TU WIEN Informatics

Visualizing Software Evolution

A Plot-Based Exploration of Software Quality Metrics

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Simon Garuda Linder, BSc
Matrikelnummer 11728330

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl
Mitwirkung: Univ.Lektor Dipl.-Ing. Dr.techn. Alexander Schatten
Philip König, BA MSc

Wien, 16. Juni 2026

Simon Garuda Linder | Edgar Weippl

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

Visualizing Software Evolution

A Plot-Based Exploration of Software Quality Metrics

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Simon Garuda Linder, BSc
Registration Number 11728330

to the Faculty of Informatics

at the TU Wien

Advisor: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl
Assistance: Univ.Lektor Dipl.-Ing. Dr.techn. Alexander Schatten
Philip König, BA MSc

Vienna, June 16, 2026

Simon Garuda Linder | Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Simon Garuda Linder, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 16. Juni 2026

Simon Garuda Linder

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Mein herzlicher Dank gilt meinen Betreuern, Herrn Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl, Univ.Lektor Dipl.-Ing. Dr.techn. Alexander Schatten und Philip König, BA MSc, für ihre wertvolle Unterstützung, ihre konstruktiven Rückmeldungen und ihre fachliche Begleitung während der Erstellung dieser Masterarbeit.

Darüber hinaus möchte ich mich bei allen Interviewteilnehmenden bedanken, die sich die Zeit genommen haben, ihre Erfahrungen und wertvollen Einblicke mit mir zu teilen. Ihre Beiträge waren für diese Arbeit von großer Bedeutung.

Abschließend danke ich meiner Familie und meinen Freund:innen für ihre Unterstützung, Geduld und Ermutigung während meines Studiums.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl, and my supervisors, Univ.Lektor Dipl.-Ing. Dr.techn. Alexander Schatten and Philip König, BA MSc, for their valuable support, constructive feedback, and expert guidance throughout the preparation of this master's thesis.

Furthermore, I would like to thank all interview participants who took the time to share their experiences and valuable insights with me. Their contributions were highly valuable to this research.

Finally, I would like to thank my family and friends for their support, patience, and encouragement throughout my studies.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Moderne Softwaresysteme entwickeln sich kontinuierlich weiter, wobei sich Qualitätsprobleme wie zunehmende Komplexität, technische Schulden und abhängigkeitsbezogene Risiken im Laufe der Zeit ansammeln können. Die Bewältigung dieser Probleme erfordert Entscheidungen von Stakeholdern mit unterschiedlichen Verantwortlichkeiten, technischen Hintergründen und Informationsbedürfnissen.Die Softwarequalitätsforschung hat eine Vielzahl von Metriken hervorgebracht, die die Entwicklung und Wartung qualitativ hochwertiger Systeme unterstützen. Bestehende Visualisierungen richten sich jedoch hauptsächlich an Entwicklerinnen und Entwickler, während der Informationsbedarf anderer Stakeholder und geeignete Formen der Vermittlung von Informationen über Softwarequalität an diese bislang nur unzureichend untersucht wurden. Diese Arbeit untersucht, welche Visualisierungen von Metriken zur Softwarequalität und -evolution für verschiedene Stakeholdergruppen verständlich und aufgabenrelevant sind.

Zur Beantwortung dieser Frage wurden 43 Visualisierungen entworfen, von denen 13 im Rahmen von neun semistrukturierten Experteninterviews hinsichtlich ihrer wahrgenommenen Verständlichkeit und Aufgabenrelevanz evaluiert wurden. Mit Angehörigen der drei Stakeholdergruppen, nämlich technischen Mitwirkenden, technischen Führungskräften und Produktrollen, wurden jeweils drei Interviews durchgeführt.

Von den 13 vorgeschlagenen Visualisierungen wurden zehn von mindestens einer Stakeholdergruppe als aufgabenrelevant und verständlich wahrgenommen und erreichten bei beiden Kriterien Medianwerte von mindestens 4 von 5 Punkten. Zwei Visualisierungen waren für alle drei Stakeholdergruppen aufgabenrelevant und verständlich. Darüber hinaus wiesen Teilnehmende aus Produktrollen insgesamt niedrigere Medianwerte für die Verständlichkeit auf als Teilnehmende aus den beiden anderen Stakeholdergruppen.

Die Ergebnisse zeigen, dass sich die wahrgenommene Verständlichkeit und Aufgabenrelevanz von Visualisierungen von Softwarequalitätsmetriken zwischen den Stakeholdergruppen unterscheiden. Die evaluierten Visualisierungen können Stakeholder dabei unterstützen, Informationen zur Softwarequalität zu interpretieren und sich abzeichnende Qualitätsprobleme zu erkennen, und dadurch fundierte Entscheidungen über die langfristige Qualität und Weiterentwicklung komplexer Softwaresysteme erleichtern.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Modern software systems evolve continuously, and quality concerns such as increasing complexity, technical debt, and dependency-related risks can accumulate over time. Addressing these concerns requires decisions by stakeholders with differing responsibilities, technical backgrounds, and information needs. Software quality research has produced a broad range of metrics to support the development and maintenance of high-quality systems. However, existing visualizations primarily target developers, while the information needs of other stakeholders and suitable ways of communicating software quality information to them remain underexplored. This thesis investigates which visualizations of software quality and evolution metrics are comprehensible and task-relevant for different stakeholder groups.

To address this question, 43 visualizations were designed, of which 13 were evaluated with regard to their perceived comprehensibility and task-relevance in nine semi-structured expert interviews. Three interviews were conducted with each of the three stakeholder groups: technical contributors, technical leadership, and product roles.

Of the 13 proposed visualizations, 10 were perceived as task-relevant and comprehensible by at least one stakeholder group, achieving median scores of at least 4 out of 5 on both measures, while two were task-relevant and comprehensible for all three stakeholder groups. Additionally, participants in product roles had lower median comprehensibility scores overall than participants in the other two stakeholder groups.

The results show that the perceived comprehensibility and task-relevance of software quality metric visualizations vary across stakeholder groups. The evaluated visualizations can support stakeholders in interpreting software quality information and recognizing emerging quality concerns, thereby facilitating informed decisions about the long-term quality and evolution of complex software systems.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**

2 Related Work **5**
2.1 Software Quality Metrics 5
2.2 Visualization of Software Quality 7
2.3 Stakeholder Information Needs 8
2.4 Evaluation of Software Visualizations 8
2.5 Research Gap 9

3 Methodology **11**
3.1 Problem Identification and Motivation 11
3.2 Definition of the Objectives for a Solution 12
3.3 Design and Development 12
3.4 Demonstration 22
3.5 Evaluation 25
3.6 Communication 25

4 Results and Discussion **27**
4.1 Overview of Evaluation Data 27
4.2 Evaluation Criteria 27
4.3 Overall Results and Discussion 28
4.4 Limitations 32

5 Conclusion **35**

A Interview Outline **37**
A.1 Einleitung 37
A.2 Fragen zur Person und Rolle 38

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

A.3 Bewertung der Visualisierungen 38
A.4 Abschlussfragen 39
A.5 Optionale Nachfragen 39

B Visualizations **41**
B.1 Metrics 41
B.2 Visualizations evaluated in the interviews 42
B.3 Visualizations not evaluated in the interviews 50
B.4 Unsuccessful visualization ideas 60

Overview of Generative AI Tools Used **63**

List of Figures **65**

List of Tables **67**

Bibliography **69**

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

Modern software systems consist of numerous components and external dependencies. [BG20] As software systems evolve over time, their complexity can increase, technical debt can accumulate and remain unnoticed, and dependency-related risks can be overlooked. [Leh80, Kra22] These challenges do not end once initial development is complete. Maintenance and evolution are ongoing parts of the software lifecycle, during which software quality benefits from continuous monitoring and management. [Hin96] If these challenges are not identified and addressed in time, they can negatively affect the long-term condition of a software system.

Poor software quality can cause a range of problems, including operational failures, increased effort for finding and correcting defects, project failure, and potential security vulnerabilities. [Kra22] The Consortium for Information & Software Quality estimates that the cost of poor software quality in the United States amounted to approximately USD 2.41 trillion in 2022. [Kra22] Addressing such concerns requires decisions about which quality issues to address, when to address them, and which resources to allocate.

These decisions are influenced not only by those who implement and maintain a system, but also by stakeholders who make scheduling decisions, allocate resources, or make decisions about technical risks. [Pre10] As stakeholders involved in software-quality-related decisions differ in their responsibilities, technical backgrounds, and information needs, the level of detail, terminology, and type of information presented to them should vary accordingly. [SCB+18]

Software quality metrics can provide a basis for communicating such information. Software quality research has produced a rich set of metrics that capture code evolution, knowledge distribution, technical risk, and dependency health [Kan03]. How these metrics should be communicated visually to practitioners and whether appropriate representations differ across roles such as developers, technical managers, and non-technical managers remains largely unaddressed. Existing software visualization research over-

whelmingly targets developers, leaving managerial and non-technical stakeholders as an understudied audience [CMZ+20], and rigorous evaluation of visualizations against specific stakeholder needs is rare in the field [MGAN18]. At the same time, the lack of stakeholder-differentiated metric communication has been identified as an explicit gap in software quality practice [HP22].

The aim of this thesis is to design and evaluate visualizations of software quality metrics that extend beyond a developer-centered perspective. It focuses on additional stakeholders who are interested in and are able to influence software quality. The visualizations were evaluated through semi-structured expert interviews with regard to their perceived task-relevance and comprehensibility. The interviews address the following three research questions:

1. Which of the proposed visualizations are effective ways to visualize software quality and evolution metrics for at least one stakeholder group, based on median task-relevance and comprehensibility?
2. How does the median comprehensibility of the proposed visualizations differ between stakeholder groups?
3. Among the proposed visualizations, are there any that are task-relevant and comprehensible for all stakeholder groups, and which are stakeholder-group-specific, based on median scores?

Within this thesis, a broad range of software quality metrics was considered and used to design 43 visualizations. These were designed in an iterative process with the help of an industry partner. Of the 43 visualizations, 13 were evaluated within the expert interviews. Three stakeholder groups were defined: technical contributors, product roles, and technical leadership. The visualizations were evaluated with professionals from each stakeholder group.

The technical contributors consisted of:

- Lead developer
- Lead developer and penetration tester
- Senior developer

The product roles consisted of:

- CEO (without a technical background)
- CIO
- Product Owner

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The technical leadership consisted of:

- CEO (with a technical background)
- Business Domain Tester
- Senior Manager

This thesis contributes a set of 13 visualizations together with an evaluation of their perceived task-relevance and comprehensibility.

The remainder of this thesis is structured as follows. Chapter 2 reviews related work on software quality metrics, software visualization, stakeholder-specific information needs, visualization evaluation, and the research gap. Chapter 3 presents the research design, visualizations, stakeholder groups, and the interview procedure. Chapter 4 reports and discusses the results of the expert interviews. Finally, Chapter 5 summarizes the work, its limitations, and opportunities for future research.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Related Work

This thesis lies at the intersection of two established research areas, specifically software quality metrics and software visualizations. Section 2.1 reviews the software quality metrics from which the visualizations in this thesis are constructed. Section 2.2 reviews approaches to visualizing such metrics and the audiences for which they have been designed and evaluated. Section 2.3 draws on empirical studies to establish what the stakeholder groups considered in this thesis require. Section 2.4 reviews how software visualizations have been evaluated and identifies the criteria relevant here. Section 2.5 states the resulting research gap.

2.1 Software Quality Metrics

Prior work has proposed a broad range of metrics for assessing different properties of software systems. These metrics can be derived directly from repository data, calculated from source code, or produced by specialized models. This thesis considers several metric groups that describe complementary aspects of software quality and software development:

- **Commit Type Metrics:** provide basic information about a commit, such as whether it is classified as a bug-fixing, refactoring, or feature commit.
- **Metadata-related metrics:** identify commits, projects, files, and developers. Although this information is not directly displayed in our visualizations, it is required to support filtering and aggregation in a visualization.
- **Process and History Metrics:** describe the development history of a file at a particular point in time. Examples include the number of bug-fix commits associated with a file, its age, and the number of developers who have modified it.

- **Churn Metrics:** describe the amount of code changed in a commit, for example through the number of added, removed, or modified lines.
- **Degree of Knowledge Metrics:** estimate how well a developer knows a file or a set of code artifacts.
- **Risk Score:** represents the predicted defect risk of a file when it is committed, as produced by a machine-learning model.
- **Cognitive Complexity:** estimates how difficult source code is to understand based on its control-flow structure.
- **Dependency and Update Indicators:** describe how a project manages its external dependencies over time, including dependency updates, known vulnerabilities, and the time required to remediate vulnerabilities.

2.1.1 Process, History, and Churn Metrics

Commit type, process and history, and churn metrics are derived directly from version-control data. Beyond providing context on the evolution of a system, they have an established empirical justification. Rahman and Devanbu [RD13] compared process metrics against static code metrics and found process metrics to be the stronger basis for defect prediction.

Rahman and Devanbu [RD13] also found that static code metrics do not change much between releases. Commit type, process and history metrics and churn metrics, on the other hand, reflect where work is actually happening. This property is relevant for our visualization. Metadata-related metrics connect all metrics to commits, files, projects, and developers.

2.1.2 Degree of Knowledge Metrics

This thesis uses the Degree of Knowledge (DoK) formulation of Carlson [Car15], which is computed from version-control data alone. It builds on the measure of Gîrba et al. [GKSD05] and extends it in two respects. First, it incorporates a developer's earlier commits rather than only the current snapshot. Secondly, it uses exponential decay so that the knowledge attributed to a developer diminishes as time passes.

Knowledge-related metrics can reveal whether knowledge is broadly distributed across a team or concentrated in a small number of individuals.

Three measures derived from DoK are used in this thesis. Knowledge difference expresses the DoK of the developer modifying a file relative to the highest DoK held by any developer for that file. The bus factor is the minimal number of developers who would have to become unavailable (i.e., hit by a bus) before no one remaining knows a given file or folder well. The relative bus factor expresses an individual's share of the bus factor of a file or folder relative to the rest of the team. All three are computed here directly from DoK values rather than following a published estimation algorithm.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.1.3 Defect Risk

The risk-score metric used in this thesis [Fel25] estimates the defect risk of a file at commit time using a machine-learning model.

2.1.4 Cognitive Complexity

Cognitive Complexity, proposed by Campbell [Cam18], estimates the mental effort required to understand code from aspects of control flow such as nesting and interruptions of linear flow. It was designed to measure understandability rather than testability, which distinguishes it from Cyclomatic Complexity, and it does not measure the inherent difficulty of the problem being solved.

2.1.5 Dependency and Update Indicators

These describe the state of external libraries used by a project, covering maintenance and security concerns that internal code metrics cannot capture.

2.1.6 Selection of Metrics

The set of metrics available for visualization was determined in part by an existing analysis pipeline that derived metrics through version-control data. This provided a starting point, which was then filtered to remove metrics where we could not find any use cases for any of our stakeholder groups. After that, we expanded the list with new metrics which we believed were useful based on the information needs of the stakeholder groups and within the scope of this thesis.

2.2 Visualization of Software Quality

Software quality metrics produce large, temporal, multidimensional datasets that are difficult to analyze in tabular form. Comparing modules or projects adds additional complexity. Software visualizations can support the analysis of such systems [MGAN18] by making patterns, trends, and anomalies more accessible, and can facilitate communication about quality between stakeholders with different technical backgrounds.

Although our visualizations are not presented as a dashboard, we follow Sarikaya et al. [SCB+18] in treating the audience as an explicit dimension of design. They argue that designing for users with differing visual and analytic literacy cannot be addressed by assuming fixed designs for singular audiences. This is supported by Treude and Storey [TS10], who found that across several development teams the use of dashboards varied significantly with the role of the user, and that managers relied on dashboards to communicate project status to executives.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.2.1 Intended Audiences of Existing Approaches

In a literature review by Chotisarn et al. [CMZ+20], 105 full papers on software visualizations published between 2013 and 2019 were reviewed. As shown in Table 2.1, they recorded the target roles of these papers. Developer was by far the most frequent, occurring in 63 papers. However, roles who do not write code, such as project manager and end-user, appeared only a few times. Additionally, in 23 cases, since the target audience could not be identified, the papers not only do not differentiate between different stakeholders, but they do not mention a specific target audience at all. This review predates more recent measurement aimed at engineering leadership, such as the SPACE [FSM+21] and DevEx [GSN22] frameworks. These concern the delivery process and the experience of developers rather than properties of the codebase and are therefore not substitutes for the indicators considered here.

Table 2.1: Frequency of target roles recorded by Chotisarn et al. [CMZ+20] across 105 papers published between 2013 and 2019. Papers may be assigned more than one role.

Role	Frequency
Developer	63
Practitioner	11
Software Engineer	8
Maintainer	5
End-user	4
Project Manager, Researcher, Tester	3
Analyst	2
Architect	2
Team Member	1
Not Identified	23

2.3 Stakeholder Information Needs

The level of detail, terminology, and type of information required can be expected to differ between stakeholder groups. Sarikaya et al. [SCB+18] argue that designing for users with differing visual and analytic literacy cannot be addressed by assuming fixed designs for singular audiences. Within software engineering, Buse and Zimmermann [BZ12] provide empirical support from a survey of 110 developers and managers at Microsoft. They concluded that due to different information needs of stakeholders, tools should support different views for managers and developers.

2.4 Evaluation of Software Visualizations

Software visualizations should be evaluated with respect to whether they support stakeholders in understanding information and whether the information is relevant to the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

stakeholder. Munzner's nested model [Mun09] distinguishes validation at the level of the domain problem characterization, the operation and data type abstraction, the visual encoding and interaction design, and the algorithm design. We cover the first three levels with our measures of comprehensibility and task-relevance, which are further defined in Section 3.1.

2.5 Research Gap

Software quality metric visualizations are well established, but as shown above, they have predominantly been designed for and evaluated with developers in mind. What has not been established is whether concrete visualizations of these metrics are comprehensible and task-relevant for non-developer stakeholders who are essential for the success of a software project. Additionally, as stated in Section 2.3, a fixed design cannot serve all stakeholders, as they differ both in visual and analytic literacy [SCB+18] and in the information they need [BZ12]. These two dimensions motivate our evaluation criteria. We aim to test whether a visualization suits the visual and analytic literacy of a stakeholder by measuring the perceived comprehensibility. With task-relevance, we aim to test whether a visualization satisfies the information needs of a stakeholder. This thesis addresses this gap by designing a set of software quality metric visualizations and evaluating their comprehensibility and task-relevance with technical contributors, technical leadership, and product roles.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

Methodology

As this thesis follows a *design science approach*, it is guided by the design science framework from Peffers et al. [PTRC07]. This framework consists of six steps:

1. problem identification and motivation
2. definition of the objectives for a solution
3. design and development
4. demonstration
5. evaluation
6. communication

3.1 Problem Identification and Motivation

In this step, we define a research problem and justify the value of our solution. To define the problem more precisely, we use two evaluation measures: comprehensibility and task-relevance. Comprehensibility describes how easy a visualization is to understand, while task-relevance describes how useful a visualization is for a particular stakeholder. Together, these two measures allow us to assess the visualizations with respect to Levels 1 to 3 of Munzner's nested validation model [Mun09]. Task-relevance addresses whether the displayed information is needed by the stakeholder and whether the selected data supports the stakeholder's tasks, corresponding to Levels 1 and 2. Comprehensibility addresses whether the chosen representation is easy to understand, corresponding to Level 3.

To define the target audience of the visualizations, we group stakeholders into three categories: technical contributors, technical leadership, and product roles. Technical

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

contributors work directly with code, for example, developers and some types of software testers. Technical leadership refers to stakeholders with a technical background who do not contribute code themselves, such as a technical product owner or a security lead. Product roles include stakeholders without a technical background who do not implement software, for example, a non-technical product owner or a business analyst. We only consider stakeholders who both influence and are interested in the quality of software. Stakeholders outside this scope, such as regulatory stakeholders, are not considered.

The problem addressed in this thesis is that current visualizations of software quality metrics mainly target developers rather than other stakeholders. To address this gap, we propose a set of software quality metric visualizations and evaluate them using semi-structured expert interviews. The value of these visualizations is reflected in the task-relevance measure, which captures their usefulness to a stakeholder.

3.2 Definition of the Objectives for a Solution

In this step, we derive the objectives of the solution from the problem statement defined in the previous step. These objectives are as follows:

- to provide a set of visualizations for software quality metrics
- to ensure that these visualizations are task-relevant and comprehensible for at least one stakeholder group
- to design some visualizations to be suitable across all stakeholder groups, while others are more specific to particular groups
- to keep the set of visualizations small enough to allow a meaningful evaluation within the scope of this thesis.

Together, these objectives define the requirements that the proposed visualizations should satisfy.

3.3 Design and Development

The visualizations were developed using three complementary approaches. First, we formulated potential use cases and scenarios in order to identify which data would be required to support them. Second, we considered potentially relevant software quality metrics and explored which use cases these metrics could address. Third, we consulted the literature for relationships between the metrics. Based on these inputs, we designed visualizations that matched the intended use cases and had available data.

The resulting visualizations were reviewed in discussions with supervisors in internal feedback sessions. These discussions helped identify weaknesses, refine existing visualizations, and generate additional ideas. During this time, we consulted with an industry partner

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

to obtain external feedback. Based on this feedback, the visualizations were repeatedly adjusted and reviewed until the resulting set of visualizations was expected to meet the objectives defined in the previous step.

The design process resulted in 43 visualizations:

- 13 visualizations were selected for evaluation in the interviews (found in Subsection 3.3.1 and Appendix B).
- 17 visualizations were considered useful but were not selected for evaluation due to interview time constraints (found in Appendix B).
- 13 visualizations were rejected due to not fulfilling a clear use case or becoming obsolete (found in Appendix B).

3.3.1 Visualizations Designed

These are the metrics used in the visualizations:

- Lines of Code: the number of lines of code that a file has
- Lines of Code modified: the number of lines of code that are written, replaced, or deleted in a file in a commit.
- Bug-fix: a commit that is made to fix a bug or defect in the code.
- Cognitive Complexity [Cam18]: a measure of how difficult code is to read. The value 25 was chosen as a threshold because SonarQube uses it as a default threshold. [1].
- Highly Complex Files: the number of files that have a cognitive complexity greater than or equal to 25.
- Number of Dependencies Updated: how many external libraries are updated.
- Degree of Knowledge [Car15]: a measure of how well a person knows a file depending on what share of a file they have written and when they modified the file.
- Knowledge Difference: a measure derived from the degree of knowledge showing how well the developer who wrote the code knows the code compared to the person who knows this file the best. A developer modifying a file that another developer just wrote results in a knowledge difference of -1. On the other hand, a developer who knows a file better than anyone else has a knowledge difference of 0 when they modify this file further.

[1] SonarQube Threshold, accessed on 17 August 2026.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Bus Factor of a file: the number of people who would have to become unavailable at the same time for no one to still know that file well. Knowing a file well happens when a developer's degree of knowledge exceeds a threshold for this file.
- Bus Factor of a folder: derived from the degree of knowledge, it is the number of people who would have to become unavailable at the same time for at least half of the files in that folder to no longer be known well by any developer.
- Relative Bus Factor: describes how large one person's share of knowledge is compared to the rest of the team:
  - 1: Only this person knows the file/folder.
  - 0.5: This person knows about as much as all others combined.
  - 0: This person does not know the file/folder at all.
- Risk Score: the output of a machine-learning model that estimates how likely it is that a file contains a bug.

All visualizations include interactive features, such as showing data on hover and zooming. This is not possible within this thesis, since it is not an HTML document.

These are the visualizations presented to the interviewees:

1 Line Plot: Cumulative lines of code modified and cumulative Bug-fixes over time

Figure B.1: The blue line shows how many lines of code have been changed up to that point (cumulative churn). The red line shows how many bug-fixes have been made up to that point (cumulative bug-fixes).

It was chosen because it gives an overview of how much code is being written and how many bugs are fixed. Under the assumption that there is a constant ratio of bugs to the number of lines written, it can indicate whether bugs are being resolved at a rate that may prevent the accumulation of technical debt.

2 Tree map: Lines of code modified last year

Figure B.2: Here, the color shows how many lines of code were modified, considering only changes from the last year. The size of the rectangle shows its current number of lines of code.

This visualization gives an overview of a project, showing where work was done and which files are small or large.

3 Line Plot: Number of files that are highly complex that have been worked on per month

Figure B.3: This plot shows the number of files worked on in a month that have a cognitive complexity above 25.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

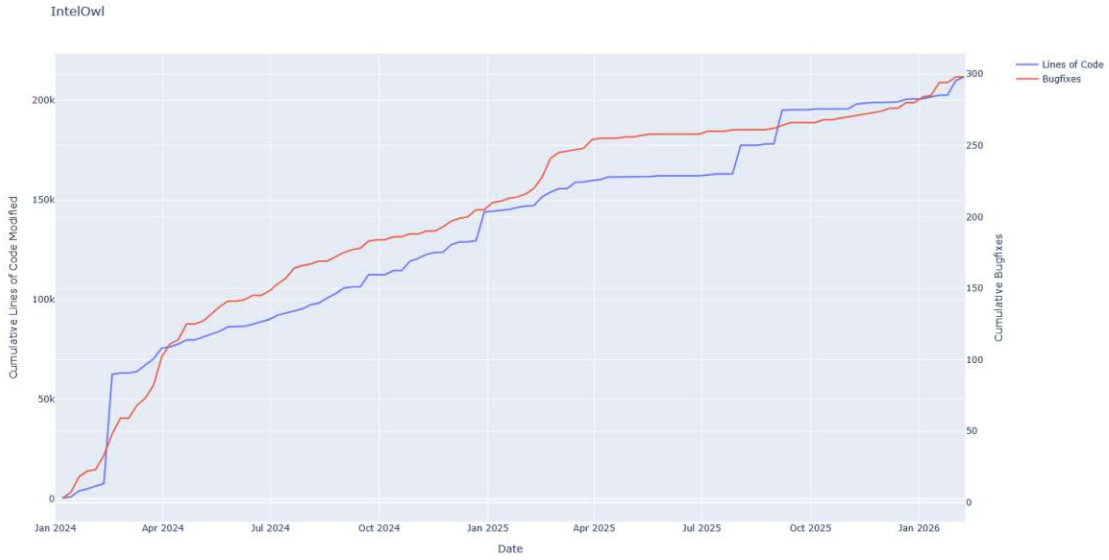

Figure 3.1: Cumulative Lines of Code modified and cumulative Bug-fixes over Time

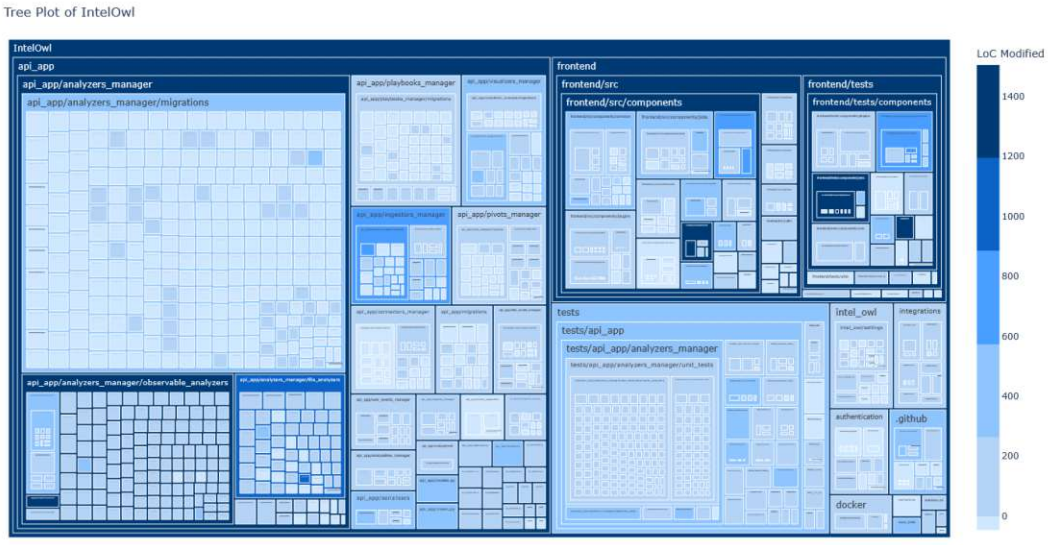

Figure 3.2: Lines of code modified last year

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

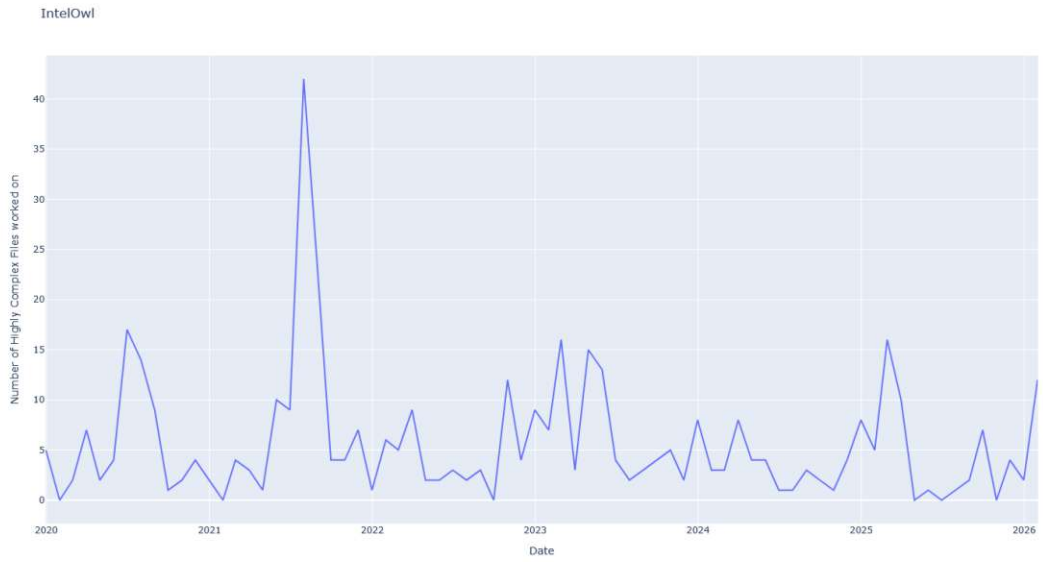

Figure 3.3: Number of files that are highly complex that have been worked on per month

With this plot, one can see a trend in the number of highly complex files that are worked on. This shows whether complex files are often created or worked on or if it is a rare occurrence.

4 Histogram: Average cognitive complexity per developer

Figure B.4: This shows the average cognitive complexity of the files that a person has changed. An average above 20 is considered complex.

This plot was chosen to show the cognitive complexity of the files modified by each developer.

5 Histogram: Current cognitive complexity of every file

Figure B.5: This shows the current cognitive complexity of each file.

With this visualization, one can see how many files have what cognitive complexity. This gives an overview of the cognitive complexity of the entire project.

6 Line Plot: Number of dependencies updated in a given window

Figure B.6: This plot shows how often the dependencies are updated.

Looking at this plot gives an overview of how frequently dependencies are updated.

7 Line Plot: Proportion of commits that are bug-fixes per month

Figure B.7: This view contains two line plots, one above the other. The lower plot shows the proportion of bug-fix commits among all commits. The upper plot shows how many commits there were at that time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

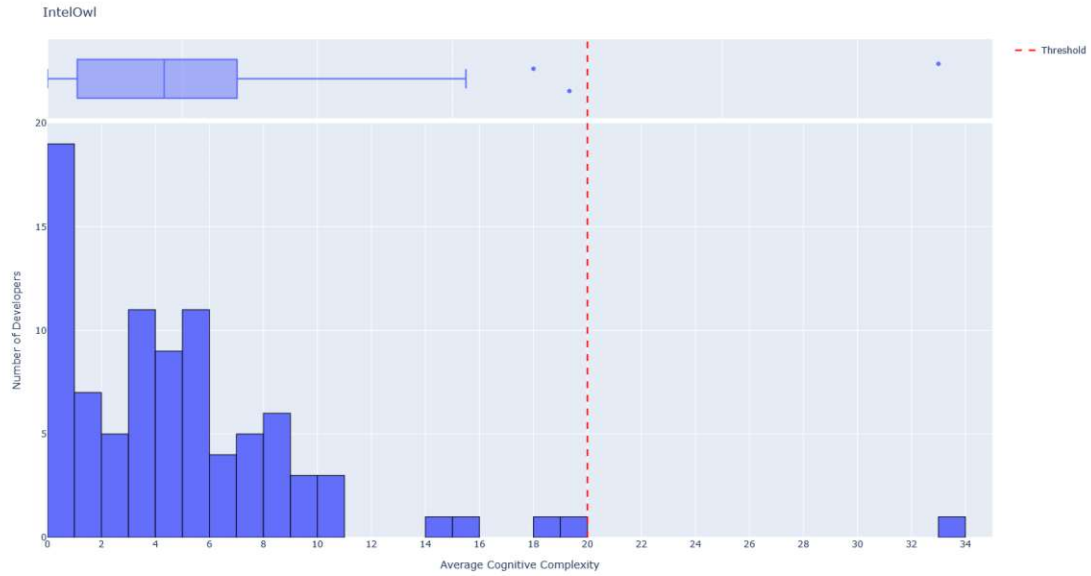

Figure 3.4: Average cognitive complexity of edited files per developer

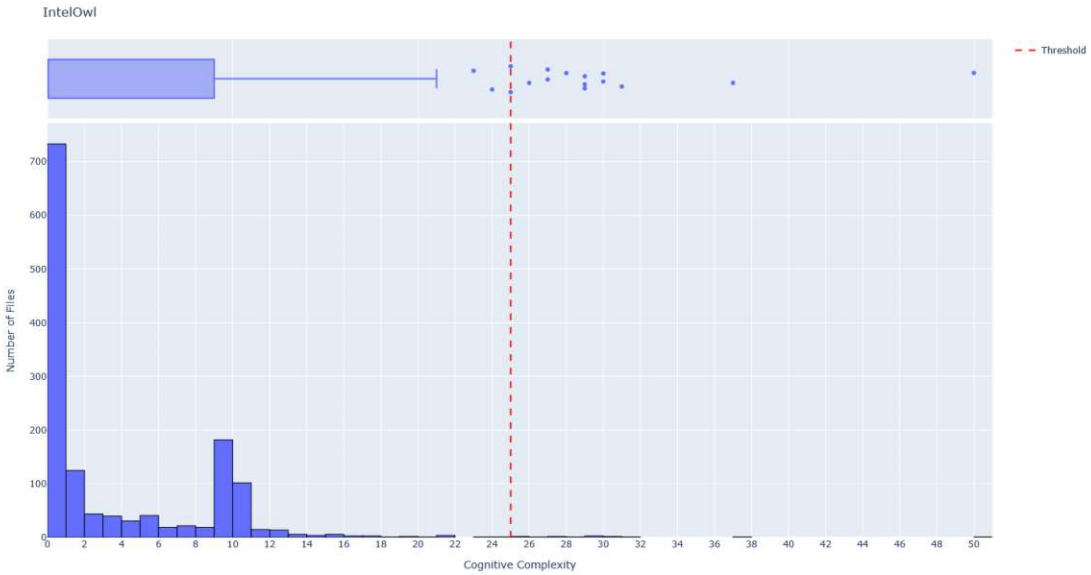

Figure 3.5: Current cognitive complexity of every file

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

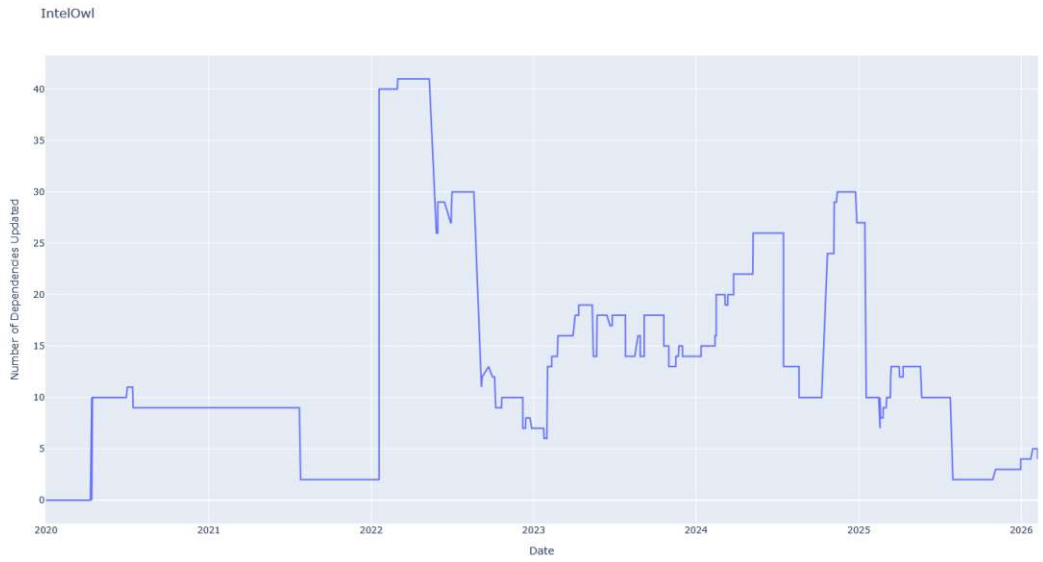

Figure 3.6: Number of dependencies updated in a given window

It was chosen to compare the number of bug-fix commits with the number of feature commits.

8 Histogram: Number of bug-fixes that a file was involved in last year

Figure B.8: This shows how often a file was involved in a bug-fix commit during the last year.

This visualization identifies which files were involved in the most bug-fix commits. These files may warrant investigation or refactoring because they appear to be bug-prone.

9 Scatter Plot: Average knowledge difference per month and number of developers per month

Figure B.9: Lower points show the average monthly knowledge difference, while the upper line shows the number of developers who made a commit in that month.

Knowledge difference can indicate whether developers typically modify files for which they have substantial prior knowledge or files primarily known by other developers.

10 Histogram: Average knowledge difference per developer

Figure B.10: This shows the average knowledge difference per developer.

Low values may indicate that a developer often modifies files with which they have limited prior knowledge, whereas values close to zero may indicate that they work on files they know well.

11 Tree Map: Current risk score

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

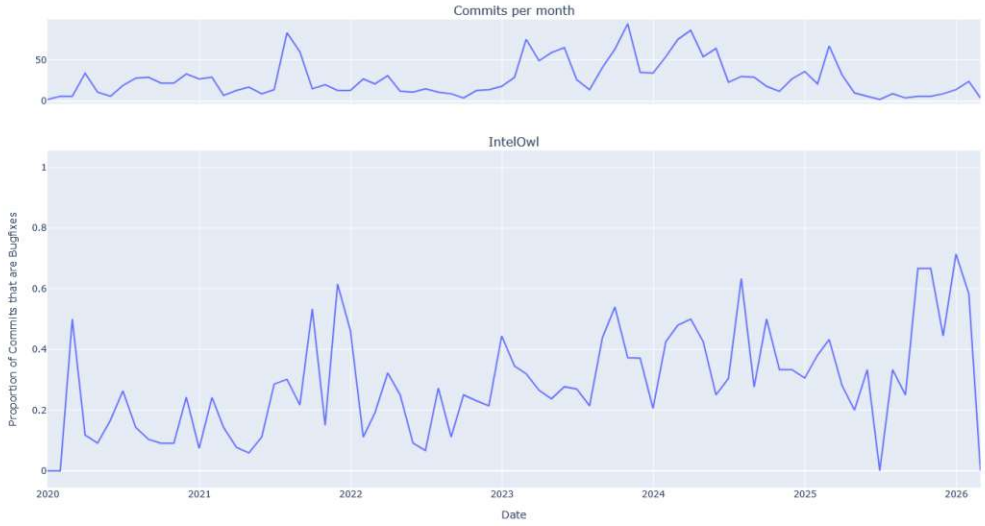

Figure 3.7: Proportion of commits that are bug-fixes per month

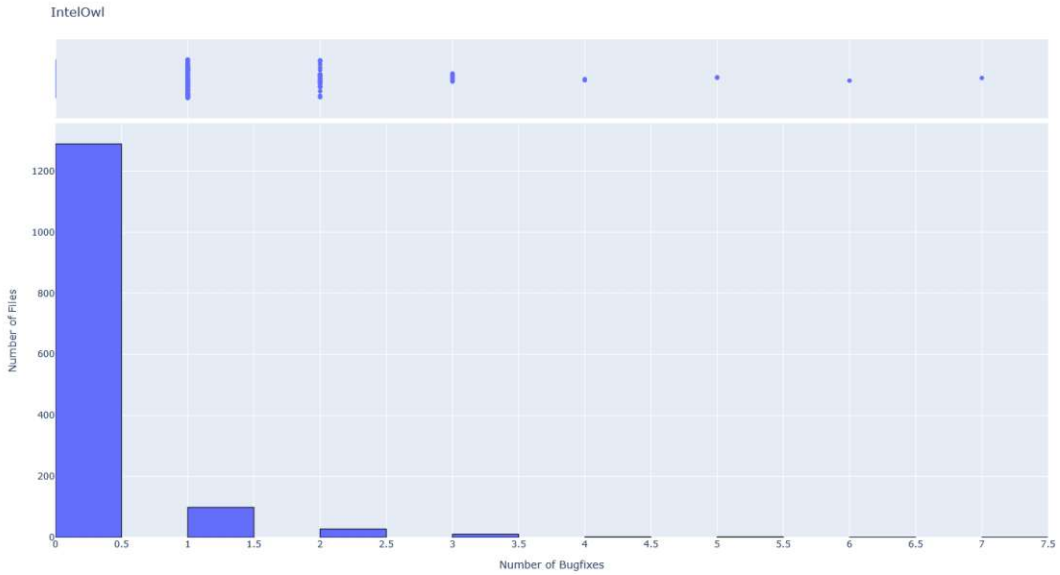

Figure 3.8: Bug-fixes a file was involved in last year

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

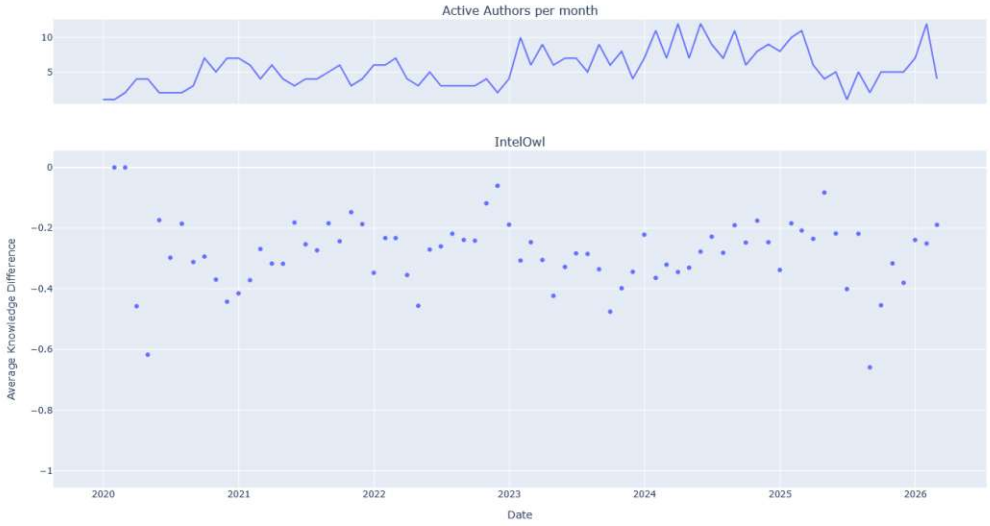

Figure 3.9: Average knowledge difference per month and number of developers per month

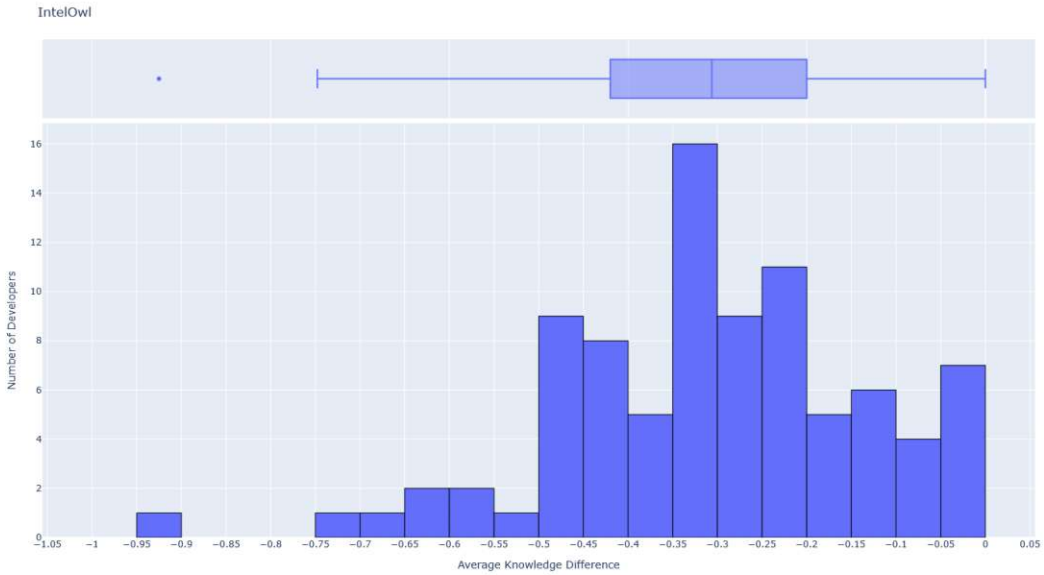

Figure 3.10: Average knowledge difference per developer

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

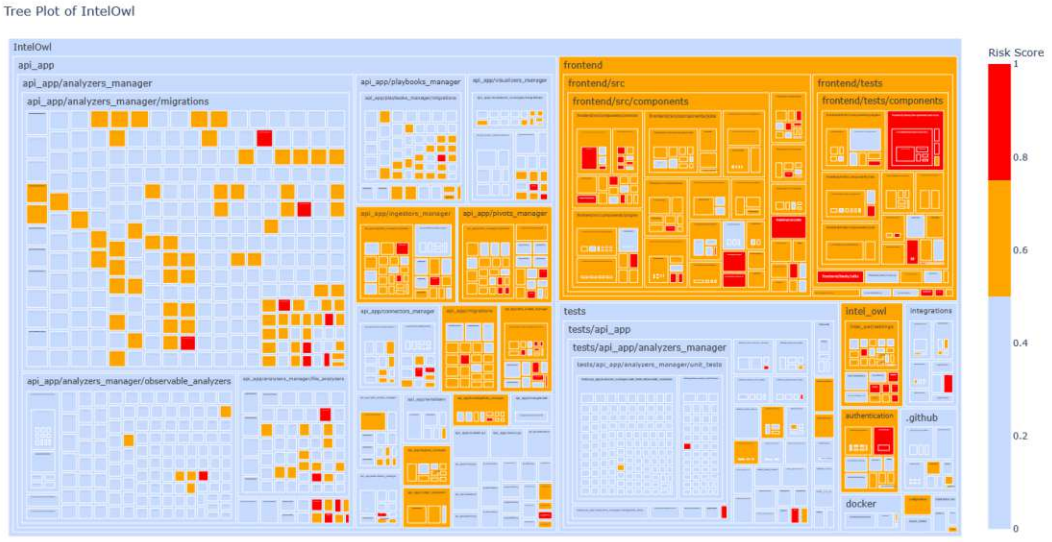

Figure 3.11: Current risk score

Figure B.11: The color of the treemap shows the file's current risk score, and the size of the rectangle shows how many lines of code the file has.

This visualization identifies files with a particularly high estimated risk of containing a defect.

12 Bus factor simulator

Figure B.12: The color of a rectangle shows the bus factor of the file. The size of a rectangle corresponds to the file's lines of code. In the interactive version, users can select developers to include or exclude, simulating the effect of individual developers leaving the team.

With the bus factor simulator, we can see which files have little knowledge and which files have a lot of knowledge, and how this changes if the team changes.

13 Relative bus factor simulator

Figure B.13: The color of a rectangle shows the relative bus factor of the selected person. The size still corresponds to the file's lines of code.

This visualization shows where someone has a lot of knowledge compared to the rest of the team. If the person were to leave the team, then the knowledge in these areas would be the most affected.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

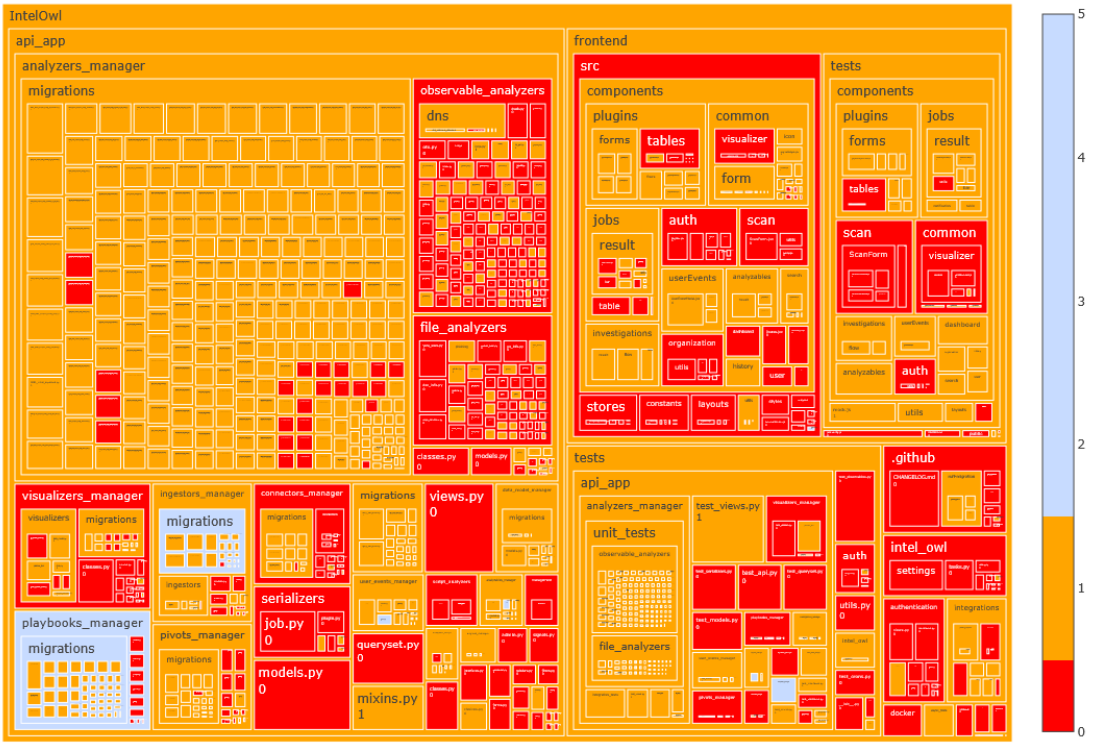

Figure 3.12: Bus factor simulator

3.4 Demonstration

In this step, the proposed solution is demonstrated through semi-structured expert interviews. This method was chosen because the target audience for the visualizations consists of domain professionals whose feedback is central to the study. Interviews allowed for direct feedback on the visualizations while also allowing follow-up questions and clarification during the conversation. A questionnaire was less suitable, as it would not have allowed the same level of interaction or the opportunity to explore the reasoning of the participants in greater depth. Compared to a more rigid format, interviews also made it easier to identify uncertainty, misunderstandings, and bias. Bias is discussed further in Chapter 4.

Task-based evaluation was also considered, but it was not well suited to the aims of this study. The goal was not to test whether participants could complete predefined tasks, but to explore how different stakeholders interpret the visualizations, what they consider useful, and how they might use the presented information in practice. For this reason, a semi-structured interview format was selected. It provided enough structure to ensure comparability across interviews while still allowing flexibility for follow-up questions and deeper exploration of stakeholder-specific perspectives.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

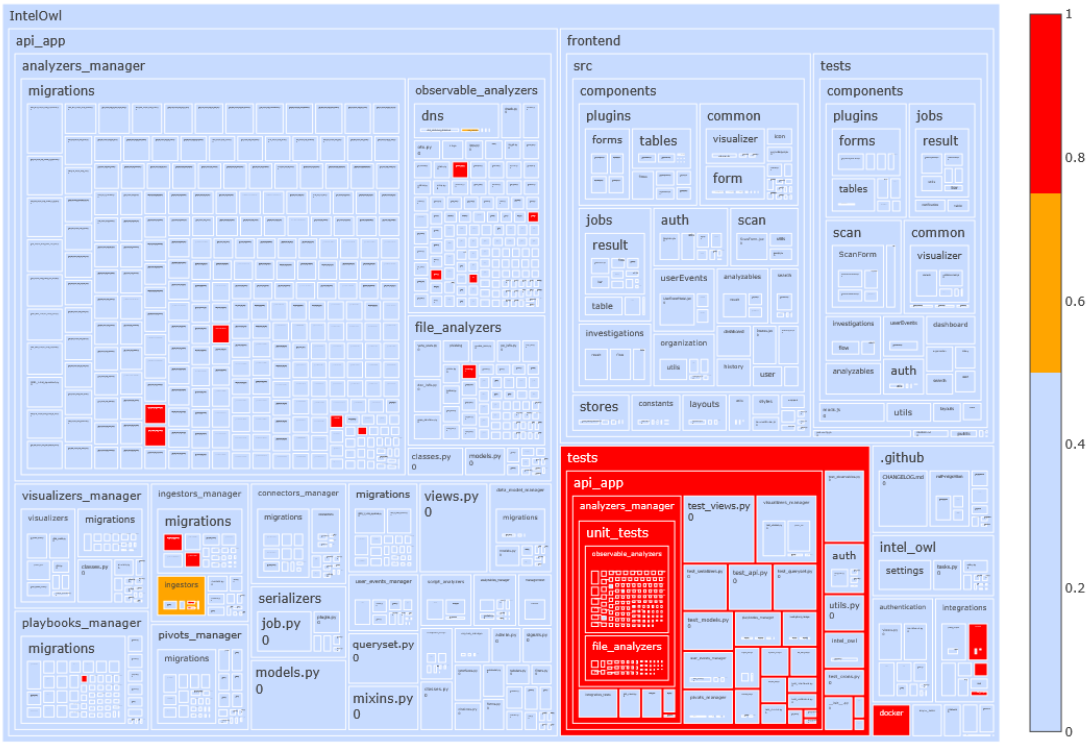

Figure 3.13: Relative bus factor simulator

We carried out three interviews per stakeholder group, resulting in nine interviews in total. Döring and Bortz [DB16] describe ten methodological steps for conducting such interviews, ranging from content-related preparation to archiving the material. For the purposes of this thesis, these steps are grouped into three phases: preparation, interview, and post-processing.

3.4.1 Preparation

The preparation phase consists of two steps, content-related preparation, and organizational preparation.

Content-related preparation included defining the topic and refining the research questions. The topic has already been defined within the design science process and is found in Section 3.1, and the research questions are defined in Chapter 1. In addition, criteria for selecting interview participants were defined, and the interview outline was developed and practiced in advance. The interview outline can be found in Appendix A.

To reduce bias and capture a range of perspectives, the aim was to recruit a heterogeneous group of participants with variations in background, age, sex, and professional experience. At least five years of work experience was required for participation. Organizational

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

preparation included identifying and contacting potential participants, arranging interview times and locations, and informing participants that audio recordings would be made. A practice interview was also conducted, and the final set of visualizations to be included in the interviews was selected. The resulting participant group was heterogeneous in several respects. Participants came from small, medium-sized, and large organizations, including non-profit contexts. The sample included different age groups ranging from their late twenties to late sixties and both male and female participants.

3.4.2 Interview

Following Döring and Bortz, the interview phase consists of five parts: beginning of the conversation, conducting and recording the interview, conclusion of the interview, farewell, and interview notes.

At the beginning of the conversation, the interviewer and participant met at the agreed location. The conversation began with a brief informal exchange, after which the participant was given time to read and sign the consent documents and review general information about the interview.

Conducting and recording the interview formed the main part of the session. The visualizations were presented one by one. The visualizations were randomized for each interviewee except for visualization 13, which always followed visualization 12 due to the relative bus factor being derived from the bus factor. We deemed this to be a more natural order, since we expected that explaining the relative bus factor before the bus factor could lead to additional confusion. For each visualization, the participant received contextual information and was asked questions about how they would interpret the visualization, how they might use the information in practice, and how easy it was for them to extract relevant information from it. Before moving on to the next visualization, participants rated both comprehensibility and task-relevance on a scale from 1 to 5. A rating of 1 indicated that the visualization was not at all comprehensible or not at all task-relevant, while a rating of 5 indicated that it was very easy to understand or highly task-relevant. Each of the nine participants provided two ratings for each of the thirteen visualizations. Most interviews lasted approximately 60 to 90 minutes.

At the step conclusion of the interview, there was sometimes a short informal closing conversation in which participants shared additional thoughts about the interview or the visualizations. This was followed by a farewell, in which the participant was thanked again for their time and the interview was formally concluded.

Afterwards, the interviewer completed the interview notes. These notes included the date, location, possible interruptions, and lessons learned.

3.4.3 Post-Processing

The post-processing phase comprises the steps transcription, analysis of the transcripts, and archiving the material.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

In the step transcription, an LLM was used to convert the audio recordings into an initial transcript draft. All participants were informed of this in advance. A complete transcript was not required as the primary basis of analysis, since detailed notes had already been taken during the interviews and quantitative ratings had been collected for each visualization.

In the step analysis of the transcripts, the transcripts and audio recordings were used as supporting material to recover additional qualitative observations that had not been captured in the notes. The main analysis relied primarily on the interview notes and the quantitative ratings, with transcripts used selectively where additional clarification was needed.

The final step, archiving the materials, concerned the handling and retention of the collected data. After the thesis has been completed and graded, the audio recordings and transcripts will be deleted. Only aggregated quantitative and qualitative results will be retained.

3.5 Evaluation

In the fifth step of the design science framework, the proposed solution is evaluated based on the quantitative and qualitative data collected during the interviews. This evaluation serves to answer the research questions defined earlier and is presented in detail in Chapter 4.

3.6 Communication

The final step of the design science framework is communication. This step is fulfilled through this thesis, which presents the research problem, the objectives, the artifact design, the demonstration, and the evaluation of the proposed solution.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Results and Discussion

This chapter presents and discusses the results of the semi-structured interviews described in Chapter 3. Based on the comprehensibility and task-relevance ratings provided by the interviewees, it evaluates which visualizations are effective for different stakeholder groups and examines differences between these groups.

4.1 Overview of Evaluation Data

We conducted semi-structured expert interviews, during which all nine interviewees rated all 13 visualizations in terms of comprehensibility and task-relevance. Each of the three stakeholder groups was represented by three participants. Further details on the interview design and participant selection are provided in Chapter 3. To aggregate the results, the median was used as it is less sensitive to outliers than the arithmetic mean.

4.2 Evaluation Criteria

A visualization was considered effective for a stakeholder group if it achieved a median score of at least 4 out of 5 on both task-relevance and comprehensibility. A score of 4 in task-relevance indicates that the visualization was perceived as clearly useful, whereas a score of 3 was interpreted as only somewhat helpful and therefore not sufficient for this classification. The same threshold was applied to comprehensibility, where a score of 4 indicates that the visualization was understandable despite only minor difficulties. A visualization was considered highly effective if it achieved the maximum median score of 5 out of 5 on both metrics.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.3 Overall Results and Discussion

The median comprehensibility scores per stakeholder group and visualization are shown in Table 4.1. The corresponding median task-relevance scores are shown in Table 4.2. One product role stakeholder assigned a task-relevance score of 1 to all visualizations. This participant works as a tester and stated that their work environment already defines when testing is required. As a result, they make few decisions based directly on the software quality of the project they are working on. Their task-relevance scores were therefore excluded from the task-relevance analysis, as this response indicates a flaw in the definition of the target audience rather than a weakness of the visualization. Their comprehensibility scores were kept, as the interviewee still evaluated how easy the visualizations were to understand.

The frequency of the median comprehensibility results is shown in Figure 4.1. Overall, the comprehensibility ratings were high. Technical leadership always had a median comprehensibility score of at least 4. Product roles gave a median score of 3 twice, while technical contributors gave a score of 3 only once. However, in most cases, the median comprehensibility scores were 4 or higher.

Figure 4.2 shows the corresponding plot for task-relevance. Compared to comprehensibility, the median task-relevance scores vary more strongly. This is expected since not every visualization needs to be equally useful to everyone. Nevertheless, several visualizations received median task-relevance scores of 4 or 5. Product roles perceived the highest task-relevance overall, while technical contributors perceived the least task-relevance, having given the highest number of low ratings. Technical leadership was positioned between the other two stakeholder groups, with the most median task-relevance scores of 3.

4.3.1 Effective Visualizations

One of the main objectives of this thesis was to identify effective visualizations. Table 4.3 summarizes the effectiveness of each visualization. Four visualizations have a median score of 5 out of 5 on both task-relevance and comprehensibility for at least one stakeholder group and are therefore considered highly effective. Six additional visualizations reached a median score of at least 4 out of 5 on both metrics for at least one stakeholder group and are therefore considered effective. The remaining three visualizations did not reach this threshold for any stakeholder group and are therefore classified as not effective.

This means that 10 out of the 13 visualizations were deemed useful to at least one stakeholder group. This suggests that most of the proposed visualizations are suitable for practical use when assigned to the stakeholder groups for which they were designed. In contrast, the three non-effective visualizations require further refinement before they can be recommended for use. In particular, their task-relevance should be improved, which may require changes to the information presented rather than only to the visual design.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Visualization ID	Technical Contributors	Product Roles	Technical Leadership
Figure B.1	5.0	5.0	5.0
Figure B.2	5.0	4.0	4.0
Figure B.3	5.0	5.0	5.0
Figure B.4	5.0	4.0	4.0
Figure B.5	5.0	4.0	5.0
Figure B.6	5.0	3.0	4.0
Figure B.7	5.0	4.0	5.0
Figure B.8	5.0	4.0	4.0
Figure B.9	3.0	3.0	4.0
Figure B.10	4.0	4.0	5.0
Figure B.11	5.0	5.0	5.0
Figure B.12	4.0	4.0	5.0
Figure B.13	4.0	5.0	5.0

Table 4.1: Median comprehensibility ratings by stakeholder group

Visualization ID	Technical Contributors	Product Roles	Technical Leadership
Figure B.1	4.0	5.0	4.0
Figure B.2	3.0	4.0	3.0
Figure B.3	3.0	4.5	3.0
Figure B.4	4.0	3.5	4.0
Figure B.5	4.0	4.0	3.0
Figure B.6	3.0	3.0	2.0
Figure B.7	1.0	3.5	3.0
Figure B.8	5.0	3.5	4.0
Figure B.9	2.0	4.0	4.0
Figure B.10	2.0	2.5	3.0
Figure B.11	4.0	5.0	5.0
Figure B.12	2.0	4.5	2.0
Figure B.13	2.0	5.0	5.0

Table 4.2: Median task-relevance ratings by stakeholder group excluding one product role stakeholder

Effectiveness	Visualizations
Highly effective	Figure B.1, B.8, B.11, B.13
Effective	Figure B.2, B.3, B.4, B.5, B.9, B.12
Not effective	Figure B.6, B.7, B.10

Table 4.3: Effectiveness classification of visualizations

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

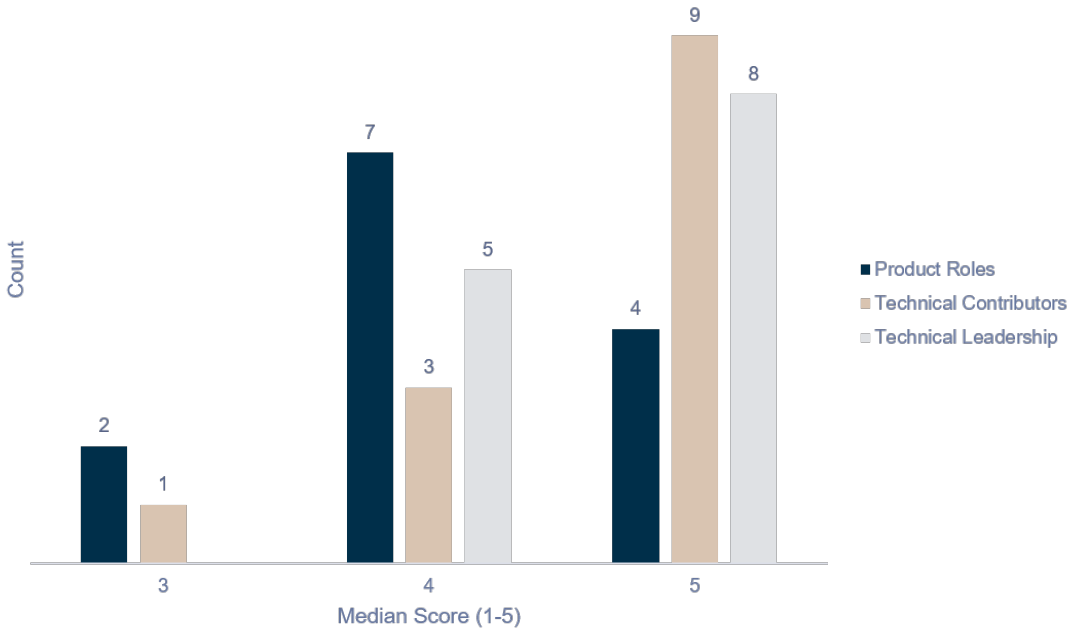

Figure 4.1: Median Comprehension Scores for all 13 Visualizations by Stakeholder Group

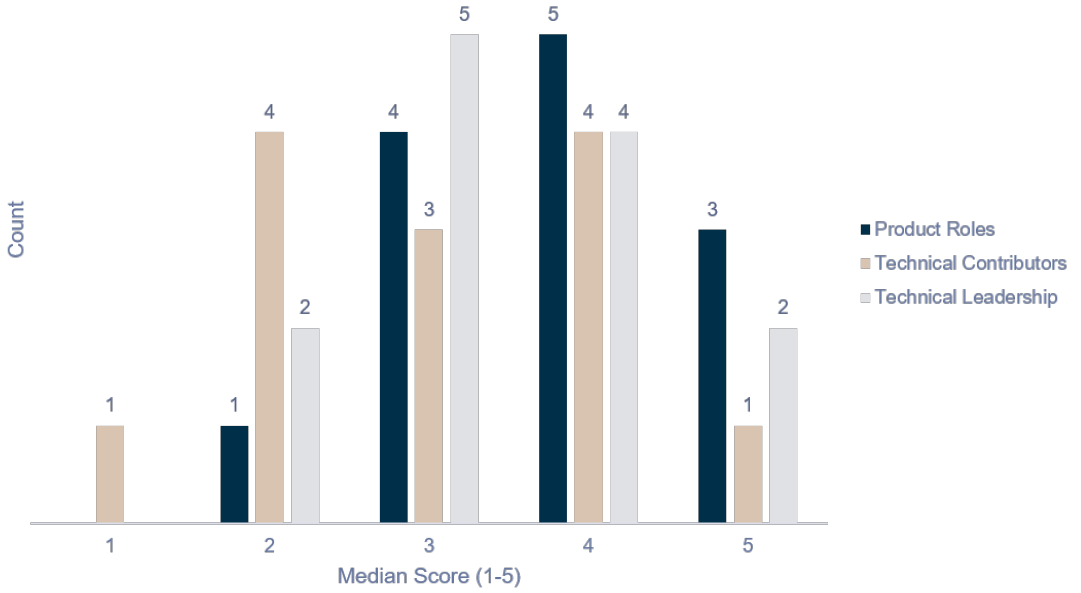

Figure 4.2: Median Task-Relevance Scores for all 13 Visualizations by Stakeholder Group

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.3.2 Differences in Comprehensibility Between Stakeholder Groups

Another objective of this thesis was to examine whether comprehensibility differs between stakeholder groups, in order to assess whether visualizations should vary not only in content, but also in design. As shown in Table 4.1, product role stakeholders gave the lowest comprehensibility ratings on average. With the exception of visualization 13, they gave either the lowest or shared the lowest score for every visualization. Technical leadership and technical contributors had the same average median comprehensibility score overall. However, apart from visualization 9, the visualizations for which they gave median scores below 5 differed between the two groups.

When comparing visualization types, all stakeholder groups had the most difficulty with the scatter plot. For product roles, tree maps were perceived to be easier to understand compared to histograms and line plots. During the interviews, several participants also showed confusion about the box plot that was included with the histogram. However, in some cases, the histogram itself also caused comprehension difficulties.

Overall, most visualizations are deemed comprehensible by all stakeholder groups. This suggests that the proposed visualizations are generally understandable and do not pose major comprehension barriers. However, the qualitative feedback indicates that design improvements are possible and could further improve the comprehensibility.

Plot Type	**Technical Contributors**	**Product Roles**	**Technical Leadership**
Histogram	5.0	4.0	5.0
Line Plot	5.0	4.0	5.0
Tree Map	5.0	4.5	5.0
Scatter Plot	3.0	3.0	4.0

Table 4.4: Median ratings by plot type and stakeholder group

4.3.3 Stakeholder-General and Stakeholder-Specific Visualizations

A further objective of this thesis was not only to identify visualizations that are useful across stakeholder groups, but also to design visualizations that address the needs of specific stakeholder groups. As shown in Table 4.5, this objective was achieved. Two visualizations were considered effective by all stakeholder groups, four were effective for two stakeholder groups, and four were effective for one stakeholder group. This indicates that the solution includes both stakeholder-general and stakeholder-specific visualizations.

Although two visualizations were useful across all stakeholder groups, most were only effective for a subset of stakeholders. This suggests that a single uniform set of visualizations is unlikely to meet the needs of all roles equally well. Instead, the findings support a stakeholder-aware approach in which some visualizations are shared across groups while others are tailored to specific stakeholder needs.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

In addition, several visualizations were not considered task-relevant by technical contributors. Since many established visualizations already exist for developers, there was less need to design new visualizations specifically for them. In this context, the fact that several visualizations were perceived as useful by other stakeholder groups can be seen as a positive result. It indicates that the proposed visualizations help address the existing gap identified in Chapter 2 of software quality metric visualizations for non-developer stakeholders.

Deemed Effective for	**Visualizations**
all stakeholder groups	Figure B.1, B.11
two stakeholder groups	Figure B.4, B.5, B.8, B.13
one stakeholder group	Figure B.2, B.3, B.9, B.12
no stakeholder group	Figure B.6, B.7, B.10

Table 4.5: Number of stakeholder groups a visualization is deemed effective for

4.4 Limitations

The results should be interpreted in light of several limitations. First, the sample size was relatively small. The aim of this thesis was to gather qualitative feedback from different stakeholder groups and explore how task-relevant and comprehensible they perceived the visualizations to be. It was not the goal to obtain statistically significant quantitative results. Consequently, these findings should be understood as indicative rather than statistically generalizable, even though they align with the qualitative feedback.

Another limitation concerns the number of visualizations included in the evaluation. During the design process, a larger set of candidate visualizations was considered. However, because the interview format required sufficient time for participants to understand, assess, and discuss each visualization in detail, the number of evaluated visualizations had to be limited. To keep the interviews feasible while still allowing for meaningful feedback, the study therefore focused on 13 visualizations in total.

A further limitation concerns the product role stakeholder who assigned a task-relevance score of 1 to all visualizations. Although this created a challenge for the quantitative analysis, the interview also provided an important insight into the intended audience of the visualizations. In addition to being interested in and able to influence software quality, stakeholders must also be in a position to act on the information provided. This interview showed that even stakeholders such as software testers, who are concerned about software quality and can influence it, may work within organizational structures in which they do not make major decisions based on software quality, but instead follow predefined processes. In such cases, the proposed visualizations may be understandable but not task-relevant. The findings therefore suggest that these visualizations are most appropriate for stakeholders who can use the information to support decision-making.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Finally, the evaluation process was subject to the risk of bias. Semi-structured expert interviews were chosen because they were well suited both to collect the required information and to reduce certain forms of bias. Nevertheless, three forms of bias were of particular concern: selection bias, courtesy bias, and social desirability bias. To reduce selection bias, efforts were made to recruit a diverse set of interviewees with different backgrounds, ages, sex, and from different types and sizes of organizations, including corporations and non-profits. Courtesy bias and social desirability bias were addressed during the interviews by probing interviewees when responses seemed overly agreeable. In this respect, the semi-structured interview format offered an advantage over a questionnaire, as it allowed follow-up questions and clarification where necessary.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Conclusion

Existing software quality metric visualizations primarily target developers. This thesis designed 43 software quality metric visualizations using a variety of software quality metrics, ranging from simple metrics such as lines of code to risk scores calculated by a machine-learning model. They were designed in an iterative process with the help of feedback from an industry partner. Thirteen of these visualizations were selected to be evaluated in nine semi-structured expert interviews. The experts belonged to one of three proposed stakeholder groups: technical contributors, product roles, or technical leadership. Each participant provided ratings of comprehensibility and task-relevance for all visualizations.

The findings answer the three research questions as follows. For RQ1, 10 of the 13 proposed visualizations were considered effective, based on median task-relevance and comprehensibility scores, for at least one stakeholder group. Regarding RQ2, the median comprehensibility scores differed between stakeholder groups. In particular, participants in product roles experienced the greatest difficulty in understanding the visualizations, as reflected by their lower median comprehensibility scores compared with technical contributors and technical leadership. Concerning RQ3, the results show that the proposed visualizations include both stakeholder-group-general and stakeholder-group-specific visualizations. Two visualizations were considered effective for all three stakeholder groups. Four visualizations were effective for two stakeholder groups. While four visualizations were effective for only one stakeholder group. The remaining three visualizations were not considered effective for any stakeholder group according to the defined criteria. Overall, the findings indicate that software quality and evolution metric visualizations can support stakeholders beyond developers.

This thesis provides software quality metric visualizations for different stakeholder groups, together with an evaluation based on semi-structured expert interviews. The results provide perceived ratings of task-relevance and comprehensibility. They offer insights into which types of visualizations are easier or more difficult to interpret for different

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

stakeholders, as well as which information stakeholders consider relevant to their tasks. The observed differences between stakeholder groups suggest that visualizations should be designed with their target stakeholders in mind, as the same representation may not be suitable for every stakeholder group.

Future work could incorporate the collected feedback and evaluate revised versions of the visualizations. It could also complement this exploratory interview study with a larger quantitative study, allowing the observed differences to be statistically tested. Finally, additional visualizations could be designed and evaluated specifically for non-developer stakeholders.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

APPENDIX A

Interview Outline

A.1 Einleitung

Hallo,

vielen Dank, dass du dir Zeit für dieses Interview nimmst.

Wir führen heute ein semistrukturiertes Interview durch. Zu Beginn werde ich dir ein paar kurze Fragen zu deiner Rolle, deinen Aufgaben und deinen Tätigkeiten in Softwareprojekten stellen.

Danach zeige ich dir zwölf Visualisierungen. Für jede Visualisierung bekommst du kurz Zeit, sie dir anzusehen. Anschließend werde ich dir einige Fragen dazu stellen.

Ziel des Interviews ist es, zu verstehen,

- wie verständlich diese Visualisierungen sind und
- wie relevant die dargestellten Informationen für deine Tätigkeiten in einem Softwareprojekt sind.

Für mich ist es besonders hilfreich, wenn du deine Gedanken während des Betrachtens laut aussprichst. Zum Beispiel, wenn dir etwas sofort klar ist, wenn dich etwas verwirrt, wenn du den Sinn einer Darstellung hinterfragst oder wenn dir etwas positiv auffällt.

Es gibt natürlich keine richtigen oder falschen Antworten, da nicht du getestet wirst, sondern die Visualisierungen. Wenn du etwas nicht verstehst, ist das also nicht deine Schuld, sondern entweder eine Visualisierung, die schwer zu verstehen ist, oder eine Visualisierung, die für andere Stakeholder gemacht ist.

Hast du Fragen?

Wurde die Datenschutzerklärung unterschrieben?

Super! Dann geht es los.

A.2 Fragen zur Person und Rolle

- Welche Rolle hast du aktuell in Softwareprojekten?
- Wie lange arbeitest du bereits in diesem Umfeld?
- Welche Aufgaben nimmst du typischerweise wahr?
- Welche Entscheidungen triffst du in Bezug auf Softwarequalität?
- Welche Informationen berücksichtigst du, wenn du diese Entscheidungen triffst?
- Welche Tools und Services verwendest du, um Softwarequalität zu messen?

A.3 Bewertung der Visualisierungen

Erste Visualisierung zeigen.

A.3.1 Verständlichkeit

Kernfragen

- Was glaubst du, zeigt diese Visualisierung?
- Welche Teile sind sofort verständlich?

Erweiterte Fragen

- Was ist die zentrale Aussage?
- Welche Elemente sind unklar?

A.3.2 Relevanz für die Tätigkeit

Kernfrage

- Ist diese Information für deine Rolle relevant?

Erweiterte Fragen

- Welche Entscheidungen könnte die Visualisierung unterstützen?

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

A.3.3 Eigene Bewertung

- Wie verständlich ist die Visualisierung für dich?
  (1 = gar nicht verständlich, 5 = sehr verständlich)
- Wie relevant ist diese Visualisierung für deine Arbeit?
  (1 = gar nicht relevant, 5 = sehr relevant)
- Für welche anderen Stakeholder glaubst du, dass diese Visualisierung relevant sein könnte?

Diese Fragen für jede der zwölf Visualisierungen wiederholen.

A.4 Abschlussfragen

Damit sind wir mit den Visualisierungen fertig. Noch ein paar Fragen zum Abschluss:

- Welche der gezeigten Visualisierungen waren für deine Rolle am relevantesten?
- Welche waren am verständlichsten?
- Welche würdest du im Arbeitsalltag tatsächlich verwenden?
- Welche Informationen fehlen dir in den Darstellungen?
- Gibt es sonst noch etwas, was du hinzufügen möchtest?

Damit sind wir fertig.

Hast du noch Fragen?

Nochmal vielen Dank fürs Mitmachen.

A.5 Optionale Nachfragen

Diese Fragen können bei Bedarf verwendet werden, um Antworten zu vertiefen:

- Kannst du das genauer erklären?
- Warum?
- Hast du ein Beispiel aus deinem Arbeitsalltag?
- Was genau ist daran unklar?
- Was müsste sich ändern, damit du die Visualisierung verwenden würdest?

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

APPENDIX B

Visualizations

Here we find the entire set of metrics used in this work and all visualizations that were created within this thesis. The description of the first 13 plots is identical to the description that can be found in Chapter 3, as these are the visualizations that are evaluated in the interview. We also included 17 additional visualizations that we deemed task-relevant and comprehensible but could not evaluate within this thesis, as well as 13 ideas that we rejected.

B.1 Metrics

These are the metrics used in the visualizations:

- Lines of Code: the number of lines of code that a file has.
- Lines of Code Modified: the number of lines of code that are written, replaced, or deleted in a file in a commit.
- Bug-fix: a commit that is made to fix a bug or defect in the code.
- Cognitive Complexity [Cam18]: a measure of how difficult code is to read. The value 25 was chosen as a threshold because SonarQube uses it as a default threshold.[1]
- Highly Complex Files: the number of files that have a cognitive complexity greater than or equal to 25.
- Number of Dependencies Updated: how many external libraries are updated.
- Degree of Knowledge [Car15]: a measure of how well a person knows a file, depending on what share of a file they have written and when they modified the file.

[1] SonarQube Threshold, accessed on 17 August 2026.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- Knowledge Difference: a measure derived from the degree of knowledge, showing how well the developer who wrote the code knows the code compared to the person who knows this file the best. A developer modifying a file that another developer just wrote results in a knowledge difference of -1. On the other hand, a developer who knows a file better than anyone else has a knowledge difference of 0 when they modify this file further.

- Bus Factor of a File: the number of people who would have to become unavailable at the same time for no one to still know that file well. Knowing a file well happens when a developer's degree of knowledge exceeds a threshold for this file.

- Bus Factor of a Folder: derived from the degree of knowledge, it is the number of people who would have to become unavailable at the same time for at least half of the files in that folder to no longer be known well by any developer.

- Relative Bus Factor: describes how large one person's share of knowledge is compared to the rest of the team:

  - 1: Only this person knows the file/folder.
  - 0.5: This person knows about as much as all others combined.
  - 0: This person does not know the file/folder at all.

- Risk Score: the output of a machine-learning model that estimates how likely it is that a file contains a bug.

- SZZ Bug Introductions: number of bugs introduced in a time period.

- hcpf: quantifies how complex and dispersed the recent change history of a file is.

B.2 Visualizations evaluated in the interviews

These are the visualizations presented to the interviewees:

B.2.1 1 Line Plot: Cumulative lines of code modified and cumulative bug-fixes over time

Figure B.1: The blue line shows how many lines of code have been changed up to that point (cumulative churn). The red line shows how many bug-fixes have been made up to that point (cumulative bug-fixes). It was chosen because it gives an overview of how much code is being written and how many bugs are fixed. Under the assumption that there is a constant ratio of bugs to the number of lines written, it can indicate whether bugs are being resolved at a rate that may prevent the accumulation of technical debt.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

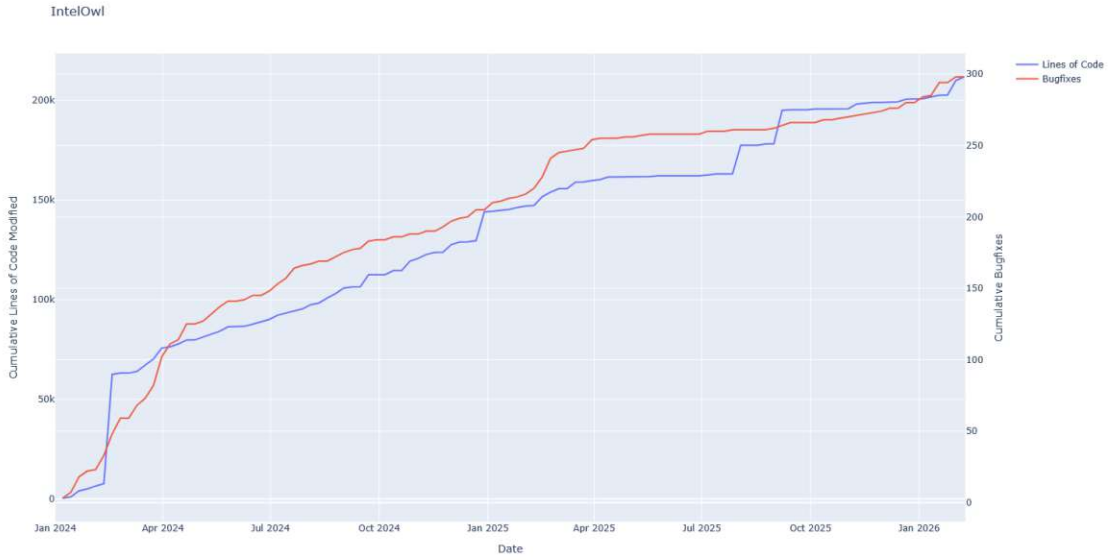

Figure B.1: Cumulative lines of code modified and cumulative bug-fixes over time

B.2.2 2 Tree Map: Lines of code modified last year

Figure B.2: Here, the color shows how many lines of code were modified, considering only changes from the last year. The size of the rectangle shows its current number of lines of code. This visualization gives an overview of a project, showing where work was done and which files are small or large.

B.2.3 3 Line Plot: Number of files that are highly complex that have been worked on per month

Figure B.3: This plot shows the number of files worked on in a month that have a cognitive complexity above 25. With this plot, one can see a trend in the number of highly complex files that are worked on. This shows whether complex files are often created or worked on or if it is a rare occurrence.

B.2.4 4 Histogram: Average cognitive complexity per developer

Figure B.4: This shows the average cognitive complexity of the files that a person has changed. An average above 20 is considered complex. This plot was chosen to show the cognitive complexity of the files modified by each developer.

B.2.5 5 Histogram: Current cognitive complexity of every file

Figure B.5: This shows the current cognitive complexity of each file. With this visualization, one can see how many files have what cognitive complexity. This gives an overview

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

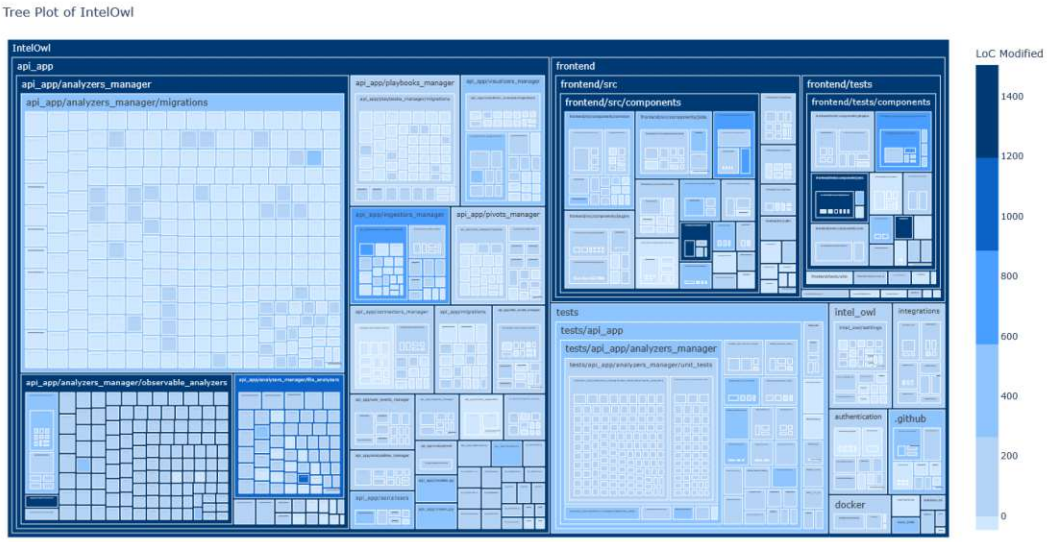

Figure B.2: Lines of code modified last year

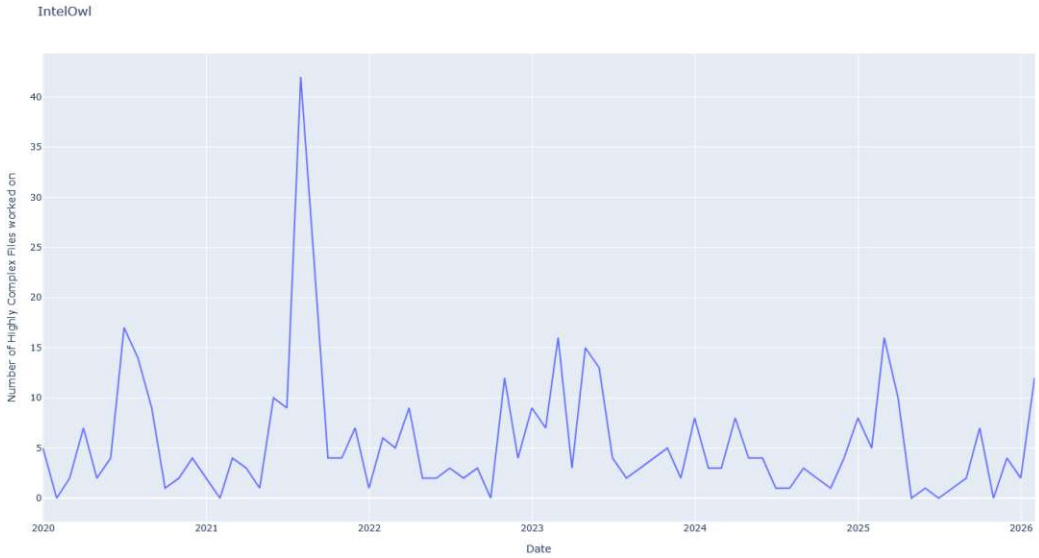

Figure B.3: Number of files that are highly complex that have been worked on per month

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

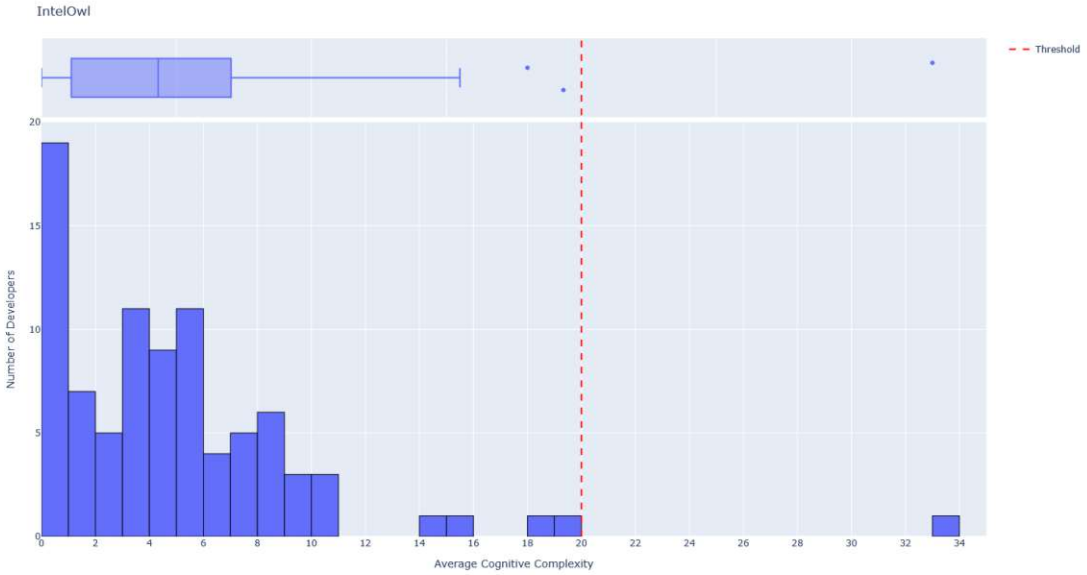

Figure B.4: Average cognitive complexity of edited files per developer

of the cognitive complexity of the entire project.

B.2.6 6 Line Plot: Number of dependencies updated in a given window

Figure B.6: This plot shows how often the dependencies are updated. Looking at this plot gives an overview of how frequently dependencies are updated.

B.2.7 7 Line Plot: Proportion of commits that are bug-fixes per month

Figure B.7: This view contains two line plots, one above the other. The lower plot shows the proportion of bug-fix commits among all commits. The upper plot shows how many commits there were at that time. It was chosen to compare the number of bug-fix commits with the number of feature commits.

B.2.8 8 Histogram: Number of bug-fixes that a file was involved in last year

Figure B.8: This shows how often a file was involved in a bug-fix commit during the last year. This visualization identifies which files were involved in the most bug-fix commits. These files may warrant investigation or refactoring because they appear to be bug-prone.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

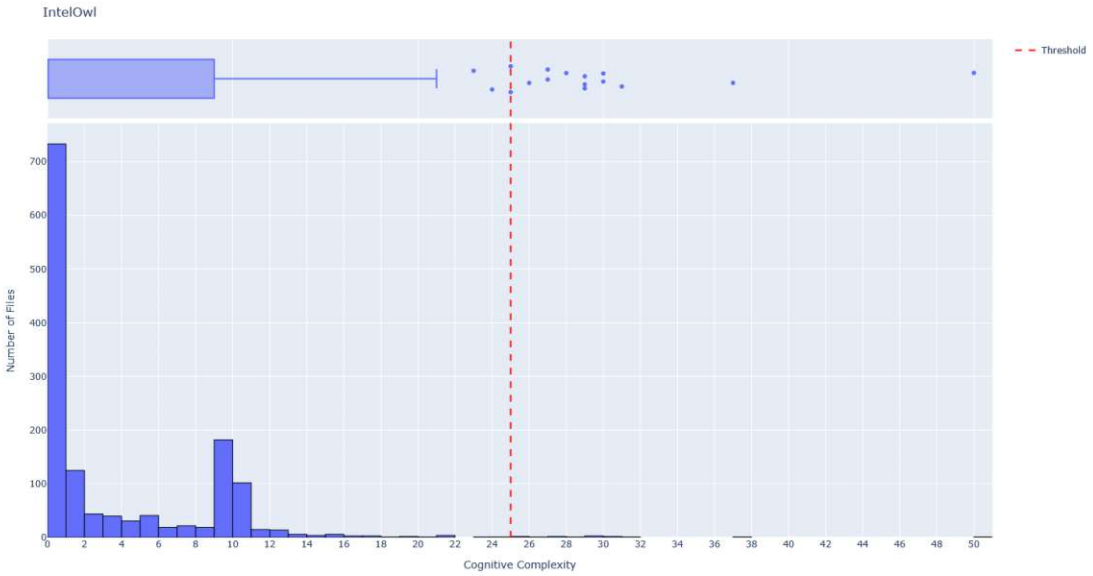

Figure B.5: Current cognitive complexity of every file

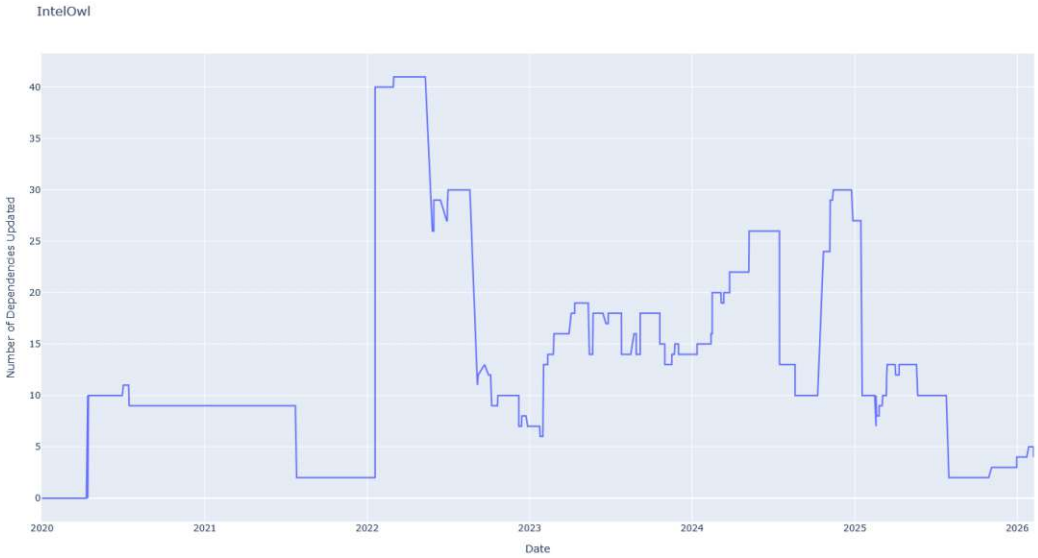

Figure B.6: Number of dependencies updated in a given window

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

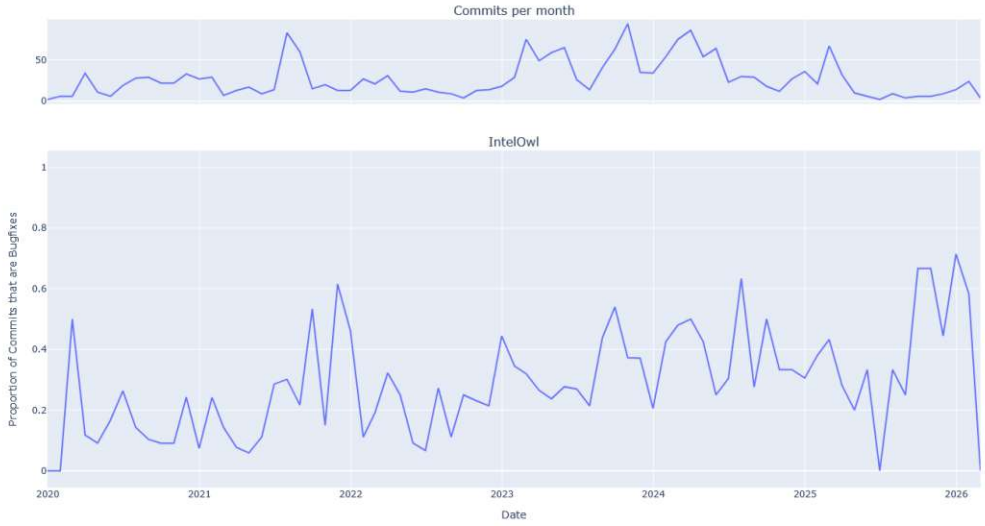

Figure B.7: Proportion of commits that are bug-fixes per month

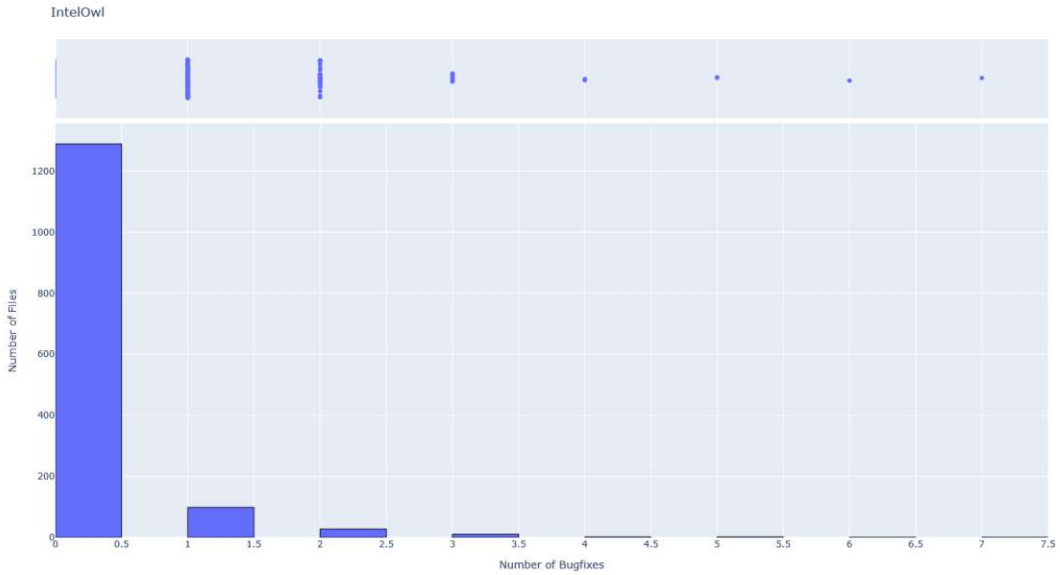

Figure B.8: Bug-fixes a file was involved in last year

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

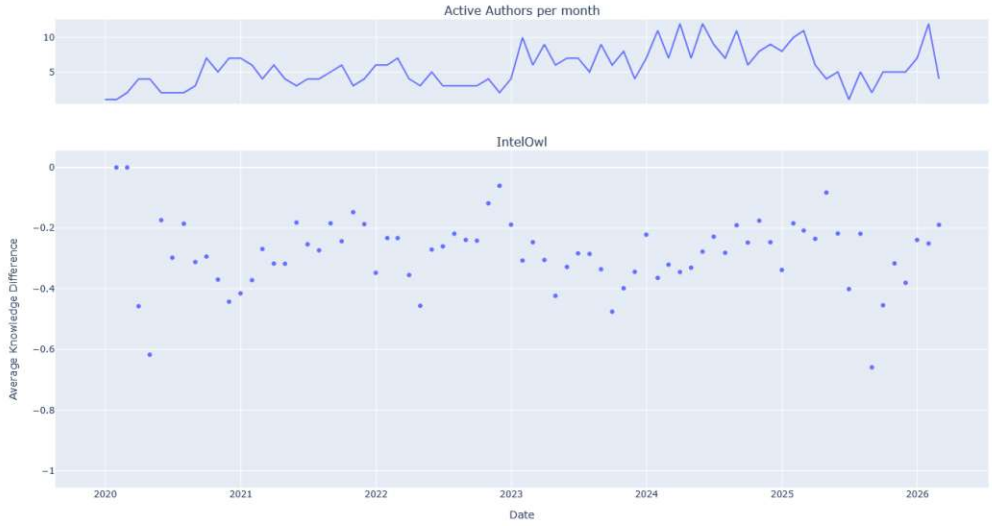

Figure B.9: Average knowledge difference per month and number of developers per month

B.2.9 9 Scatter Plot: Average knowledge difference per month and number of developers per month

Figure B.9: Lower points show the average monthly knowledge difference, while the upper line shows the number of developers who made a commit in that month. Knowledge difference can indicate whether developers typically modify files for which they have substantial prior knowledge or files primarily known by other developers.

B.2.10 10 Histogram: Average knowledge difference per developer

Figure B.10: This shows the average knowledge difference per developer. Low values may indicate that a developer often modifies files with which they have limited prior knowledge, whereas values close to zero may indicate that they work on files they know well.

B.2.11 11 Tree Map: Current risk score

Figure B.11: The color of the tree map shows the file's current risk score, and the size of the rectangle shows how many lines of code the file has. This visualization identifies files with a particularly high estimated risk of containing a defect.

B.2.12 12 Bus factor simulator

Figure B.12: The color of a rectangle shows the bus factor of the file. The size of a rectangle corresponds to the file's lines of code. In the interactive version, users can select

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

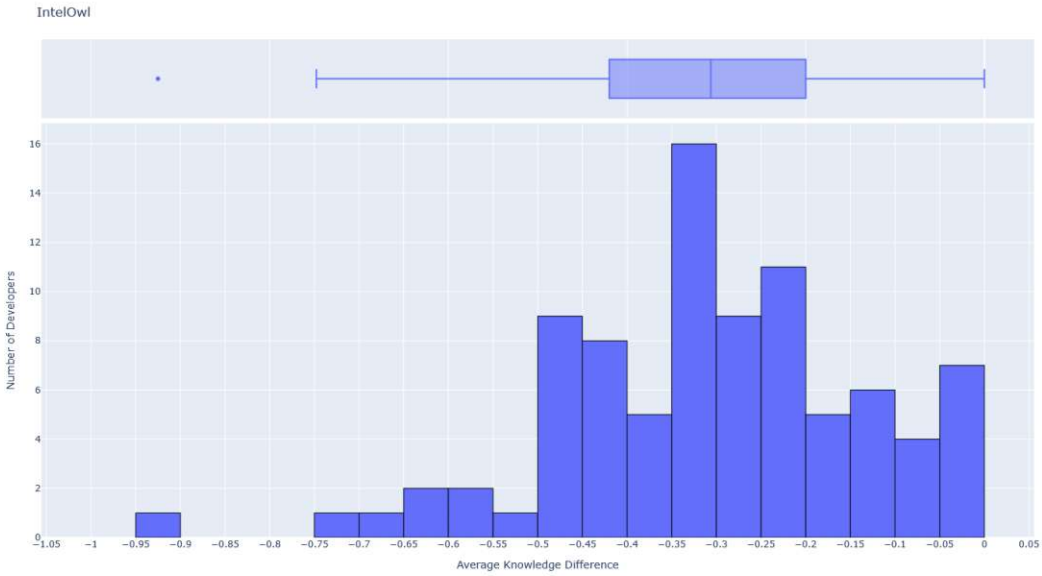

Figure B.10: Average knowledge difference per developer

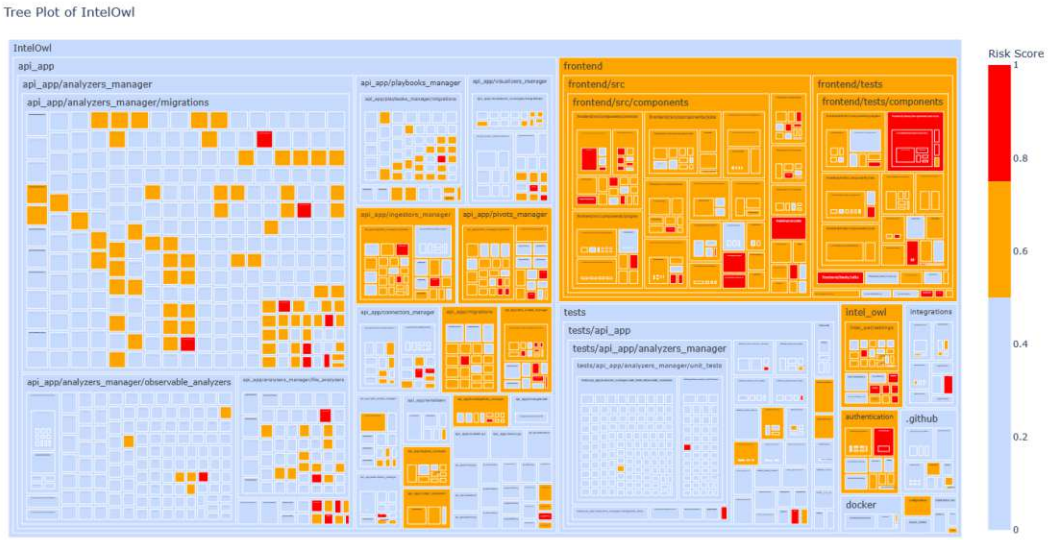

Figure B.11: Current risk score

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

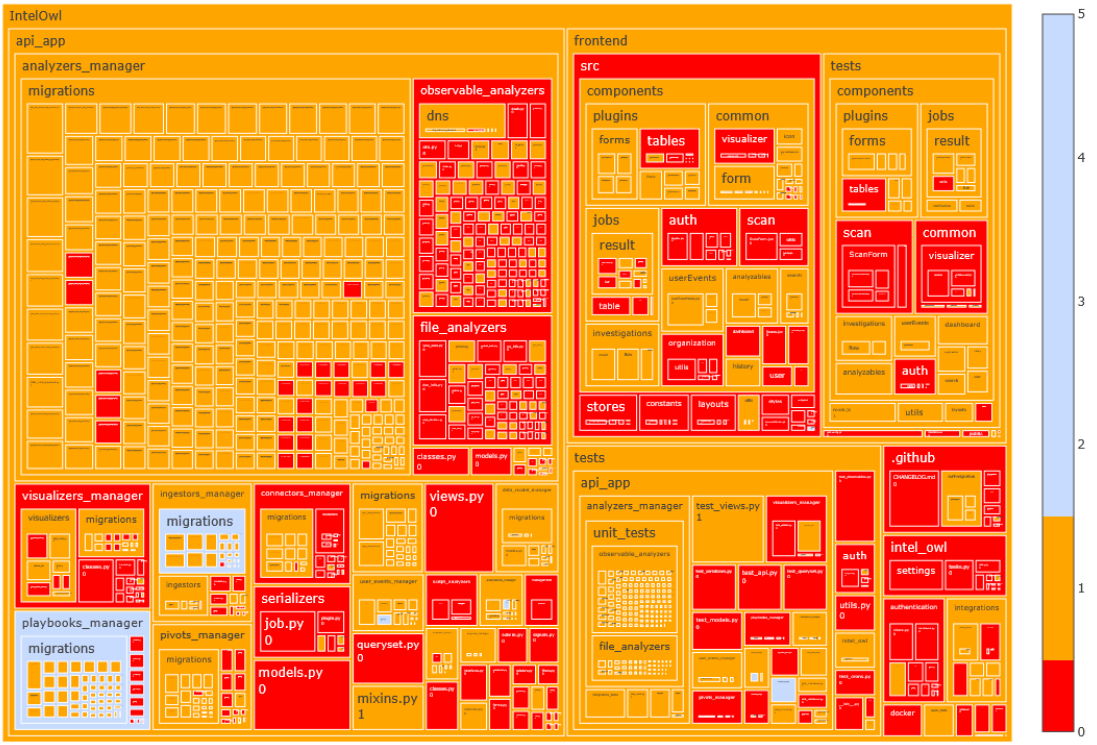

Figure B.12: Bus factor simulator

developers to include or exclude, simulating the effect of individual developers leaving the team. With the bus factor simulator, we can see which files have little knowledge and which files have a lot of knowledge, and how this changes if the team changes.

B.2.13 13 Relative bus factor simulator

Figure B.13: The color of a rectangle shows the relative bus factor of the selected person. The size still corresponds to the file's lines of code. This visualization shows where someone has a lot of knowledge compared to the rest of the team. If the person were to leave the team, then the knowledge in these areas would be the most affected.

B.3 Visualizations not evaluated in the interviews

B.3.1 14 Line Plot: Average lines modified per commit per month

Figure B.14: This plot shows how many lines of code an average commit includes in a given month. A value of 500 in April 2026 means that a commit in April 2026 modified an average of 500 lines of code.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

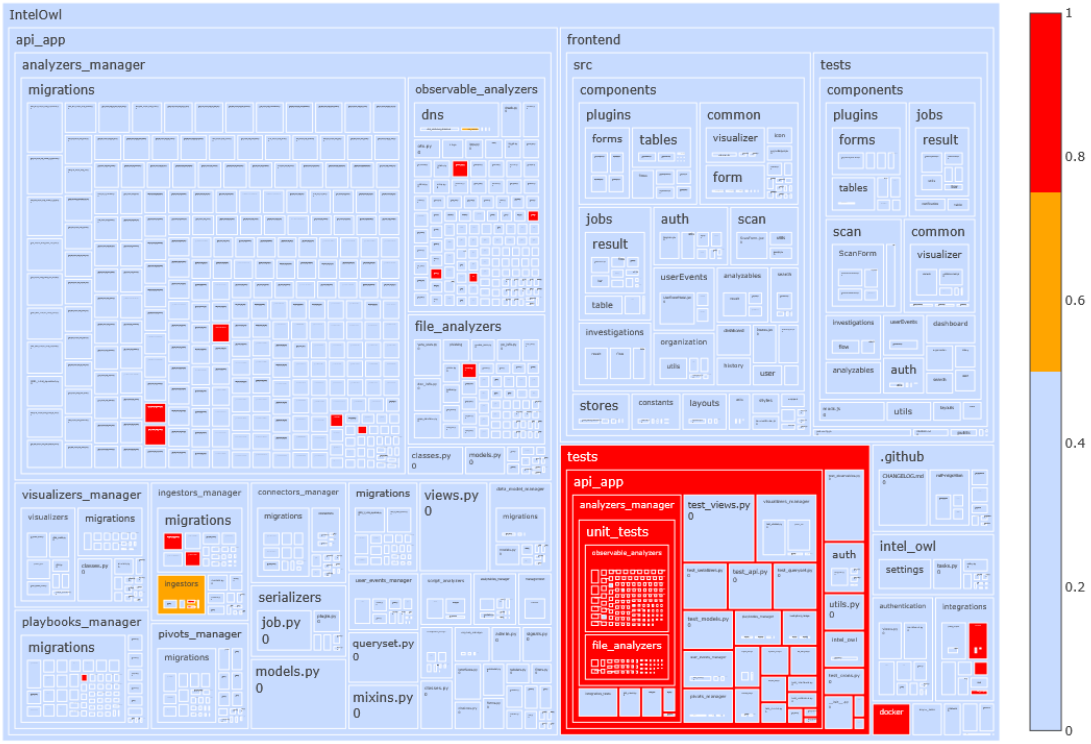

Figure B.13: Relative bus factor simulator

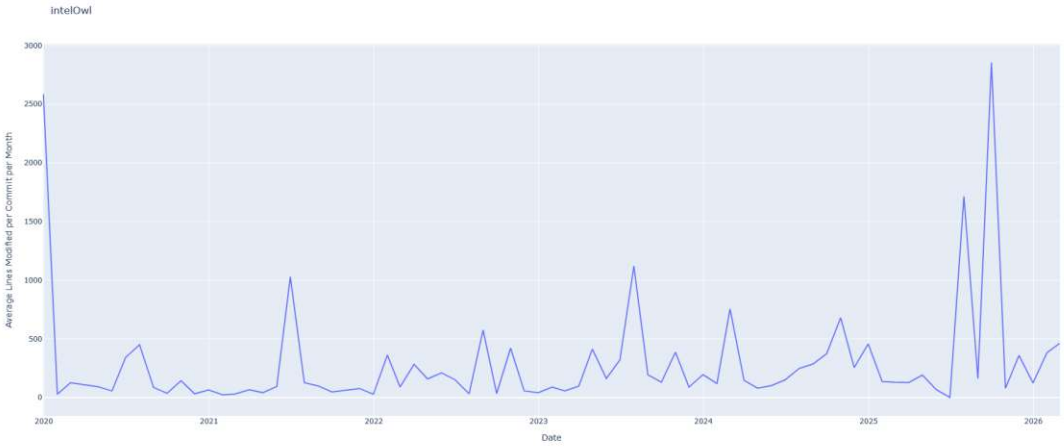

Figure B.14: Average lines modified per commit per month

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

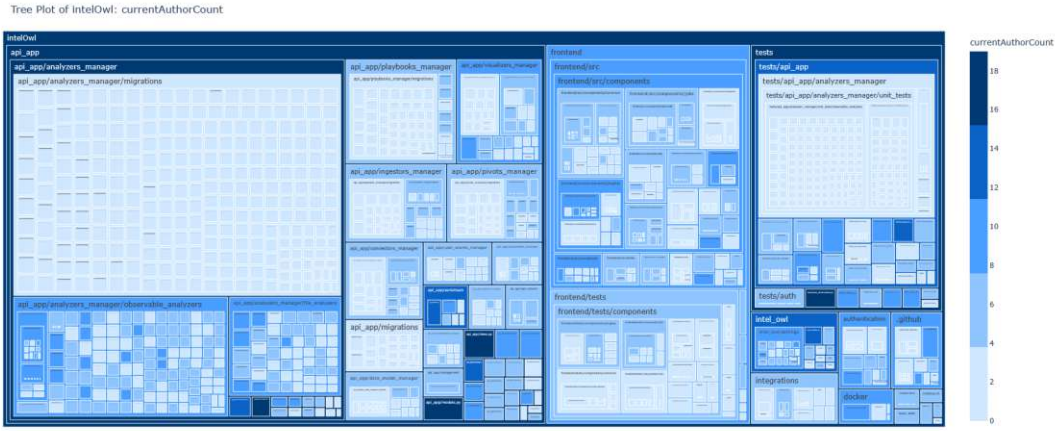

Figure B.15: Current number of authors per file

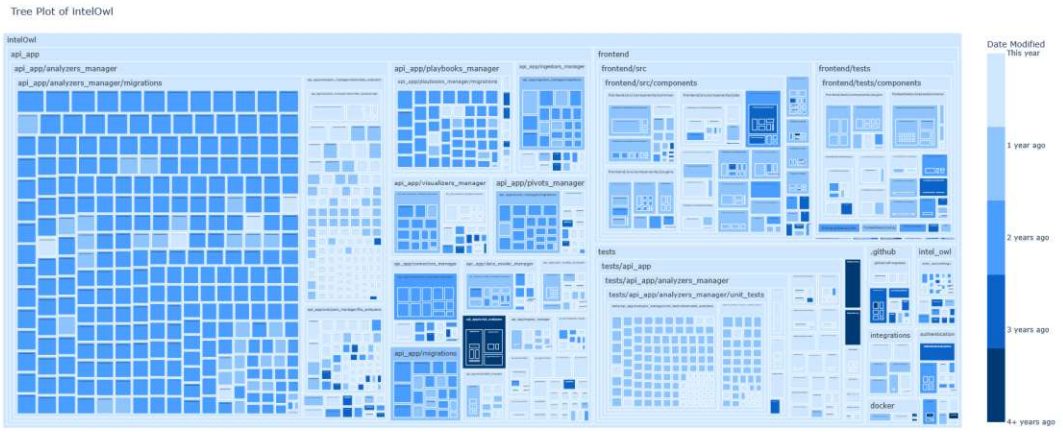

Figure B.16: Date and lines of code of most recent modification

B.3.2 15 Tree Map: Current number of authors per file

Figure B.15: This tree map shows how many people have worked on a file. The size of a rectangle shows the size of the file in lines of code. A small, dark rectangle means that a file has few lines of code but has been worked on by many developers.

B.3.3 16 Tree Map: Date and lines of code of most recent modification

Figure B.16: This tree map shows the date and lines of code of the latest modification for each file. The color indicates how old a file is, while the size represents how many lines of code were modified in the most recent changes to that file.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

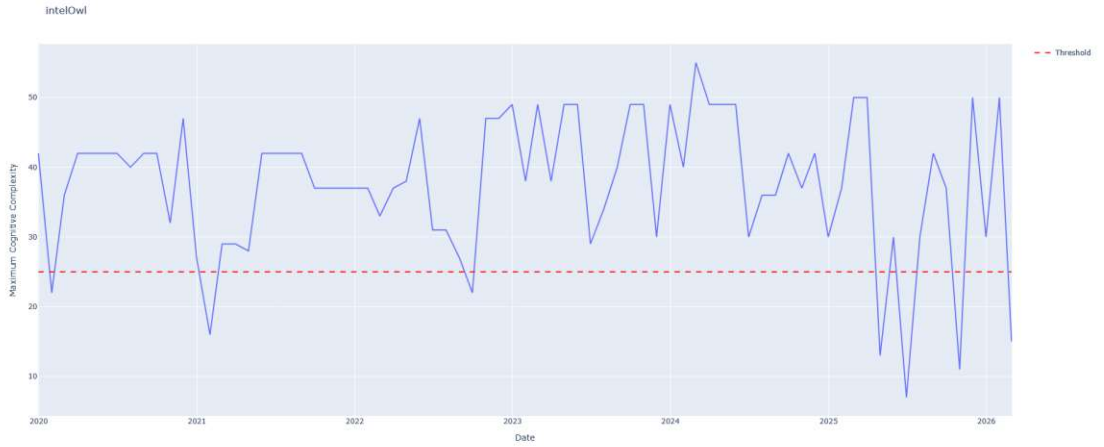

Figure B.17: Maximum cognitive complexity per month

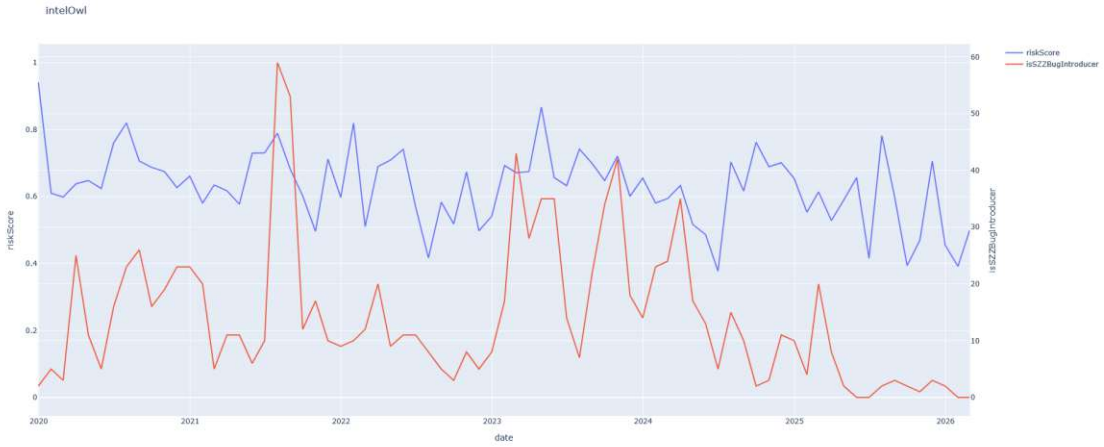

Figure B.18: Average monthly risk score and sum of SZZ bug introductions

B.3.4 17 Line Plot: Maximum cognitive complexity per month

Figure B.17: This plot shows the maximum cognitive complexity among all files modified in a given month.

B.3.5 18 Line Plot: Average monthly risk score and sum of SZZ bug introductions

Figure B.18: This plot shows two metrics per month. The blue line shows the average risk score of all modifications performed in a given month. The red line shows the number of bugs per month according to the SZZ bug introducer.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

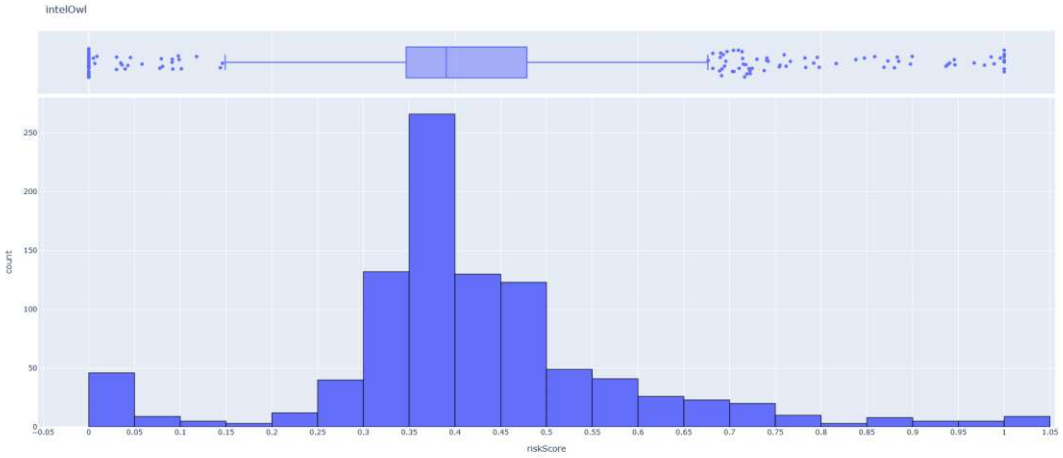

Figure B.19: Distribution of risk scores (of commits made last year)

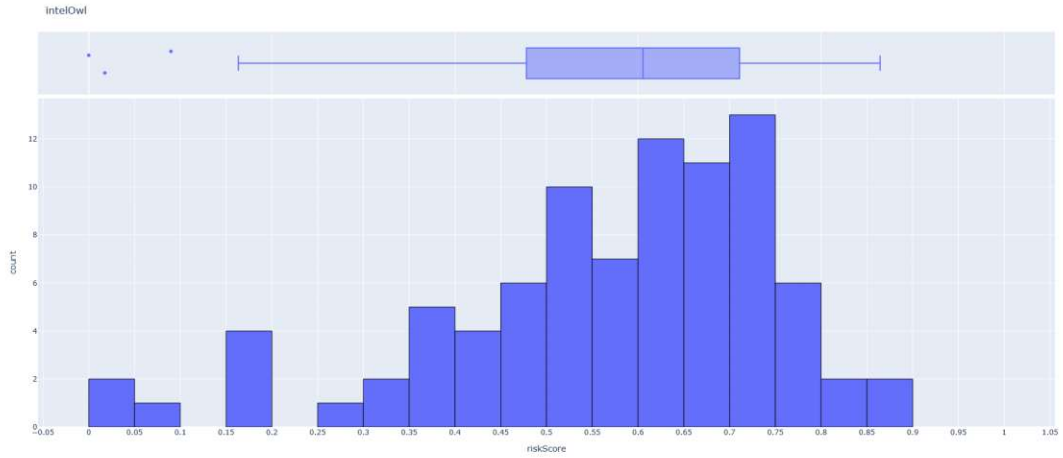

Figure B.20: Average risk scores per author

B.3.6 19 Histogram: Distribution of risk scores (of commits made last year)

Figure B.19: This histogram shows all risk scores in the last year.

B.3.7 20 Histogram: Average risk scores per author

Figure B.20: This plot shows the average risk score of all modified files per developer.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

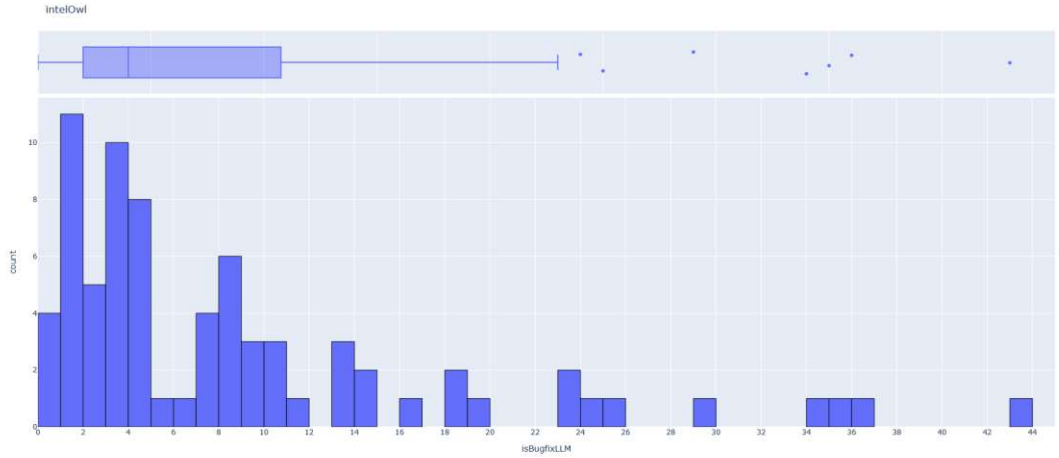

Figure B.21: Monthly bug-fixes histogram

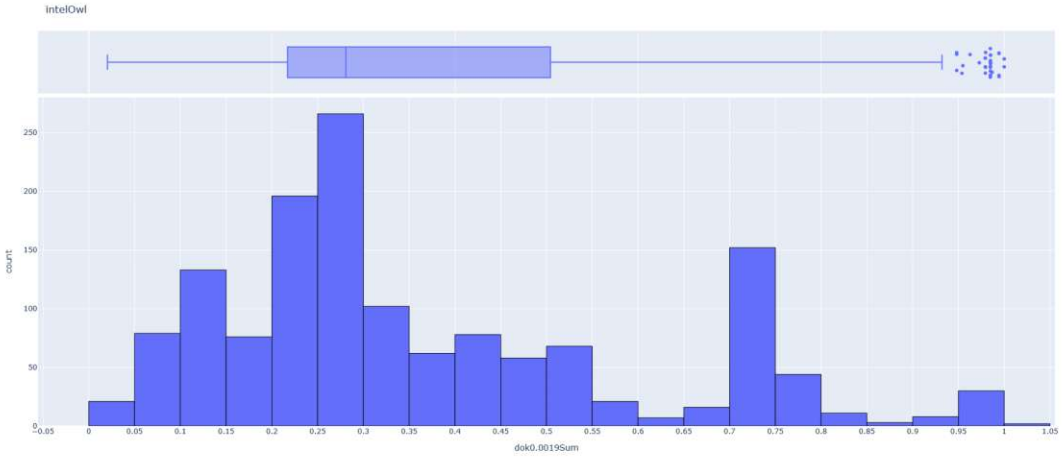

Figure B.22: Sum of the degree of knowledge per file

B.3.8 21 Histogram: Monthly bug-fixes histogram

Figure B.21: This histogram shows how often a particular number of bug-fixes occurs per month. A bar at 10 with a height of 5 means that there were 5 months in which exactly 10 bug-fixes occurred.

B.3.9 22 Histogram: Sum of the degree of knowledge per file

Figure B.22: This histogram shows the frequency of the sum of all degree of knowledge values per file.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

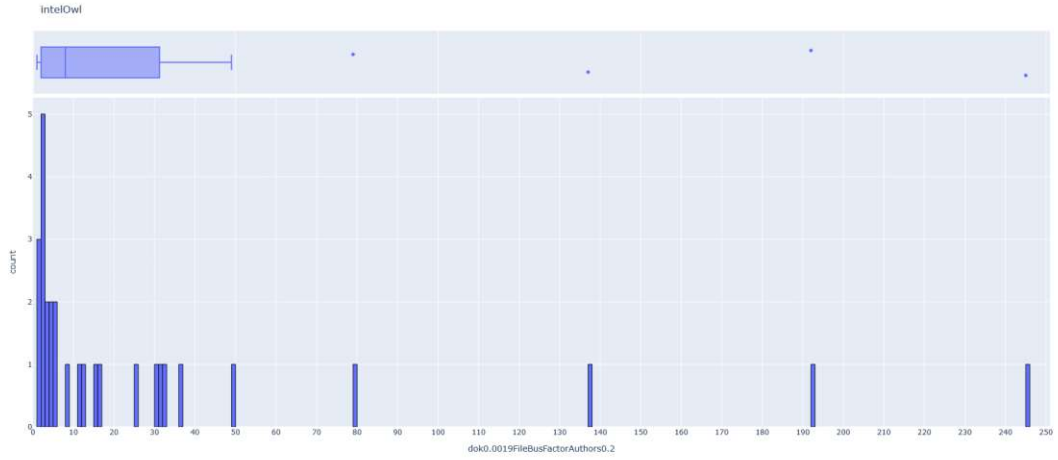

Figure B.23: Bus factor per author

B.3.10 23 Histogram: Bus factor per author

Figure B.23: This histogram shows, for each developer, how many files they have a bus factor in. A bar with a height of 2 at an x-value of 10 means that there are two developers who have a bus factor in 10 files. In other words, 10 files in which their degree of knowledge is ≥ 0.2, or 10 files that the person knows well.

B.3.11 24 Tree Map: Sum of degree of knowledge

Figure B.24: This tree map shows the degree of knowledge value of files, weighted by their size. The size of the rectangles corresponds to the current lines of code in the files. The color corresponds to the sum of the degree of knowledge values of all developers for a given file.

B.3.12 25 Line Plot: Total lines of code of the project over time

Figure B.25: This plot shows how the project size in lines of code has changed over time.

B.3.13 26 Line Plot: History of average degree of knowledge summed over all files

Figure B.26: This plot shows the average degree of knowledge value across all files at a given point in time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



Figure B.24: Sum of degree of knowledge

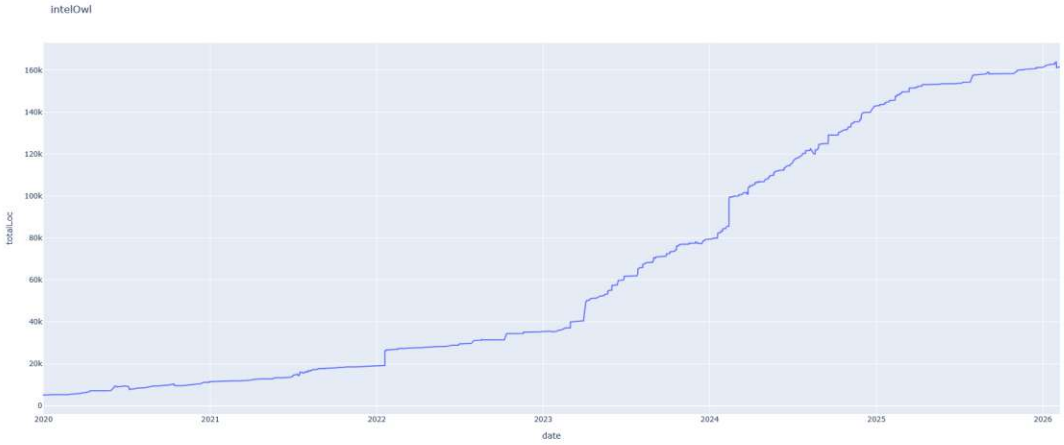

Figure B.25: Total lines of code of the project over time

B.3.14 27 Line Plot: Average degree of knowledge of current author per commit per month over time

Figure B.27: This plot shows the average knowledge a developer has of the code they modify per commit in a given month.

B.3.15 28 Histogram: Degree of knowledge of current committer

Figure B.28: This histogram shows how frequently a developer had a particular level of knowledge of a file when committing changes to it. A bar with a height of 100 in the range between 0.5 and 0.55 means that there were 100 cases in which a developer had knowledge between 0.5 and 0.55 when committing a file.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

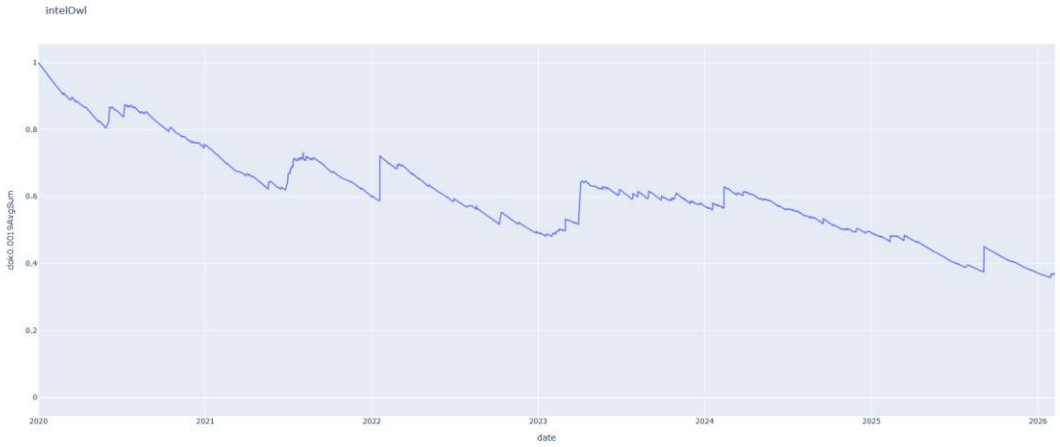

Figure B.26: History of average degree of knowledge summed over all files

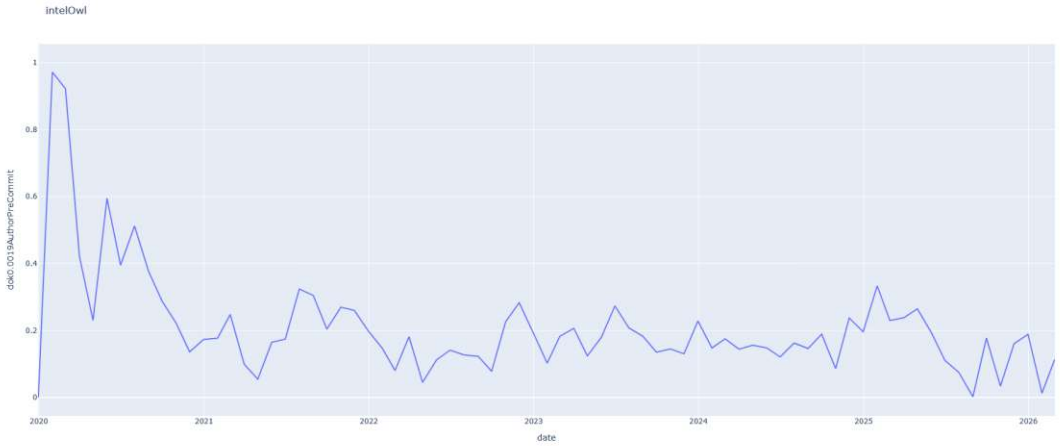

Figure B.27: Average degree of knowledge of current author per commit per month over time

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

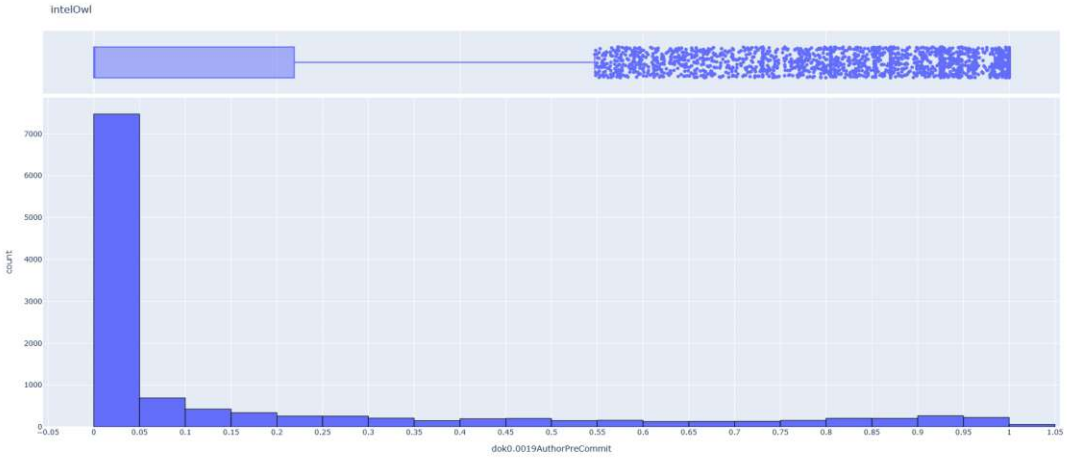

Figure B.28: Degree of knowledge of current committer

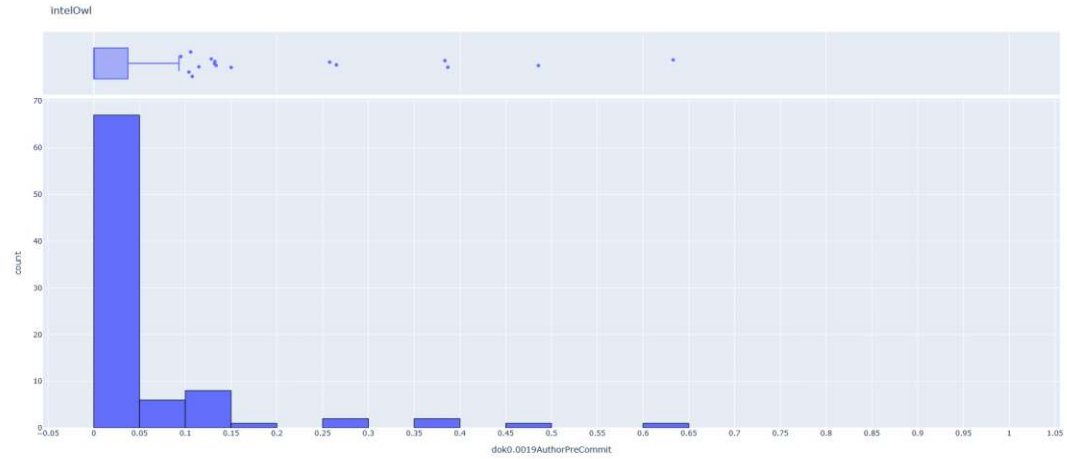

Figure B.29: Average degree of knowledge of current committer per author

B.3.16 29 Histogram: Average degree of knowledge of current committer per author

Figure B.29: This histogram shows the average knowledge developers have of the files they modified, aggregated across all developers.

B.3.17 30 Histogram: Average knowledge difference per commit

Figure B.30: This histogram shows the frequency of the knowledge difference per commit.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

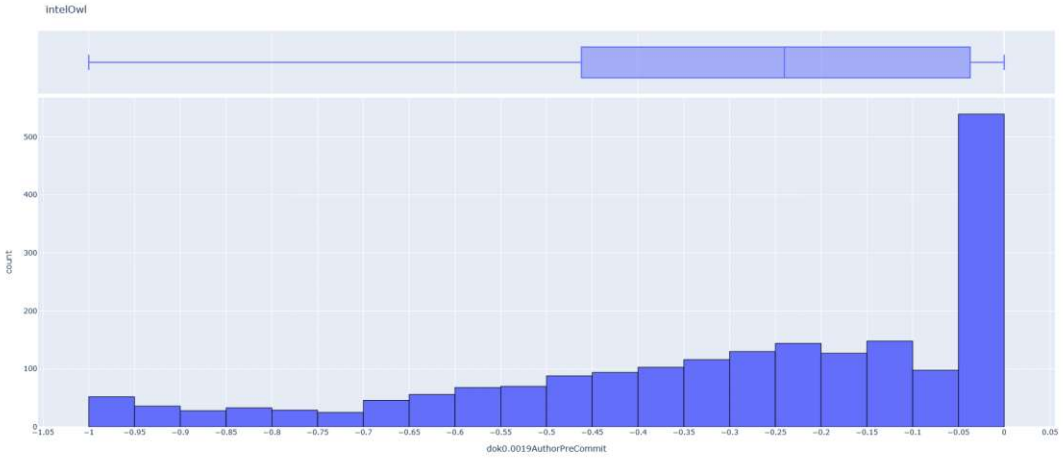

Figure B.30: Average knowledge difference per commit

B.4 Unsuccessful visualization ideas

The following section describes additional plot ideas that were discussed but rejected or adapted. The basic idea and the main problem are briefly documented for each.

B.4.1 Maximum monthly risk score

Idea:
Maximum monthly risk score in any visualization.
Problem:
The value is almost always close to 1 and it correlates too strongly with the number of commits. If the value is not close to 1, it is likely to be a month with very few commits.

B.4.2 hcpf

Idea:
Visualizations of hcpf.
Problem:
The metric is not task-relevant for any stakeholder and is difficult to interpret by humans.

B.4.3 Bug-fix lines of code compared with the number of bug-fixes

Idea:
Compare bug-fix lines of code with the number of bug-fixes in a line plot.
Problem:
The lines of code in bug-fixes are not particularly interesting.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

B.4.4 Bug-fix lines of code compared with total lines of code modified

Idea:
Compare bug-fix lines of code with the total modified lines of code.
Problem:
Not particularly interesting.

B.4.5 Bug-fixes per week

Idea:
Number of bug-fixes per week.
Problem:
Absolute values need to be put in relation to the modified code.

B.4.6 Number of bug-fixes per month and total lines of code

Idea:
Number of bug-fixes per month together with total lines of code.
Problem:
Total lines of code makes little sense here. Modified lines of code would have been more meaningful, and a cumulative representation would be better. Such a plot already exists.

B.4.7 Lines of code modified per week

Idea:
Modified lines of code per week.
Problem:
Uninteresting, since other variants of this visualization already exist.

B.4.8 Lines of code modified per week and number of bug-fixes

Idea:
Modified lines of code, cumulative or non-cumulative, per week together with the number of bug-fixes.
Problem:
Both values are easier to interpreted when shown cumulatively. This plot already exists.

B.4.9 Maximum cognitive complexity compared with lines of code modified

Idea:
Compare maximum cognitive complexity with modified lines of code.
Problem:
Visually too demanding, and no recognizable correlation is apparent.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

B.4.10 Histogram of lines modified in a commit per commit

Idea:
Histogram of modified lines of code per commit.
Problem:
The information gain is unclear. It is not obvious what concrete action should be taken based on this information.

B.4.11 Histogram of bus factor per file over two years

Idea:
Histogram of bus factor per file, compared over a period of two years.
Problem:
Historical bus-factor values are not deemed to be relevant.

B.4.12 Histogram of bus factor per file

Idea:
Histogram of bus factor per file.
Problem:
The Bus Factor Simulator is better.

B.4.13 Histogram of last commit date

Idea:
Histogram of the date of the last commit.
Problem:
The tree map is the more interesting representation for this information.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Overview of Generative AI Tools Used

gpt-5.X: Used to improve wording and structure of thesis and of interview questions, translating, transcription of audio, helped coding, helped refine methodology and research design, create latex tables, used to check work to prevent mistakes and discuss ideas. Additionally the abstract was translated to german.

Copilot: Used as an assistant for coding

Claude Opus 5: Assisted in writing the related work

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

3.1 Cumulative Lines of Code modified and cumulative Bug-fixes over Time . 15
3.2 Lines of code modified last year 15
3.3 Number of files that are highly complex that have been worked on per month 16
3.4 Average cognitive complexity of edited files per developer 17
3.5 Current cognitive complexity of every file 17
3.6 Number of dependencies updated in a given window 18
3.7 Proportion of commits that are bug-fixes per month 19
3.8 Bug-fixes a file was involved in last year 19
3.9 Average knowledge difference per month and number of developers per month 20
3.10 Average knowledge difference per developer 20
3.11 Current risk score 21
3.12 Bus factor simulator 22
3.13 Relative bus factor simulator 23

4.1 Median Comprehension Scores for all 13 Visualizations by Stakeholder Group 30
4.2 Median Task-Relevance Scores for all 13 Visualizations by Stakeholder Group 30

B.1 Cumulative lines of code modified and cumulative bug-fixes over time . . 43
B.2 Lines of code modified last year 44
B.3 Number of files that are highly complex that have been worked on per month 44
B.4 Average cognitive complexity of edited files per developer 45
B.5 Current cognitive complexity of every file 46
B.6 Number of dependencies updated in a given window 46
B.7 Proportion of commits that are bug-fixes per month 47
B.8 Bug-fixes a file was involved in last year 47
B.9 Average knowledge difference per month and number of developers per month 48
B.10 Average knowledge difference per developer 49
B.11 Current risk score 49
B.12 Bus factor simulator 50
B.13 Relative bus factor simulator 51
B.14 Average lines modified per commit per month 51
B.15 Current number of authors per file 52
B.16 Date and lines of code of most recent modification 52
B.17 Maximum cognitive complexity per month 53

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

B.18 Average monthly risk score and sum of SZZ bug introductions 53
B.19 Distribution of risk scores (of commits made last year) 54
B.20 Average risk scores per author 54
B.21 Monthly bug-fixes histogram 55
B.22 Sum of the degree of knowledge per file 55
B.23 Bus factor per author 56
B.24 Sum of degree of knowledge 57
B.25 Total lines of code of the project over time 57
B.26 History of average degree of knowledge summed over all files 58
B.27 Average degree of knowledge of current author per commit per month over time 58
B.28 Degree of knowledge of current committer 59
B.29 Average degree of knowledge of current committer per author 59
B.30 Average knowledge difference per commit 60

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

2.1 Frequency of target roles recorded by Chotisarn et al. [CMZ+20] across 105 papers published between 2013 and 2019. Papers may be assigned more than one role. 8

4.1 Median comprehensibility ratings by stakeholder group 29
4.2 Median task-relevance ratings by stakeholder group excluding one product role stakeholder 29
4.3 Effectiveness classification of visualizations 29
4.4 Median ratings by plot type and stakeholder group 31
4.5 Number of stakeholder groups a visualization is deemed effective for . . . 32

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[BG20] Paolo Boldi and Georgios Gousios. Fine-grained network analysis for modern software ecosystems. *ACM Transactions on Internet Technology (TOIT)*, 21(1):1–14, 2020.

[BZ12] Raymond PL Buse and Thomas Zimmermann. Information needs for software development analytics. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 987–996. IEEE, 2012.

[Cam18] G Ann Campbell. Cognitive complexity: An overview and evaluation. In *Proceedings of the 2018 international conference on technical debt*, pages 57–58, 2018.

[Car15] Patrick Eric Carlson. *Engaging developers in open source software projects: harnessing social and technical data mining to improve software development*. PhD thesis, Iowa State University, 2015.

[CMZ+20] Noptanit Chotisarn, Leonel Merino, Xu Zheng, Supaporn Lonapalawong, Tianye Zhang, Mingliang Xu, and Wei Chen. A systematic literature review of modern software visualization. *Journal of Visualization*, 23(4):539–558, 2020.

[DB16] Nicola Döring and Jürgen Bortz. *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften*. Springer, 5 edition, 2016.

[Fel25] Jodok Felder. Predicting bugs in source code: A machine learning approach for predicting faults by utilizing code and change metrics. Master's thesis, Technische Universität Wien, 2025.

[FSM+21] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. The space of developer productivity: There's more to it than you think. *Queue*, 19(1):20–48, 2021.

[GKSD05] Tudor Gîrba, Adrian Kuhn, Mauricio Seeberger, and Stéphane Ducasse. How developers drive software evolution. In *Eighth international workshop on principles of software evolution (IWPSE'05)*, pages 113–122. IEEE, 2005.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[GSN22] Michaela Greiler, Margaret-Anne Storey, and Abi Noda. An actionable framework for understanding and improving developer experience. *IEEE Transactions on Software Engineering*, 49(4):1411–1425, 2022.

[Hin96] David S Hinley. Software evolution management: A process-oriented perspective. *Information and Software Technology*, 38(11):723–730, 1996.

[HP22] Philipp Haindl and Reinhold Plösch. Value-oriented quality metrics in software development: Practical relevance from a software engineering perspective. *IET Software*, 16(2):167–184, 2022.

[Kan03] Stephen H Kan. *Metrics and models in software quality engineering*. Addison-Wesley Professional, 2003.

[Kra22] Herb Krasner. The cost of poor software quality in the US: A 2022 report. Technical report, Consortium for Information and Software Quality (CISQ), 2022.

[Leh80] Meir M Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980.

[MGAN18] Leonel Merino, Mohammad Ghafari, Craig Anslow, and Oscar Nierstrasz. A systematic literature review of software visualization evaluation. *Journal of Systems and Software*, 144:165–180, 2018.

[Mun09] Tamara Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009.

[Pre10] Roger S Pressman. *Software Engineering: A practitioner's approach*. McGraw-Hill, 7 edition, 2010.

[PTRC07] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007.

[RD13] Foyzur Rahman and Premkumar Devanbu. How, and why, process metrics are better. In *2013 35th international conference on software engineering (ICSE)*, pages 432–441. IEEE, 2013.

[SCB+18] Alper Sarikaya, Michael Correll, Lyn Bartram, Melanie Tory, and Danyel Fisher. What do we talk about when we talk about dashboards? *IEEE transactions on visualization and computer graphics*, 25(1):682–692, 2018.

[TS10] Christoph Treude and Margaret-Anne Storey. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 365–374, 2010.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN