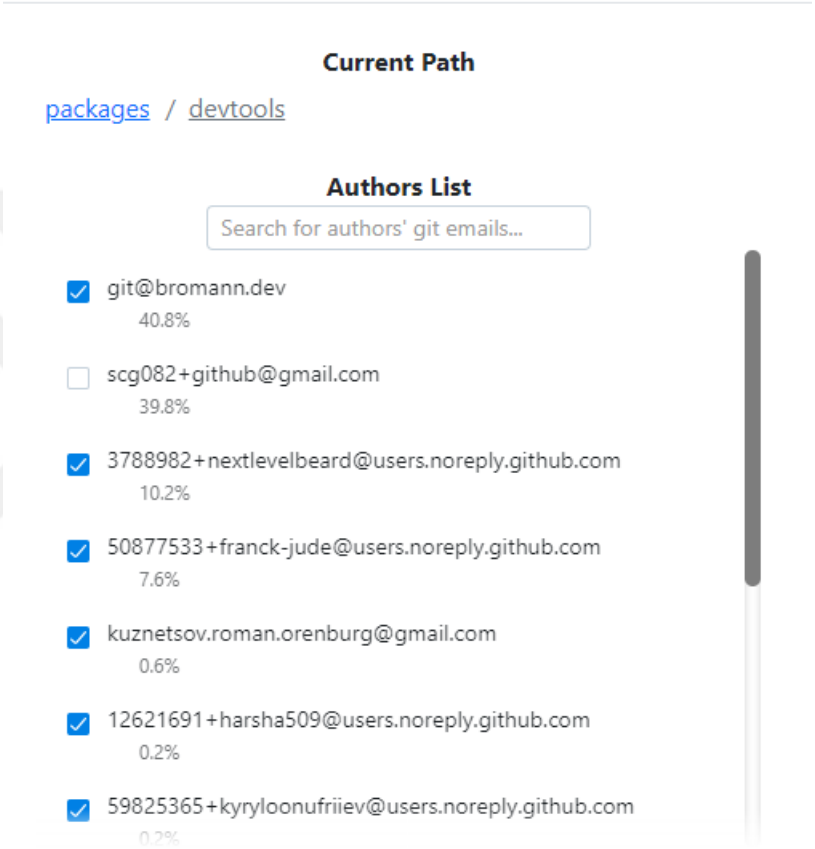

Figure 3.7: BFViz simulation mode contributors closeup

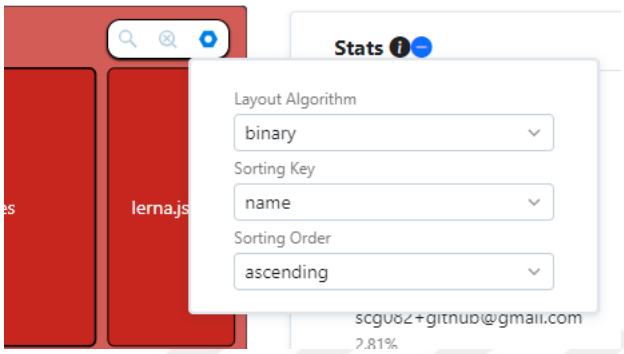

Figure 3.8: BFViz zoom and configuration island

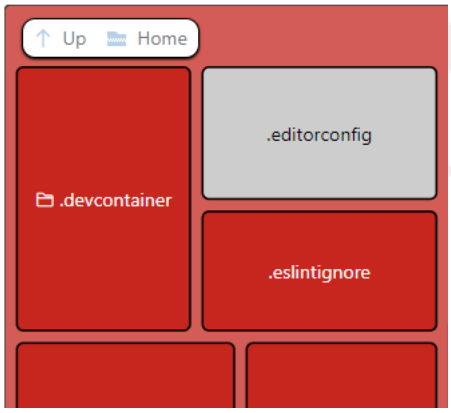

Figure 3.9: BFViz navigation island

and emails for common substrings that indicate the user is a bot. These include *'dependabot'*, *noreply@github.com*, and the regular expression pattern '*bbot\b*' being matched in the name. After bot filtering, merging duplicate users is performed by trimming and splitting names into first, penultimate, and last names and then cross-comparing their Levenshtein distances [23]. It is also important to note that the bus factor data for BFViz is currently only calculated over the past one-and-a-half years to circumvent long run times.

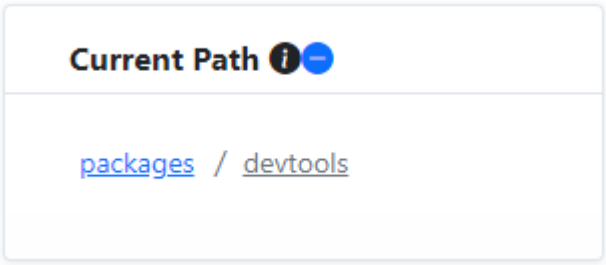

Figure 3.10: BFViz project navigation panel

The formula we use is as follows:

$$DOA_i = 3 \cdot FA + \sum_j DL_i^j + 2.4 \log(1 + \sum_k \sum_j DL_k^j) -$$
$$- 2.4 \log(1 + \sum_{k \neq i} \sum_j DL_k^j),$$
$$FA = \exp(\frac{-t_{FA}}{S}), DL_i^j = \exp(\frac{-t_{DL^j}}{S}),$$
$$DOA_i^{norm} = \frac{DOA_i}{max_i(DOA_i)}$$

The DOA_i represents the contribution of an engineer to a specific file. FA determines whether the engineer created the file and is zero for non-creators. DL_i^j is the number of commits made by an engineer i to the file j. The knowledge contribution from all parts of the equation decays exponentially according to the number of days that have passed since the contribution was made. In line with the original work, we set inverse decay speed $S = 90$, which means that the knowledge from a contribution halves in just over two months. An engineer is said to be a major contributor to a file if the $DOA \geq 0.001$. With the normalized DOA^{norm} version of the algorithm, the condition becomes $DOA^{norm} \geq 0.75$. We use the normalized DOA version of the algorithm.

The algorithm to obtain the bus factor from the DOA data remains the same as defined by Avelino et al. [2] and utilized by Jabrayilzade et al. [4]. With the DOA data for all of the files in the project, we start moving the authors with the highest DOA scores in the target scope one by one to a separate list. This list is called the *key developers* list. The target scope can be the entire project or a

subdirectory in the project. We stop as soon as the percentage of the children files of the target scope covered by the remaining authors goes below 50%. This 50% threshold is used as-is from the Avelino [2] and Jabrayilzade [4] algorithms. At this point, the bus factor of the target project is the number of developers in the *key developers* list. If any of these developers were to leave, it would decrease the bus factor of the target scope.

This algorithm is called every time the target scope changes. That generally happens as the user navigates around the project in the application. Every time the user navigates to a folder, BFViz calls this algorithm to calculate the bus factor for it. Additionally, the simulation mode also utilizes this algorithm, with one modification. It simulates departure by filtering out the targeted developers from the key developers' list if they are present in it. If they are part of the set of key developers, it means they are contributing to the bus factor. Removing them reduces the bus factor where applicable and the user is shown this difference. If they are not on the list, then it means they were not part of the set of contributors making up the bus factor in the first place. As such no change will be reported in the bus factor if they depart.

3.3.2 Visualization

Our main accomplishment is the visualization of the hierarchy of a project in a treemap form. This is performed by using React[2] for frontend components and D3.js[3] for visualizing the treemap itself.

Our main candidates for visualization were circle-packing and treemaps. Our target was a visualization that could be used to visualize nested hierarchies, similar to directory structures on filesystems. We also wanted the visualization method where we could employ size to convey the file or folder's size, as well as color to reflect the category of the bus factor. Both treemaps [24] and circle-packing [25] are well-suited to hierarchical, and specifically filesystem data. Our

[2]https://react.dev/
[3]https://d3js.org/

decision to proceed with treemaps eventually also allowed us to embed our navigation system into the visualization itself. This would not have been possible with visualization types, such as bar charts and pie charts, that other works we have mentioned use. One major factor behind our decision to use treemaps was that, by intuition and subsequent observation, their rectangular tiles utilize rectangular screen space more efficiently compared to circular nodes in circle-packing [24]. Another reason to choose treemaps over circle-packing was our decision to only display immediate children of folders instead of the whole project. We wanted the visualization to contain minimal clutter. Despite this, we had trouble with treemaps using linear mapping of sizes between tiles and their origin files and folders. This issue was especially prominent when the variance in file and folder sizes was high: tiles from smaller folders and files tended to disappear. For this reason, instead of linearly mapping file and folder size to the tile sizes on the treemap, we tried several non-linear functions. We tried multiple variations of *square root*, and *log* functions with different parameters and additional terms to enforce minimum tile size. Eventually, by trial and error, we settled on $tileSize = log_2(fileSize)$. The $tileSize$ parameter itself is an input to the D3.js layout algorithms which then use them to lay out the tiles on the treemap.

The colors of the tiles were chosen initially based on JetBrains brand color schemes after lots of revisions. Eventually, we decided to use the globally utilized convention of red, yellow, and green. Red signals imminent danger, such as the bus factor being in the *Dangerous* range. Yellow signals a warning, indicating that the bus factor is low enough to warrant concern. Green signals that the bus factor is safely high. We foresaw problems in visibility for those with colorblindness and other visual difficulties so we added a custom color picker to let the users choose colors at their convenience.

A similar logic applies to a range of bus factor categories as well: Software projects can be of many sizes and we surmised that a fixed range would not be suitable for projects of all sizes. This is why we provided a slider to let the user configure the range boundaries for the categories. The slider is limited to a maximum bus factor of 20.

Chapter 4

User Evaluation

In this study, we defined realistic usage scenarios of BFViz and evaluated them using a list of tasks, in Table 4.5 that target BFViz's features with those scenarios. We also performed semi-structured interviews, in Table 4.6, to get more feedback from participants about BFViz. The last component of the evaluation was a feature ranking.

We chose four projects for our evaluation case studies and enlisted four users from the three projects to evaluate BFViz. They are shown in Table 4.3.

4.1 Methodology Overview

There are four main steps in our evaluation. The first step is the recruitment and shortlisting of participants. The second is to shortlist eligible projects provided by the shortlisted participants. The third is to carry out the evaluation. The fourth is to collate results. First is a list of tasks for the participant to perform. Next, we held a semi-structured interview about their experience. We end with a short survey asking them to rank the features of the application based on their experience. Participants were encouraged to think aloud for the entire duration of the study session.

No.	Job Title
1	CIO / CEO / CTO
2	Tester / QA Engineer
3	DevOps Engineer / Infrastructure Developer / Site Reliability Engineer
4	Team Lead
5	Developer / Programmer/ Software Engineer
6	Architect
7	Data Analyst / Data Engineer / Data Scientist
8	Product / Project Manager
9	Tools Analyst

Table 4.1: Roles Targeted for Survey Respondents

While we did not follow an explicit guide on structuring our study, all three of the data collection methods we have employed appear frequently in works assessing new tools.

The entire study process from start to end is shown in Figure 4.1.

4.2 Participants' Recruitment

We recruited participants for the user evaluations in two ways: first was by using the list of users who had already signed up with JetBrains signaling their participation in surveys and test; The second was by inviting colleagues from other teams in JetBrains Research to evaluate the tool.

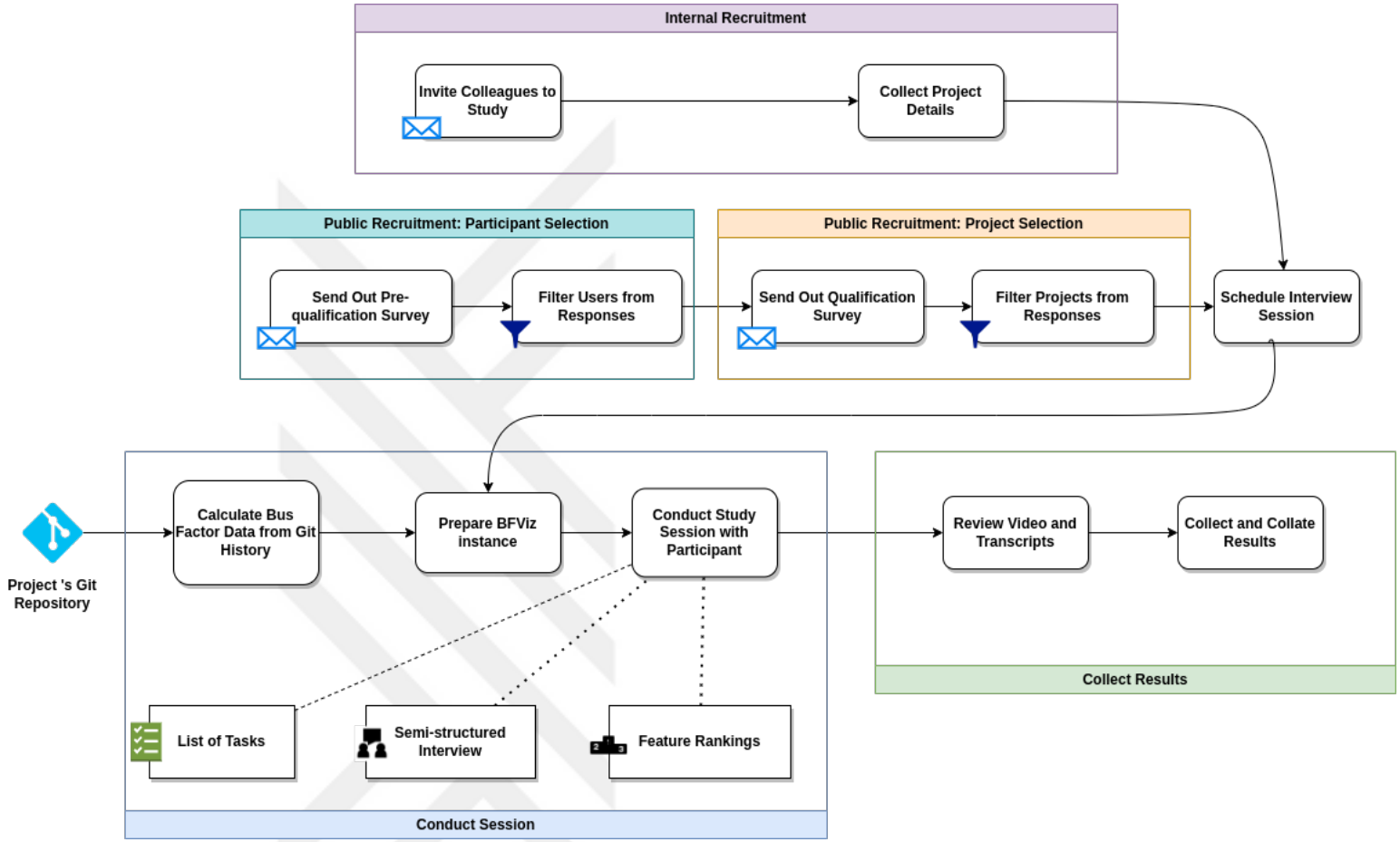

Figure 4.1: User Study Overview

No.	Question
1	In how many open-source projects did you actively participate in last year? Active participation here means that, on average, you did at least three commits and/or one code review monthly during the last six months.
2	Please provide the links for up to three open-source projects you have participated in most actively. If you are selected for the study, we will run our tool on some of the projects you have worked on and ask you to use our tool to analyze the code ownership of this project. We will make no changes to the code while running this tool, and will solely analyze the code metadata that can be mined from the Git repo.
3	Please write the alias (e.g. GitHub username), under which you have contributed to these projects
4	Do you have any kind of vision impairment?

Table 4.2: Qualification Survey

4.2.1 Public Recruitment

Before conducting evaluation interviews, we sent out pre-qualification and then qualification surveys to developers who had already registered with JetBrains for trials and case studies. The pre-qualification process was to filter out candidates who were not either in an upper management role, or a technical role having frequent interaction with code.

The pre-qualification roles list is shown in Table 4.1. From the pre-qualification survey, we filtered down our list of respondents to those who had five plus years of experience and were in one of the roles that we were targeting. They were then invited for the qualification survey. The qualification survey is shown in Table 4.2. From this survey's responses we chose GitHub projects with more than 15 contributors.

The result of the respondents who qualified and contacted us back upon being

No.	Name	Language(s)	Contributors	Stars	Forks	Source
1	*REDACTED*	Python	100	203	37	GitHub
2	*REDACTED*	Terraform, Go	106	186	114	GitHub
3	compose-multiplatform-core	Kotlin, Java, C++	N/A	325	894	JetBrains

Table 4.3: Study Projects

Participant ID	Project No.	Designation	Selection Process
A	1	Owner	Public Recruitment
B	2	Engineering Manager	Public Recruitment
C	3	Software Engineer	Internal Recruitment
D	3	Software Engineer	Internal Recruitment

Table 4.4: Participants and Project Map

informed of their eligibility was two candidates, Participants A and B.

4.2.2 Internal Recruitment

We had some candidates from our public recruitment drive who we were not able to establish contact with after they had passed the eligibility criteria. The number of eligible projects from about 500 potential candidates we contacted was six which was quite low to start with. Out of these, four did not respond further. As a result, we decided to explore other avenues from where we could recruit participants. We contacted colleagues from other teams in JetBrains Research to

see if someone working on an internal project was also interested in evaluating the tool. Eventually, two colleagues C and D, working on the same project, signed up for and went ahead with the evaluation. The summary of participants and projects is shown in Table 4.3 and Table 4.4. The identities of participants and projects were anonymized or redacted wherever anonymity was promised or requested. For the compose-multiplatform-core project, we could not retrieve the number of contributors as GitHub does not have it included in GitHub Insights.

Project C, which the participants from this stream used BFViz on for the study, was an exceptional case. This project was a fork a popular open-source projects. We were informed that the team only works on targeted portions of this project and then pulls the changes back upstream. As such, they were shown a targeted version of their project in BFViz, with parts of the project that they do not interact with, removed, on their request. Both participants C and D were shown the same version though, there was no difference between them.

4.3 Tasks

Each task is designed to lead the participant to interact with one of the features implemented in BFViz. We aim to evaluate the five main features of our tool: 1) File/folder BF Visualization, 2) Navigation, 3) Filtering, 4) Contributors' List, and 5) Simulation. Some of the features have more than one task in our usability test. In total, we have 11 tasks. Each task has a success criteria. If the participant manages to achieve the success criteria in three minutes, we consider that task as completed. The list of tasks is designed to cover all three of our research questions, examining feature-wise and overall usefulness of BFViz to inspect contribution and bus factor data about a project, as well as its ease of use.

4.4 Semi-structured interviews

We perform a semi-structured interview to cover points on the usefulness of the app (RQ3) and its features (RQ1). The usability of the app and its features can be evaluated according to the success rate of participants in performing requested tasks. We also collect the task completion time. Also, by reviewing the videos, we collect the feedback received from participants while performing the tasks. We record the answers to our usability-related questions during the semi-structured interview. The responses to these questions are collected by reviewing the videos and are then presented in a summarized and anonymized form in the results. The questions are listed in Table 4.6.

4.5 Short questionnaire

We conclude by asking the participant to answer a short questionnaire, ranking the features of the tool according to their usefulness and experience. It is designed to evaluate the usefulness of specific features of BFViz (RQ1). The features we ask the users to rank are:

- Bus Factor Visualization of Files and Folders
- Navigation
- Filtering
- Contributors' List
- Bus Factor Simulation

No.	Task
1	Please go to the following website *URL to BFViz deployment for their project* and describe what you see. Can you recognize the tiles that you see? Please use the zoom feature if the tiles are not visible.
2	Can you identify the folder that has the largest size (in bytes) in the current folder?
3	Can you find the bus factor of folder **X**? (where **X** is project-specific)
4	What is the bus factor of the whole project?
5	Please modify the bus factor ranges in a way that fits your project.
6	Please navigate to the folder that has a bus factor less than 2, then navigate back to the root of the project.
7	Please navigate to the folder with the code you understand and worked on and name its title.
8	Please find out who are the top five contributors to this folder. What do the numbers listed for each of these contributors mean?
9	Using the tool, please filter out all files with file extension **Y**? (where **Y** is project-specific)
10	Using the tool, please filter out all files and folders with a specific prefix of your choice (such as '.' or 'data').
11	Please try to find how the bus factor of the project changes, if the two top contributors to this folder leave the project.

Table 4.5: List of Tasks

No.	Question	RQ
1	Which benefits do you personally see in having the ability to visualize bus factor compared to plain value reporting?	RQ3
2	What do you like/dislike about visualizing the bus factor of each file and folder in the project?	RQ1 and RQ3
3	Do you think that the ability to modify the BusFactor ranges used in BFViz to indicate Dangerous, Low, and OK is helpful to visualize the bus factor of your project?	RQ1 and RQ3
4	What do you think about the default color scheme used for showing the bus factor values in BFViz?	RQ2
5	What do you like about simulating the impact of a top contributor leaving the project?	RQ1
6	Which benefits do you personally see in showing the top contributors for an artifact?	RQ1
7	How would you evaluate your overall experience of using the tool?	RQ2
8	Which benefits do you personally see in filtering schemes for the artifacts?	RQ1
9	Is it easy to navigate between different folders with the tool?	RQ2
10	Is the information provided for each contributor helpful for you?	RQ2
11	In your opinion, is using regular expressions for filtering a good decision? What are the alternatives?	RQ2

Table 4.6: List of interview questions

Chapter 5

Results

The results of the user evaluation study are presented in this section.

To reiterate, the study was across four participants, two with owners of open source GitHub projects that were chosen via a targeted search, and two with colleagues involved in JetBrains projects. Projects 1 and 2 from Table 4.3 are the open-source GitHub ones, while the third project is a project belonging to JetBrains. Each study consisted of three stages, a list of tasks for the user to perform, a semi-structured interview, and a feature-ranking matrix.

We got all four participants to carry out the tasks outlined in Table 4.5. The results of the first part of the evaluation, the list of tasks to perform, for all four projects are shown in Table 5.1. The results for the second part of the evaluation, the semi-structured interview, are shown in Tables 5.2, 5.3, 5.4, and 5.5. The results for the third part of the evaluation, the feature ranking, are shown in Table 5.7. These results are detailed in the sections that follow.

5.1 Tasks

This section presents the results for the first part of the evaluation, which is the list of 11 tasks that each participant performed. The results are shown in Table 5.1. Out of all the 44 task attempts, four instances each of the 11 tasks that make up our list, 35 resulted in successful completion, marked by a '✓' in the table. Nine task attempts failed, marked by a '✗' in the table with a red background. There is no overlap between the failed tasks except that of Participants C and D both of whom failed tasks 9 and 10. Out of the four participants, only participant *A* managed to complete all 11 tasks. The rest of the participants completed nine, eight, and seven tasks, respectively.

As for the success rate of individual tasks, only four tasks, numbered 1, 2, 4, and 7 managed a 100% completion rate. Tasks 3, 5, 6, 8, and 11 managed a 75% completion rate. Tasks 9 and 10 had a 50% completion rate, with only two participants able to complete them.

The overall success rate across all task attempts was 79.6%.

No	Task	*A*	*B*	*C*	*D*	Pass%
1	Please go to the following website (link to be created) and describe what you see. Can you recognize the tiles that you see? Please use the zoom feature if the tiles are not visible.	✓	✓	✓	✓	**100%**
2	Can you identify the folder that has the largest size (in bytes) in the current folder?	✓	✓	✓	✓	**100%**
3	Can you find the bus factor of folder **X**? (where **X** is project-specific)?	✓	✓	✗	✓	**75%**
4	What is the bus factor of the whole project?	✓	✓	✓	✓	**100%**
5	Please modify the Bus Factor ranges in a way that fits your project.	✓	✓	✓	✗	**75%**
6	Please navigate to the folder that has the bus factor less than 2, then navigate back to the root of the project.	✓	✓	✓	✗	**75%**
7	Please navigate to the folder with the code you understand and worked on and name its title	✓	✓	✓	✓	**100%**
8	Please find out who are the top 5 contributors to this folder. What do the numbers listed for each of these contributors mean?	✓	✗	✓	✓	**75%**
9	Using the tool, please filter out all files with file extension **Y**?	✓	✓	✗	✗	**50%**
10	Using the tool, please filter out all files and folders with a specific prefix of your choice (such as '.' or 'data').	✓	✓	✗	✗	**50%**
11	Please try to find, how the bus factor of the project changes, if the two top contributors to this folder leave the project	✓	✗	✓	✓	**75%**
	Per Participant Task Pass %	**100%**	**81.8%**	**72.7%**	**63.6%**	**79.6%**

Table 5.1: Task completion results for all four participants

5.2 Semi-structured Interview

This section presents the results of the second part of the evaluation, which is the semi-structured interview. The participants responded to 11 questions targeting different features and research questions.

We have included anonymized and summarized versions rather than verbatim versions of the participants' responses. Anonymization was done by removing any references to their project or team members where names were explicitly mentioned. Summarization entailed shortening run-on sentences, collating points that were mentioned in response to different questions, and gathering relevant comments made elsewhere to the related questions.

Tables 5.2, 5.3, 5.4, 5.5 present the responses of participants *A*, *B*, *C*, and *D* respectively. Table 5.6 presents a list of themes and observations that the participants made. This table is sorted in descending order by the number of participants who made the observations. The observations are a mix of features that the participants explicitly pointed out were useful, features that they had issues with, and additional features that they wanted to see implemented. The table also links the observations back to the research questions that they are related to.

Observations that all the participants made were that they preferred bus factor data visualized rather than as text reports, that the color scheme and navigation style are suitable for the information being conveyed, and that both filtering and tile-sizing need improvement. Three out of four participants also pointed out that the Contributors' List was a useful feature. At the same time, we had three participants who pointed out that the contributors list had a problem in some cases where the percentages were cut off.

No.	Question	Answer
1	Which benefits do you personally see in having the ability to visualize bus factor compared to plain value reporting ("listing numerical bus factor values)? (RQ 3)	There is value in raw data for further analysis or to integrate it with other systems or implement monitoring for it. I could try and implement visualization in Excel but it will take me a long time, or I could just use this tool and all of the data I need will be there. If you give users the ability to create their own analyses with the data, it would really complete the application.
2	What do you like/dislike about visualizing the bus factor of each file and folder in the project? (RQ 1 & 3)	There are always some folders that are not part of the core code or are automatically generated. The bus factor would not really apply to them since committing them does not mean that the author is suddenly knowledgeable about the project for it to be included in the calculation of the project's overall bus factor. Having the bus factor separately for files and folders eliminates this problem.
3	Do you think that the ability to modify the BusFactor ranges used in BFViz to indicate Dangerous, Low, and OK is helpful in visualizing the bus factor of your project? (RQ 1 & 3)	I think it would be really useful on projects with more contributors. As a manager, I would also like to highlight important areas with high activity or low activity to redistribute workloads.
4	What do you think about the default color scheme used for showing the bus factor values in BFViz? (RQ 2)	Green, yellow, red, I think is international, it makes perfect sense. In the beginning, I was a bit confused as to why everything was red, but I realized the threshold was set to two and the majority of the folders had a bus factor of one.
5	What do you like about simulating the impact of a top contributor leaving the project? (RQ 1)	Not my first choice in terms of features but that might be because I'm the main contributor to my project
6	Which benefits do you personally see in showing the top contributors for an artifact? (RQ 1)	I think it is useful in a variety of cases. For example, I would like to try it on the Celery project. I am not involved in the code of that project beyond a few lines and issue reporting, but I would still like to check out areas with a lot of contributors to see what is going on.

Table 5.2 – *Continued on the next page*

No.	Question	Answer
7	How would you evaluate your overall experience of using the tool? (RQ 2)	It is valuable for me but I also know most of the project already. It should be even more useful for other projects. Some issues to solve are that the root link is missing from the current path and is separate on the navigation island. The color panel is not the first thing in focus, which I think it should be. It would be valuable to have some sort of pop-up UI tutorial. Tile sizes should have been configurable. The value of the size is not apparent. It would be nice to have it sorted by the number of commits. The size panel is a bit messy.
8	Which benefits do you personally see in filtering schemes for the artifacts? (RQ 1)	Filtering, I would like to see implemented better, I think the suggested filters option would be helpful if it was fixed to not include folders. I would also like folders-only and a no-folders option in the filtering as well.
9	In your opinion, is using regular expressions for filtering a good decision? What are the alternatives? (RQ 2)	No, wildcards would have been better because they are the standard for filtering files and directories. Regular expressions are too complex and unfamiliar for most users. Alternatives could be using checkboxes, dropdown menus, or sliders to filter by different criteria.
10	Is it easy to navigate between different folders with the tool? (RQ 2)	I think overall it was pretty easy. I think I instinctively got it. I have an issue with the root folder not appearing in the clickable path. It is separate on the navigation island. I would also like to have the simulation mode on the same screen instead of being on a separate modal.
11	Is the information provided for each contributor helpful to you? (RQ 1)	Yes but again, in this project, I am the main contributor, so I already know this information.

Table 5.2: Questions and answers from the user evaluation interview with Participant *A* for Project *1*

No.	Question	Answer
1	Which benefits do you personally see in having the ability to visualize bus factor compared to plain value reporting ("listing numerical bus factor values)? (RQ 3)	As a manager, I need to understand what the 'health' of my projects look like. This tool visualizes it from one aspect. There should be some measure for the similarity in the different parts of the code as well. Some similarity index could tell us who could work on certain sections of the code most effectively, given how similar the code they have already worked on is to the target code. Based on front-end/back-end, based on language. Computing the bus factor from VCS is prone to having biases that are not recognized. We cannot have a bias-free system but it would be nice to be informed of the biases. It would be nice to have the exact formula that is being used. It would also be nice to have different definitions of the bus factor, and let people choose the version of the bus factor algorithm that they want to use.
2	What do you like/dislike about visualizing the bus factor of each file and folder in the project? (RQ 1 & 3)	It is definitely helpful to have information at this granularity level. It would be nice to have something that ranks file importance, interactions, and dependencies as well. Not all files are equal. I also have some reservations about the bus factor computed, the numbers appear to be on the lower side, I have a team of 5-6 people and they are all active contributors yet the bus factor shown is around 2-3.
3	Do you think that the ability to modify the BusFactor ranges used in BFViz to indicate Dangerous, Low, and OK is helpful in visualizing the bus factor of your project? (RQ 1 & 3)	With the range, I can adjust the range boundaries individually in steps but the actual situation is much more complex. These situations are much better visualized with a continuous gradient or a spectrum to mirror the on-ground reality. Having this segmentation is fine for a prototype and I was able to understand it but I would prefer having a spectrum.

Table 5.3 – *Continued on the next page*

No.	Question	Answer
4	What do you think about the default color scheme used for showing the bus factor values in BFViz? (RQ 2)	The colors are nice since red immediately catches my attention while yellow does not. The sizes assigned to the tiles seem to not be on a linear scale and it appears to be misleading at first glance. I was not sure about the units for the colors though. I can see the numbers. The colors do their job, they are not too aggressive, but I would like to have the scale and units.
5	What do you like about simulating the impact of a top contributor leaving the project? (RQ 1)	This is the killer feature. As an engineering manager, this is a real situation I have to deal with: When someone is leaving, which parts of the project will be impacted? What are the parts of the project that I don't know? What parts should I organize knowledge transfer for? This is the real seller for me. The interface could have been better. An idea I have is where I should direct new resources to best mitigate the bus factor, let's say an incoming intern: Where should I place them to improve the health of the project?
6	Which benefits do you personally see in showing the top contributors for an artifact? (RQ 1)	It seems ok but one thing that you might have missed is some parts of the code are generated. For example, we have 'cassettes' that mock interactions of a server being provisioned. Now these are not code contributions, but they are huge and the user who is merging it does not become knowledgeable about the project according to your metric just by merging it because it is not really code.

Table 5.3 – *Continued on the next page*

No.	Question	Answer
7	How would you evaluate your overall experience of using the tool? (RQ 2)	Code is shared knowledge and it is important for me as an engineering manager to understand exactly which parts of the project will be affected if someone leaves. It would be nice if it could visualize and give statistics over multiple projects with variable project weightage or have this integrated into an IDE. It would be nice to have periodic exports. I would like to see dependencies in the different parts of the code. Highlighting the criticality of the code would be helpful. For this, we could need to analyze the overall AST of the project. We could also remove folders that are noisy, the ones that are auto-generated, like docs, using regular expressions. I have a small screen so I had trouble viewing the components. I could not see the list of contributors. But when I zoom out, it is visible. Having a tree view that shows the full names of the files would be nice. I can see the tooltip but it is not enough. This is a Go project but Java filenames are even worse. It is on the right path, it will have biases but I would like to see transparency in how and what it is doing, so I can consider those biases. Some code-level insight would be great, such as refactoring suggestions, module merging suggestions, and so on. I would also like to see this integrated into GitHub or GitLab. In GitHub, the Insights tab could be a good destination for it. You can have it in the IDE too. I would not use a different platform to consume this information. However since it is based on Git data, it should be close to the source to avoid security concerns. For open-source projects, it is nice but internal closed-source projects are where it will shine. It's a nice experience and it solves a real problem that we face.
8	Which benefits do you personally see in filtering schemes for the artifacts? (RQ 1)	It was nice. I like that you provided regular expressions. I like that you provided a list of suggestions. I really think that is a huge pro. It allows me to navigate quickly and nicely. It is the core barebone feature that I need. The rest are cosmetic.

Table 5.3 – *Continued on the next page*

No.	Question	Answer
9	In your opinion, is using regular expressions for filtering a good decision? What are the alternatives? (RQ 2)	Definitely a nice feature to have. It should be added to the simulation mode as well.
10	Is it easy to navigate between different folders with the tool? (RQ 2)	The Up and Home buttons could have been in the same panel as the current path. Having a tree view to mitigate the short names problem would be good to have.
11	Is the information provided for each contributor helpful to you? (RQ 1)	Part of the contributor list is also hidden and we don't have horizontal resize to expand the panel and see the full name and information it is providing. I could not see the full names or numbers sometimes.

Table 5.3: Questions and answers from the study interview with Participant *B* for Project *2*

No.	Question	Answer
1	Which benefits do you personally see in having the ability to visualize bus factor compared to plain value reporting ("listing numerical bus factor values)? (RQ 3)	In general, it's useful. The colors really stand out and grab my attention, they direct me to the problematic parts marked in red and that makes me wonder what's the situation there and how to improve it.
2	What do you like/dislike about visualizing the bus factor of each file and folder in the project? (RQ 1 & 3)	In general, it's nice, we need it, like I said it attracts my attention to certain areas of the project. For our project, the sizes of the tiles seem more or less equal which is not very intuitive and does not provide information about the size. Perhaps it will not be like this for other projects.
3	Do you think that the ability to modify the BusFactor ranges used in BFViz to indicate Dangerous, Low, and OK is helpful in visualizing the bus factor of your project? (RQ 1 & 3)	Customization is helpful but I'm not sure what the appropriate range would be for my team and this project which is why I think maybe I didn't use it too effectively.
4	What do you think about the default color scheme used for showing the bus factor values in BFViz? (RQ 2)	The colors seem appropriate and draw attention where they should.
5	What do you like about simulating the impact of a top contributor leaving the project? (RQ 1)	The feature button was a bit hard to find even though I was expecting a feature after the pre-evaluation briefing. Also in some places, it seems like I am at the top of the contributors' list, and removing me leads to a decrease in the bus factor even though I was not expecting myself to be so valuable. I would also recommend adding a 'Clear All' button here.
6	Which benefits do you personally see in showing the top contributors for an artifact? (RQ 1)	This one is nice and gives the whole picture but I have a couple of questions: I see a colleague with their GitHub-generated email in the list. I know that they use their company email mainly so how are we handling multiple IDs for the same people? Are we merging them in the data?

Table 5.4 – *Continued on the next page*

No.	Question	Answer
7	How would you evaluate your overall experience of using the tool? (RQ 2)	Overall it's great but I have some comments about the contributors list. Color coding contributors, that could be nice too according to some criteria. If we could also have a number showing the total number of contributors, that would be nice.
8	Which benefits do you personally see in filtering schemes for the artifacts? (RQ 1)	This should be helpful but I have some trouble seeing the files. Do we need to use filtering to see the files? I had some trouble with it but I'm not sure if it was my fault or a bug.
9	In your opinion, is using regular expressions for filtering a good decision? What are the alternatives? (RQ 2)	It wasn't completely clear how this worked because some of my filtering expressions worked, other times I was expecting to see some files with certain filters that were not shown. It's possible that my expressions were wrong. For the effectiveness of the regular expressions, I often forget filenames, I mostly use the UI or the search features to open or search for files. I'm also not sure why we would need to search for specific files in this tool.
10	Is it easy to navigate between different folders with the tool? (RQ 2)	I think navigation through the folders is important and what we want to see. For example for the UI models part of our project, we have a lot of folders here, and it's really needed.

Table 5.4 – *Continued on the next page*

No.	Question	Answer
11	Is the information provided for each contributor helpful to you? (RQ 1)	It is helpful to see the contribution percentage but for low bus factor items, could we also get suggestions on how it could be improved, and how we could increase the contribution percentage of people lower in the list? I understand that it might not always be possible to get a new hire and then train them to balance out the bus factor, but could we get suggestions on improving it with the existing team members? I should be able to set the baseline set of contributors for a repository like this which is a fork and has some external contributors. Many of them have contributions but are not part of the team, so I am unsure if they should be factored in bus factor calculations. Perhaps allowing the end-user to set some rules about who should be in and who should be out might help. The panel could have a horizontal slider to make it easier to see everything (some lines are cut off). Also, a search option for specific contributors here would be nice too.

Table 5.4: Questions and answers from the user evaluation interview with Participant *C* for Project *3*

No.	Question	Answer
1	Which benefits do you personally see in having the ability to visualize bus factor compared to plain value reporting ("listing numerical bus factor values)? (RQ 3)	It has a useful purpose. If the contents being shown are refined and more targeted with unnecessary folders and files filtered out, it will be really helpful to pinpoint where bus factor issues are in the project and help our management processes. A report would just show the folders with the most and least bus factors but it will not let me understand the reasoning behind it. The tool allows me to investigate and understand the actual situation.
2	What do you like/dislike about visualizing the bus factor of each file and folder in the project? (RQ 1 & 3)	It is hard to compare sizes because the sizes are almost the same for folders when there are many. I would perhaps like to see the sizes reported as kilobytes or megabytes, as numbers.
3	Do you think that the ability to modify the Bus Factor ranges used in BFViz to indicate Dangerous, Low, and OK helps visualize the bus factor of your project? (RQ 1 & 3)	I think it would be better to have two fields here than to have a slider to change the range of the bus factor colors. The slider was not too usable especially when the upper and lower limits were close to each other.
4	What do you think about the default color scheme used for showing the bus factor values in BFViz? (RQ 2)	They were ok, I could tell which color represented what category of bus factor values, especially the red.

Table 5.5 – *Continued on the next page*

No.	Question	Answer
5	What do you like about simulating the impact of a top contributor leaving the project? (RQ 1)	It is very helpful, especially in pointing out which folders will be 'lost' if someone leaves the team but there is a problem in how the impact is communicated in terms of the size of the folder. Sizes are not being reported in linear proportion and it could make the user worry about a folder with a dropping bus factor which might not amount to much in size or importance but is highlighted very prominently by the application. Perhaps we could show the gradients or averages of the bus factors of the child contents of folders. Otherwise, on the surface, it might not be clear for parent folders what the situation is inside. We will have to manually navigate to the subfolder if there is no change in the parent folder but there are changes in the bus factors of the subfolders.
6	Which benefits do you personally see in showing the top contributors for an artifact? (RQ 1)	The experience was fine but some features perhaps need more work, I could not get them to work as I had hoped. Filtering is one of them. There are some other improvements as well that could be made as I have pointed out.
7	How would you evaluate your overall experience of using the tool? (RQ 2)	Overall it's great but I have some comments about the contributors list. Color coding contributors could be nice too according to some criteria. If we could also have a number showing the total number of contributors would be nice as well.
8	Which benefits do you personally see in filtering schemes for the artifacts? (RQ 1)	Having filtering is definitely needed for this kind of environment but it should be functioning properly.
9	In your opinion, is using regular expressions for filtering a good decision? What are the alternatives? (RQ 2)	It wasn't completely clear how this worked because some of my filtering expressions worked, other times I was expecting to see some files with certain filters that were not shown. It's possible that my expressions were wrong. For the effectiveness of the regular expressions, I often forget filenames, I mostly use the UI or the search features to open or search for files. I'm also not sure why we would need to search for specific files in this tool.

Table 5.5 – *Continued on the next page*

No.	Question	Answer
10	Is it easy to navigate between different folders with the tool? (RQ 2)	It was all right but I have an idea about this part. It would be better if we could also include a mode that shows the directory structure as a classical directory tree. This is also related to the size visualization as it is not clear enough.
11	Is the information provided for each contributor helpful to you? (RQ 1)	It is useful but the percentage of contribution is missing in some cases. It is outside of the view, not nice, it should be visible.

Table 5.5: Questions and answers from the user evaluation interview with Participant *D* for Project *3*

No.	Observation	RQs	Participants
1	Colors are satisfactory, they convey the purpose (Danger, Warning, OK).	2	A B C D
2	The size of the tiles does not reflect the actual size of the item.	2	A B C D
3	Filtering is a useful feature to have for this application.	1, 2	A B C D
4	Visualization of bus factor is better than text reports.	1	A B C D
5	Navigation is a useful feature to have for this application.	2	A C D
6	The Contributors List is a useful feature to measure individual contributions to the team.	1, 2, 3	A C D
7	I had trouble using BFViz's implementation of regular expressions for filtering	2	A C D
8	Regular expressions are useful for filtering files and folders in this application.	1	B C D
9	The Contributors List is not fully visible sometimes.	2	B C D
10	A feature request for a mechanism for users to exclude certain files and folders from bus factor analysis.	2, 3	A B
111	The navigation panel is separate from where the path is shown, they should have been together.	2	A B
12	The formula determining the tile size should be shown and be configurable.	2	A D
13	A feature request to also handle incoming contributors on top of leaving ones in the simulation mode to improve the bus factor (and the overall health) of the project.	1, 3	B C
14	A feature request to have a directory tree view to speed up navigation.	2	B D
15	Using a slider for the bus factor category ranges is not suitable.	2	B D
16	Bus factor simulation is a great feature.	1, 3	B D
17	The bus factor is higher than I expected it to be.	3	C D

Table 5.6 – *Continued on the next page*

No.	Observation	RQs	Participants
18	The bus factor is as expected.	3	A
19	A feature request to export bus factor data.	1, 3	A
20	A feature request to highlight code activity hotspots in projects.	3	A
21	Bus factor simulation is an OK feature.	1, 3	A
22	Regular expressions are not suitable for filtering, glob patterns or some other interface should be utilized for it.	2	A
23	Simulation mode should not be on a separate modal, it should be on the same screen.	2	A
24	A feature request for a pop-up intro tutorial when the user first navigates to BFViz.	2	A
25	A feature request to suggest the best team member to work on a certain section of the code, given the bus factor and contribution data.	3	B
26	The bus factor algorithm should be transparent for the user, they should be able to analyze the biases the algorithm may introduce.	2	B
27	The bus factor algorithm in use should be switchable with different algorithms.	2, 3	B
28	The bus factor reported is lower than I expected it to be.	3	B
30	A feature request for visualizing bus factor across multiple projects.	2,3	B
31	A feature request for highlighting dependencies in the code.	2, 3	B
32	A feature request for highlighting the criticality of code.	3	B
33	A feature request to indicate changes in the bus factor of sub-folders without having to explicitly navigate to them in bus factor simulation, especially when parent folders have no overall bus factor change.	2, 3	D

Table 5.6: Common themes and observations in the semi-structured interview responses across all participants

5.3 Feature Rankings

This section presents the results of the third and last part of the evaluation studies, the feature rankings. All four participants were asked to rate the five main features of the application based on importance and usefulness. Table 5.7 show how participants A, B, C, and D ranked the five features respectively.

Table 5.8 shows the mean rank given to each feature across all participants. The first four features have relatively close mean ranks, starting at 2.25, and moving with steps of 0.25 to 3.00. Filtering has a distinctively worse mean rank of 4.50.

Table 5.9 shows the pairwise ranking between pairs of features. The number in each $cell_{ij}$ with row i and column j indicates the number participates that ranked feature i better (i.e. lower) than feature j. From the *Sum* column in the table we observe that filtering was only ranked better than the other four features twice. The numbers for the other features are close, with Contributors' List being ranked better 11 times, File/Folder Bus Factor Visualization 10 times, Bus Factor Simulation Mode nine times, and Navigation eight times. These results match the mean ranks in Table 5.8.

Rank	**A**	**B**	**C**	**D**
1	Navigation	Bus Factor Simulation	File/Folder Visualization	Contributors' List
2	Contributors' List	File/Folder Visualization	Navigation	File/Folder Visualization
3	Bus Factor Simulation	Contributors' List	Contributors' List	Bus Factor Simulation
4	Filtering	Filtering	Bus Factor Simulation	Navigation
5	File/Folder Visualization	Navigation	Filtering	Filtering

Table 5.7: Features ranked by study participants

Feature	**Mean Rank**
Contributors' List	2.25
File/Folder Visualization	2.50
Bus Factor Simulation	2.75
Navigation	3.00
Filtering	4.50

Table 5.8: Mean rank assigned to features by participants

	File/Folder Visualization	Navigation	Filtering	Contributors List	Bus Factor Simulation	**Better Rank Sum**
File/Folder Visualization	0	3	3	2	2	**10**
Navigation	1	0	3	2	2	**8**
Filtering	1	1	0	0	0	**2**
Contributors' List	2	2	4	0	3	**11**
Bus Factor Simulation	2	2	4	1	0	**9**
Worse Rank Sum	**6**	**8**	**14**	**5**	**7**	**40**

Table 5.9: Pairwise comparison of feature rankings from all four participants

Chapter 6

Discussion

This chapter discusses the results and attempts to infer answers to the research questions. We reiterate our research questions here to make it easier for the reader to keep track of them.

RQ1 What specific features of the BFViz did users find useful to measure the risk and dependence of a team on individual team members?

RQ2 Are the features implemented in BFViz are easy to use?

RQ3 Do decision-makers find BFViz a helpful tool to measure the risks and dependence of a team on individual team members?

6.1 Usefulness of Individual Features (RQ1)

In this section, we discuss the results that provide insight into the performance and perception of individual features. We have divided this further into per-feature subsections to keep the discussion as comprehensive as possible. Any responses outside of the five main features are covered in the 'Others' subsection.

We mostly focus on the semi-structured interview responses and the ranking responses for this section.

6.1.1 Bus Factor Visualization

All four participants pointed out in their semi-structured interviews that they preferred their bus factor data in visual form rather than as text reports. Participant A pointed out in Table 5.2 that they could script their visualization with Microsoft Excel macros or similar tools with the data but it would be too time-consuming. They preferred having it. Participant D was of the view that text reports are hard to navigate naturally across directories. The reason they provided was that a reader cannot click to go to a subfolder and check what the bus factor situation is. Instead, they would have to scan a list of folders, find the right one, and then go through and understand the numbers.

Another aspect that all participants unanimously agreed upon was the color scheme. From Table 5.6, all of the participants unanimously agreed that the color scheme of red, yellow, and green was useful for the purpose. Participants A and B explicitly pointed out in Tables 5.2, and 5.3 that red grabs and directs their attention to the problematic parts of their project, the ones with low bus factors. From the feature rankings, even though the mean ranks of four of the five features are quite close to each other, bus factor visualization comes out second.

6.1.2 Navigation

The mode of navigation provided was singled out as a useful feature by three out of four participants in their semi-structured interview as per Table 5.6. In Table 5.2, participant A stated that they were able to "instinctively get it". In Table 5.4, participant C explained that it was necessary if the project had a lot of directories as was the case in their project. From the rankings in Table 5.8, Navigation is fourth in the mean rank list, with a mean rank of 3.00.

6.1.3 Filtering

Filtering had a mixed response from the participants. Participants agreed that filtering is a very important component of an application like this. What they had a problem with was the implementation.

Participant *A* suggested that we use wildcards and *glob* patterns, commonly used in *.gitignore* files [1] amongst other places. Participant *B* had the opposite reaction, they were very happy to have regular expressions as the filtering interface, describing it as the "...killer bare-bones feature that I need". They also suggested that this feature be incorporated into the simulation mode. Participants *C*, and *D* had trouble with the filtering. They could not complete their filtering-related tasks either as the filtering was behaving erratically for their project. Despite this, both of the participants agreed that regular expressions are a suitable interface for filtering, anecdotally citing their experience on their IDEs using them for this purpose.

Filtering received the worst mean rank out of the five features with a mean rank of 4.5 in Table 5.8. The trouble that participants *C*, and D had with their filtering tasks, and the preference of participant *A* for glob patterns, both serve as evidence of this low mean rank.

6.1.4 Contributors List

The Contributors List was the feature with the best mean rank from the feature rankings in Table 5.8, with a mean rank of 2.25. Participants *A*, *C* and *D* stated that the Contributors List was a useful feature to have in Table 5.6. Participants *C* and *D* also highlighted that from the Contributors List, they could already spot their project would be in trouble if their project lead left. Participant *B* had some trouble on account of their small screen size and could not access the contributors list properly, so they were not able to provide any comment about

[1] `https://git-scm.com/docs/gitignore`

this.

6.1.5 Bus Factor Simulation

Bus Factor Simulation ranks third in the mean rank list shown in Table 5.8. Participant *A* rated the bus factor as OK. One major reason that they quoted for this was that as the project owner, they were by far the most active contributor to the project that we evaluated BFViz on. As such, they were not very interested because other contributors leaving would have minimal impact. On the other hand, participant *B*, the engineering manager for the project that we evaluated BFViz on for them, called it the "killer feature" and the "real seller" for "real situations" they have to deal with. They also ranked it as the most useful feature. Participant *C* also showed interest in the bus factor simulation mode but to the extent of being surprised to find themselves higher than they expected to. Participated *D* also found the bus factor simulation folder to be a useful feature, specifically pointing out the ability to pinpoint which folders will be 'lost' if someone left the team. They ranked it as the third most useful feature.

It is of note here that the most interest shown in this feature was shown by the participant working as an engineering manager. The appeal of this feature to managers is something that we had informally hypothesized while developing the idea behind this project. It is also interesting to note that participants *C* and *D* were less enamored with this feature, possibly due to having little to no managerial role in their teams. Participant *A* may be classified as an exceptional case, since they have effectively been the sole maintainer of the project analyzed for the past two years. They had 600+ commits to the project in total, all of them coming over the last two years, while the next two largest contributors had around 400 and 100 commits respectively, with almost none of them being over the past one-and-a-half years.

6.2 Usability of Features (RQ2)

6.2.1 Bus Factor Visualization

From the tasks in Table 5.1, we can see that the first six questions are involved with bus factor visualization. Across four participants who performed six related tasks each, there are a total of three failed task instances. Participant C could not get the bus factor of the specified folder for Task 3 under the stipulated time limit. They were able to do so for the next task, however. Participant D had trouble with the bus factor ranges goal in Task 5. In Task 6, our observation is that they misread the question because they navigated freely but pointed out a folder with a bus factor value of one instead of two, also saying it out loud.

With 21 bus factor visualization-related task instances being completed out of 24, we had a completion percentage of 87.5%.

One aspect that was problematic about the visualization was the tile sizes. All of the participants complained that the sizes and proportions of the tiles representing files and folders often did not line up with their real size. The formula that we used to file and folder size to tile size was designed to minimize extreme differences in tile sizes when there are significant differences between sizes at the same directory level. Unfortunately, it seems like this normalization process affects the user's ability to perceive differences between file sizes on the visualization. On observation, this is more prominent when the file sizes at a given directory level have similar orders of magnitude of file sizes.

Participants B and D also noted that it was not clear what the bus factor range sliders were for at first glance. They suggested that the slider might not be the best interface for selecting the ranges and that we should explore other options such as two number input fields.

6.2.2 Navigation

In general, participants managed to use the basic navigation interface smoothly, but all of them had some complaints or suggestions related to it.

Participants *A* and *B* noted that the navigation island Home and Up buttons were separated from the Current Path panel, which had clickable folder names. They suggested that these two should be grouped to prevent users from having to move back and forth between them when they are navigating.

Both participants *B* and *D* suggested that there should be a traditional tree view of the directory structure to make it easier for the user to explore projects with deeply nested folder hierarchies.

Participant *A* also suggested having Simulation Mode on the same screen and not having a separate modal open up for it to increase the ease of navigation.

6.2.3 Filtering

From the list of tasks, which was the first part of the evaluation, it is seen that both participants *C* and *D* had trouble with the filtering tasks, which they could not complete. Both failed Tasks 9 and 10. A point to note here is that both participants *A* and *B* evaluated the tool on the same project. It was tempting to conclude that regular expressions might perhaps not be suitable for filtering or that the participants lacked familiarity with regular expressions. Instead, the situation was to the contrary. Both participants admitted that while they were not experts, they were used to filtering and searching for files using regular expressions on their IDEs. They also agreed on regular expressions being a suitable interface for filtering. Their inability to complete the tasks seems to have stemmed from the condensation process, which led to issues with how the data was filtered.

6.2.4 Contributors List

This list was quite straightforward to use. All participants except participant B were able to complete the tasks related to it as shown in Table 5.1.

Participant B were unable to complete it because they were working with a relatively smaller screen size which introduced some clipping issues. The contributors were visible from the list but their contribution percentages were cut off. Other participants, while being able to complete the tasks, had similar problems as well with participants$C$ and D reporting the same issue in Table 5.6.

6.2.5 Bus Factor Simulation

There were two main usability issues encountered with the usage in this section. Participant B had trouble grasping whether the checkboxes next to the contributors' names meant they were being selected for staying or leaving. Both participants B and C had trouble with the long list of contributors not having a 'Reset' or a 'Clear All' button. They suggested that we add one there.

6.3 Overall Usefulness of BFViz (RQ3)

Feedback for the overall usefulness of BFViz was limited. Most of the feedback even when we asked direct questions about BFViz as an overall application tended to focus on the features that the participants felt were missing. One comment by participant B is worth highlighting: They stated that such a tool would be much better off being integrated into a system that is already commonly utilized in the development lifecycle. They suggested having it integrated into the IDE or the CI/CD system would be helpful. They declared they would be averse to using BFViz if it was implemented as a separate platform that they would have to navigate back and forth from. We received similar comments from participant A as well who along with participant B inquired if BFViz could be made part of

the GitHub Insights page.

There were several comments about enhancements that would make BFViz even more useful to the use cases the participants had in mind. The most popular ones were allowing the user to choose which files and folders to include and which ones to exclude when visualizing projects. Participants *B* and *D* suggested having some sort of rule engine to make it easier for users to define inclusion and exclusion conditions. A similar one was to allow the same for contributors, letting users choose which ones to keep and which ones to not track.

Participant *B* suggested implementing dependency highlighting and highlighting areas of the project based on the frequency of activity in those areas. They also explained that as a manager of multiple software repositories, their user experience would be improved by having bus factor visualizations of multiple projects at the same time. With such a feature, they would have a hawk-eye view of the overall bus factor situation across their entire team. They also put forth a case for having BFViz integrated in a CI/CD system and generating periodic reports, citing security in enterprise IT environments as an issue. They were of the view that IT personnel would be unwilling to let a public tool access Git repositories without checks and balances, and that it would be better utilized if it they put it in a destination that is already under their watch.

Out of all the participants, participant *B* outright proclaimed that BFViz is a tool that they find useful for solving a problem that they face in real life. Other participants were less clear about their decision. Participant *A* enquired whether they could try the tool on some other open-source projects that they were not owners of but were curious about. They also asked about when they would be able to see BFViz in action as a JetBrains product. Participant *D* also had the same question. Participant *C*, while finding certain individual features useful, did not provide a clear answer on the overall usefulness of BFViz.

Chapter 7

Threats to Validity

We introduce a new tool for bus factor visualization which can also simulate developer departure and its impact on the bus factor. We attempt to validate it by performing case studies with core contributors to Git repositories that fit our criteria. While designing, developing, and validating our tool, we ran into constraints along the way that may impact the validity of this assessment.

7.1 Construct Validity

The tool we introduce, BFViz, visualizes the bus factor for Git projects. It is, however, not responsible for the calculation of the bus factor. Rather, it consumes data from an external source and then visualizes it. This decoupling is an advantage by allowing users to use the bus factor algorithm of their choice to generate data and visualize it. We chose the definition and algorithm that produced the best metrics [4] at the time of writing this thesis. A different algorithm could have led to different results, impacting the responses of the case study participants. It is important to note that there is still no work on the ground truth of the bus factor in a project validating the different algorithms to the best of the authors' knowledge.

For the research questions that we have defined, we do not utilize metric-based definitions of 'usefulness', 'ease-of-use', or 'helpfulness', instead, we rely on the participants' definitions of those terms. We also do not explicitly ask the participants to clarify what their definitions of these terms are. Their definitions of the terms might have slight differences from one participant to the next.

7.2 Internal Validity

There are several factors that the authors had to optimize for while developing the application. The data included in the bus factor analysis only takes into account data going back 1.5 years from the last commit. As we did not take into account the contribution data before that period, our bus factor statistics could have been different. This could affect the response by case study respondents as it may not match up with their assessment of the situation. There are some thresholds and coefficients that are set manually in the algorithm. Depending on the value of these constants, the output of the algorithm and the visualization would be different. Since our tool is an interactive tool, there are also some features in which the opinion of case study respondents might be split due to personal preference.

We only had four participants with three different roles in the team as shown in Table 4.4. This number is relatively small and our results might have been different had we managed to get a higher number of respondents. We have already mentioned the problems we faced in establishing contact with eligible projects in Chapter 4. On top of that, we had to split the recruitment into two streams, only one of which went through the qualification process. This might have affected the validity of our results as well. Depending on their designation, participants might have different expectations from such an application. Someone in a more managerial position might be more interested in enlisting the aid of such a tool to make hiring or firing decisions easier. On the other hand, someone in a more technical role might use this tool to figure out who to ask for help from for a certain part of the project. The nature and domain of the project may also play

a role in the significance that is attached to bus factor data.

Other than participants, we also had an exceptional case with one of the projects. Project C is a fork of an open-source Android project. From our discussion with the participants who evaluated BFViz this project, a majority of its 170k commits can safely be assumed to originate externally outside of JetBrains. For this reason, we had to show a condensed version of the project to the participants who evaluated it, targeting the areas that they were involved with. This lead to some issues with the filtering functionalities as well as missing files and folders at some locations. This may have been another factor affecting the results.

Another factor that might have affected the validity is the summarization process of the participant responses in the semi-structured interviews. Since the original responses were both, quite verbose, and also compromising anonymity, we had to resort to anonymization and summarization.

7.3 External Validity

We perform case studies across 3 GitHub projects of different sizes and platforms to validate our tool and get user feedback. First, all of our projects were open source. We did not perform any case studies on commercial, closed-source software projects. Our results could have been different if we had targeted them. We tried maintaining diversity across our projects. Our smallest project has 188 stars while our largest project has around 330 stars. Our most popular project, Project *C* (330 stars), is still not comparable to enterprise-level projects such as the Linux kernel [1] (164k stars) and Google's Kubernetes project [2] (105k stars). In terms of number of commits (170k), it surpasses Kubernetes (120k) but is still nowhere close to Linux (1.2 million). Covering a wider spectrum of projects, in size, domain, and popularity could have yielded different results.

[1] `https://github.com/torvalds/linux`
[2] `https://github.com/kubernetes/kubernetes`

Chapter 8

Conclusion

In this thesis, we proposed and implemented, in collaboration with JetBrains, BFViz, a web-based tool to visualize bus factor and contribution data for Git-based software repositories. The tool offers a navigable visualization of the bus factor data on files and folders of the repositories, provides regular-expressions-based filtering options, a list of contributors and their contribution percentages, and a simulation mode with which users can evaluate changes in bus factor if one or more contributors leave. The objective of the tool is to simplify bus factor data consumption for users by providing an interactive visualization.

We recruited four participants across three projects in total to perform a user study on BFViz. Two of those projects (and their accompanying users) were selected from GitHub via a qualification process. The remaining two were selected via internal invites to colleagues in JetBrains. The study was split into three parts: a list of tasks for participants to perform using the tool, a semi-structured interview, and finally getting the participants to rank the five main features in descending order of usefulness.

The user study revealed that most of the features implemented were viewed as useful. Filtering was one feature that was pointed out by users as needing serious rework. Perhaps our most novel feature, the bus factor simulation mode,

was specifically highlighted by one of the four participants as solving a problem that they frequently experience as an engineering manager in their work. We also received a lot of feedback about pain points and issues that should be resolved to improve the user experience. We also received some feature requests that would increase the value that this tool provides for different use cases.

As part of our future work, we are planning to fix the issues highlighted, prioritize the features requested, and perform further studies to establish more conclusive results and gain feedback from a larger and more diverse set of users.

Bibliography

[1] V. Cosentino, J. L. C. Izquierdo, and J. Cabot, "Assessing the bus factor of git repositories," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 499–503, IEEE, 2015.

[2] G. Avelino, L. Passos, A. Hora, and M. T. Valente, "A novel approach for estimating truck factors," in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, pp. 1–10, IEEE, 2016.

[3] N. Zazworka, K. Stapel, E. Knauss, F. Shull, V. R. Basili, and K. Schneider, "Are developers complying with the process: An XP study," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, (New York, NY, USA), Association for Computing Machinery, 2010.

[4] E. Jabrayilzade, M. Evtikhiev, E. Tüzün, and V. Kovalenko, "Bus factor in practice," in *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, pp. 97–106, 2022.

[5] V. Haratian, M. Evtikhiev, P. Derakhshanfar, E. Tüzün, and V. Kovalenko, "Bfsig: Leveraging file significance in bus factor estimation," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2023, (New York, NY, USA), p. 1926–1936, Association for Computing Machinery, 2023.

[6] A. Unwin, "Why is data visualization important? What is important in data visualization?," *Harvard Data Science Review*, vol. 2, Jan 31 2020. https://hdsr.mitpress.mit.edu/pub/zok97i7p.

[7] J. O. Coplien and N. B. Harrison, *Organizational patterns of agile software development*. Prentice-Hall, Inc., 2004.

[8] P. N. Robillard, "The role of knowledge in software development," *Commun. ACM*, vol. 42, p. 87–92, jan 1999.

[9] M. Rashid, P. M. Clarke, and R. V. O'Connor, "A systematic examination of knowledge loss in open source software projects," *International Journal of Information Management*, vol. 46, pp. 104–123, 2019.

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

[11] J. Vanhanen and C. Lassenius, "Effects of pair programming at the development team level: an experiment," in *2005 International Symposium on Empirical Software Engineering, 2005.*, pp. 10 pp.–, 2005.

[12] J. Vanhanen and H. Korpi, "Experiences of using pair programming in an agile project," in *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pp. 274b–274b, 2007.

[13] P. C. Rigby, Y. C. Zhu, S. M. Donadelli, and A. Mockus, "Quantifying and mitigating turnover-induced knowledge loss: Case studies of chrome and a project at avaya," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pp. 1006–1016, 2016.

[14] F. Ricca, A. Marchetto, and M. Torchiano, "On the difficulty of computing the truck factor," in *International Conference on Product Focused Software Process Improvement*, pp. 337–351, Springer, 2011.

[15] C. Hannebauer and V. Gruhn, "Algorithmic complexity of the truck factor calculation," in *Product-Focused Software Process Improvement* (A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, and M. Raatikainen, eds.), (Cham), pp. 119–133, Springer International Publishing, 2014.

[16] T. Fritz, J. Ou, G. C. Murphy, and E. Murphy-Hill, "Degree-of-knowledge: modeling a developer's knowledge of code," in *Proceedings of the ACM/IEEE 32nd International Conference on Software Engineering-Volume 1*, p. 385–394, 2010.

[17] M. Ferreira, M. T. Valente, and K. Ferreira, "A comparison of three algorithms for computing truck factors," in *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pp. 207–217, 2017.

[18] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 107–117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[19] A. Lisan and B. Norris, "Guiding effort allocation in open-source software projects using bus factor analysis," *arXiv preprint arXiv:2401.03303*, 2024.

[20] V. Orrei, M. Raglianti, C. Nagy, and M. Lanza, "Contribution-based firing of developers?," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2023, (New York, NY, USA), p. 2062–2066, Association for Computing Machinery, 2023.

[21] K. Højelse, T. Kilbak, J. Røssum, E. Jäpelt, L. Merino, and M. Lungu, "Git-truck: Hierarchy-oriented visualization of git repository evolution," in *2022 Working Conference on Software Visualization (VISSOFT)*, pp. 131–140, 2022.

[22] E. Klimov, M. U. Ahmed, N. Sviridov, P. Derakhshanfar, E. Tüzün, and V. Kovalenko, "Bus factor explorer," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 2018–2021, 2023.

[23] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966.

[24] B. Johnson and B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," in *Proceedings of the 2nd conference on Visualization'91*, pp. 284–291, 1991.

[25] W. Wang, H. Wang, G. Dai, and H. Wang, "Visualization of large hierarchical data by circle packing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, (New York, NY, USA), p. 517–520, Association for Computing Machinery, 2006.

Appendix A

Code

The code for BFViz is available at `https://github.com/JetBrains-Research/bf-vis`