Surveying inner source adoption in IKEA

By Alexander Malm

Supervisor: Christin Lindholm Examinator: Christian Nyberg

©Alexander Malm, 2023

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Abstract

Open source has lately been gaining more traction and recognition in the software industry. The practices are being implemented within various organisations through an approach known as inner source, which involves taking the principles and practices that have been proven effective in open source environments and applying them within an internal context.

This thesis focuses on an exploration of IKEA's current software development processes, aiming for assessing the potential of inner source adoption by surveying several developer teams.

Utilizing the Goal-Question-Metric (GQM) methodology, critical performance metrics were identified and used to gather the team’s development metadata from their GitHub code repositories. This data was then analysed and assessed in combination with the current best-practices of inner source development.

Recommendations based on the findings include ensuring comprehensive documentation, fine-tuning testing practices and adherence to test results, ensuring pull request management processes and increasing project visibility within the organisation. Addressing these aspects could help facilitate the adoption of inner source practices in IKEA, bringing benefits not only to the developer teams but to the entire organisation.

Keywords: Inner source, GitHub, GrimorieLab, metrics, KPI, repository analysis

Sammanfattning

Open source-baserad utveckling har senaste åren mött ökat intresse och erkännande inom mjukvaruindustrin. Dessa metoder anpassas och implementeras inom olika organisationer, så kallad inner source, vilket innebär att principer och metoder från open source-utveckling tillämpas inom en intern kontext.

Detta examensarbete fokuserar på en granskning av IKEAs nuvarande mjukvaruutvecklingsprocesser genom att undersöka flera utvecklarteam, med målet att bedöma möjligheten för utveckling med inner source-metodik

Genom att använda Goal-Question-Metric (GQM)-metodik identifieras kritiska mätvärden som sedan användes för att samla in teamens metadata från deras källkod på GitHub. Denna data analyserades och jämfördes med de nuvarande bästa praxis för inner source-utveckling efter resultat från litteraturgranskning.

Resultaten leder till en rekommendation för vad IKEA ska ha i åtanke vid fortsatta inner source-initiativ: säkerställa dokumentation, finjustering av testrutiner och acceptanskriterier, förbättrad hanteringen av pull request-processer i delar av utvecklingen, samt ökat synliggörande av teamens produkter och dess tekniska lösningar. Med dessa rekommendationer i åtanke har implementering av inner source-utveckling på IKEA större chans att lyckas, och kan medföra fördelar för utvecklingsteamen och också för hela organisationen.

Nyckelord: Inner source, GitHub, GrimorieLab, mätvärde, repository analys

Acknowledgments

Without the invaluable assistance and direction provided by my supervisors at IKEA, Oihsik Sarkar and Supriya Chatale, along with the collective efforts of everyone at IKEA who lent a hand throughout the spring, this bachelor’s thesis would not have been possible. I also want to thank my supervisor at LTH, Christin Lindholm for providing critical input and assistance writing the thesis.

Alexander Malm

Contents

1. Introduction

This chapters describes the goals and purpose of the bachelor's thesis work, as well as the necessary background information needed to understand the scope and delimitations.

1.1. Background

IKEA is perhaps mostly known for their furniture, warehouses and Swedish meatballs. A large organisation with stores all over the world. The technical aspects of such a global organisation require a significant investment of time and resources and are an essential element to the organisation's success in a quickly evolving digital landscape [1].

IKEA consists of numerous subsidiary companies. The focus of this thesis work is on INKA, which is the largest of these subsidiaries owning a majority of the warehouse locations around the world. In this thesis work, the term "IKEA" will be used to refer to INKA, in accordance with the company's practice of using the parent brand name when further distinction is not necessary.

In large organisations such as IKEA, many teams of developers might simultaneously be trying to solve the same problems, unknowingly of other's efforts. To handle this issue, IKEA has initiatives to promote internal sharing and contribution of the software development process. This practice is often called inner source and can be described as the application of open source development practices within a local context [2][3]. Common practices in open source development include collaboration, transparency, peer review of code, and code reuse, providing benefits such as increased code quality, faster time-to-market, and improved knowledge sharing across teams [4].

This thesis work will to some extent be conducted in collaboration with Mandana Khasayar from Blekinge Tekniska Högskola. The overreaching purpose will be to examine software development teams in the context of inner source but from two distinct viewpoints, presented in one bachelor's and one master's thesis. The work presented in this thesis involves the strategy of identifying key development metrics, obtaining GitHub repository metadata that corresponds to these metrics, and analysing the data. The goal is to comprehend the team’s current development practices and ultimately assess their potential for adopting inner source methodologies. Meanwhile, the other work will evaluate the teams' maturity and readiness for inner source by assessing their team environment, culture, and attitudes, and present the result in another thesis work.

The outcomes of these studies will offer distinct assessments of the overall state of several IKEA software development teams, using two different methods. Each will present recommendations based on our unique findings, proposing potential strategies that IKEA may contemplate if they decide to pursue inner source initiatives in the future.

1.2. Purpose

This thesis work will survey the current practices and priorities of a few developer teams within IKEA, using identified key metrics for software development goals regarding inner source. The results from the team’s data will be analysed in order to propose recommendations for IKEA’s continued inner source initiatives.

1.3. Goals

The goal of this thesis work is twofold. Firstly, an understanding of the current inner source research will be obtained in a literature review. With an understanding of IKEA’ software developer processes, interviews will facilitate identification of metrics relating to inner source.

Secondly, the gathered metrics will be used to compare the teams and assess a general state within IKEA’s developer team regarding inner source. With a literature review of current research, an assessment will be given for how IKEA can proceed with inner source initiatives.

1.4. Problem definition

This thesis will answer the following questions:

1. What are the general software development processes within IKEA?
2. What metrics from the repositories' metadata can be used to evaluate teams in regard to inner source?
3. What conclusion can be drawn from the gathered metrics regarding the current state of software development?
4. What recommendations can be given to IKEA consider future inner source initiatives?

1.5. Delimitations

This work will focus on a select few teams chosen by IKEA to conduct the survey. IKEA encompasses numerous developer teams and adheres to relatively lenient guidelines concerning software development processes, leading to the adoption of various methodologies across the organisation. Consequently, the repository-centered approach of this thesis work might not accurately describe the teams using other tools of some capacity, for example other platforms for collaboration and communication. Additionally, the development metrics do not measure the quality or effectiveness of a developer team or their processes, they are utilised to assess the potential for adopting inner source practices within the team.

1.6. Motivation for thesis

The advantages of inner source development have become more well established and seen wider usage with positive outcomes, which IKEA could potentially benefit from. Benefits such as increased development time and increased developer satisfaction would benefit the digital progress within IKEA. An increased knowledge of these processes in professional and academic contexts will also bring the benefits to the overall society.

For the author of this thesis, being able to engage in the early stages of inner source adoption was a very exciting prospect. Especially in and global and stimulating environment such as IKEA.

2. Technical background

This chapter explain the tools and software used for gathering the data from the developer team’s code repositories, as well as some of the important software development practices and methodologies which are of importance for understanding this thesis work.

2.1. Inner source and inner source practices

Inner source is a collaborative software development methodology that applies the principles of open source development to projects within an organization. It promotes the sharing of knowledge and expertise across teams, and have been shown to result in improved efficiency, innovation, and employee engagement [2][4]. It also fosters collaboration and breaks down silos, meaning isolated development with low or little external insight and input [5][6].

Although here’s no exact definition of the practices, Inner sourced development consists of several key components:

1. Open communication and transparency: Encouraging open communication and sharing of information helps build trust among team members and fosters an environment beneficial for collaboration.
2. Code sharing and reuse: Inner source practices promote sharing code, libraries, and components across teams, thus reducing redundancies. This enables teams to build upon each other's work.
3. Peer review: Emphasizing peer review helps maintain code quality and ensures that contributions adhere to established standards. For IKEA, GitHub is the source control system of choice and a platform for code reviews using pull requests.
4. Meritocracy and contribution-driven culture: Recognizing and rewarding contributions based on merit.
5. Documentation and knowledge sharing: Providing comprehensive documentation, guidelines, and best practices helps ensure that team members understand the expectations and processes involved in inner source development.

2.2. Goal Question Metric-methodology

The Goal-Question-Metric (GQM) methodology will be employed to identify crucial aspects of the teams' software development processes [7]. Which will be used to identify key development metrics concerning inner source.

The GQM-approach during the thesis work has been a 4-part process.

1. Information gathering and planning: Gather information regarding inner source and IKEA’s software development practices, as well as planning the course of the survey.
2. Goal: Identifying IKEA’s goals for software development using inner source practices, from a software developer’s perspective.
3. Question: Once the goals are defined, the next step is to devise a set of questions that characterises the current processes of software development regarding the goals.
4. Metric: The final step involves defining quantitative metrics that can be used to answer the questions formulated in the previous step. One metric can correlate to several of the goals. The metrics provide a means to collect data and assess the effectiveness of improvement efforts concerning the defined goals.

2.3. GrimorieLab

GrimoireLab is an open source software tool that is designed to support the analysis of software development and community activity data [8]. GrimorieLab is the main method of gathering the data for this thesis work. It provides a set of integrated tools that enables a range of analyses on data from various sources, including code repositories.

An important feature of GrimoireLab is modular architecture. One of the more important tools is Perceval, the component used for retrieving data from various sources and APIs. Two others of its modules are built on the open source tools Elasticsearch and Kibana. Elasticsearch is a search and analytics engine that handles large volumes of data. It provides near real-time search functionality and event data analysis [9]. Kibana is a tool for visualization and exploration [10].

The key feature of GrimoireLab is the ability to generate a wide range of visualizations that can help to gain insights into the patterns and trends that underlie software development, and will be used in this work to analyse the teams.

2.4. Docker and Docker Compose

Docker provides a way to build and manage containers using a simple command-line interface [11]. The interface can then be used to build an image from a specification called a Dockerfile, which describes dependencies and configurations for an application. The resulting image can be run in a container on any system that supports Docker. Effectively, this makes individual computer’s operational system (OS) a non-issue, enabling the images to be used without the need for different configurations.

Docker Compose is a tool that allows developers to define and run multiple images [12]. With Docker Compose, developers can define a set of images that make up an application and specify how they should be configured and connected to each other, facilitating creation and management of complex applications.

GrimorieLab can be installed locally by cloning the individual components from the GitHub repositories and setting them up individually. For this thesis work however, GrimorieLab’s Docker Compose image was used [13].

2.5. Windows Subsystem for Linux

A prerequisite to run Docker and Docker Compose with a Windows OS is to use Windows Subsystem for Linux (WSL) [14]. It is a feature that enables Linux applications to run natively on Windows. It provides a compatible interface with the capability to run Linux tools and applications. Essentially, WSL enables developers to access the Linux command line tools and utilities from within the Windows environment.

2.6. GitHub REST API

The GitHub REST API allows developers to work with many different resources such as repositories, pull requests and issues by enabling programmatic interaction with GitHub through HTTP requests [15].

GrimoireLab’s backend tool, Percival, can gather a wide range of data, however it doesn't encompass the functionality for supporting the data gathering needed for this thesis work. To address this, custom Python scripts accessing the GitHub REST API were developed.

2.7. GitHub pull request

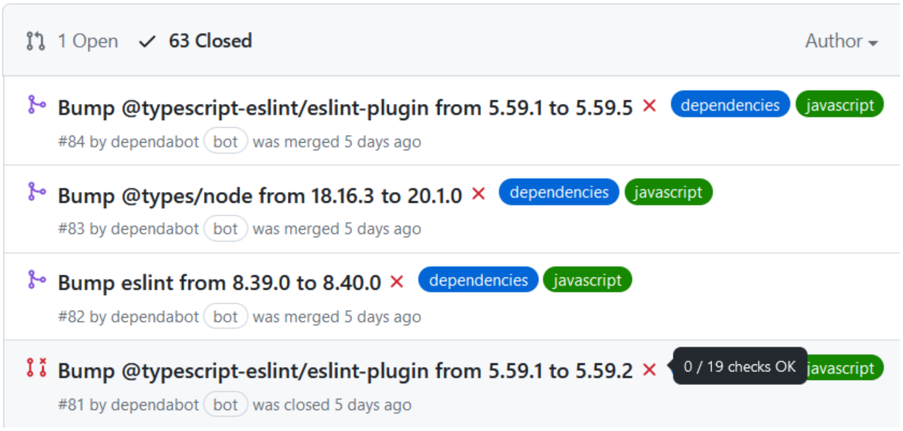
A GitHub pull request (PR) is a process that allows developers to propose, review, and merge code changes within a repository [16]. The process starts by creating a new branch of the repository’s code to work on a feature or fix a bug. Once the changes are made, the branch with the new code is pushed to the repository which initiates a pull request. During the pull request, others can review the proposed changes and provide feedback. If some part of the suggested PR needs to be changed after review, a new commit will be made on the branch, Once the review is complete, and any requested changes are made, the pull request can be approved, and the branch merged with the source code.

The pull request process is an integral part of IKEA’s continuous integration/continuous deployment (CI/CD)-pipelines [17], a process of software development practice aiming for frequent releases with small updates to the source code.

2.8. GitHub Actions and GitHub Checks

GitHub Actions is an automation platform that integrates with GitHub repositories to enable CI/CD pipeline services and handle tasks such as executing tests, building, and deploying code. Within the GitHub Actions platform, a workflow is a predefined series of tasks, called jobs, that are orchestrated in a specific sequence. Triggered by events like pull requests, workflows automatically initiate and carry out the designated jobs such as running tests, building code, and managing other aspects of the pipeline process, such as division of code reviews [18].

GitHub Checks is an API and user interface integration that allows developers to view the results of the workflow jobs directly within GitHub [19]. This simplifies code review and collaboration by displaying check results in the pull requests menu as shown in Figure 1, a screen dump from GitHub Checks' official repository [20]. A check can show a few different statuses on job completion, such as passed, fail, cancelled and more. For this thesis it's important to note a cancelled check counts as a fail when overviewing checks from the PR menu. A complete log can be found by looking at the individual commits.


1 Open ✓ 63 Closed Author

Bump @typescript-eslint/eslint-plugin from 5.59.1 to 5.59.5 ✕ dependencies javascript
#84 by dependabot bot was merged 5 days ago

Bump @types/node from 18.16.3 to 20.1.0 ✕ dependencies javascript
#83 by dependabot bot was merged 5 days ago

Bump eslint from 8.39.0 to 8.40.0 ✕ dependencies javascript
#82 by dependabot bot was merged 5 days ago

Bump @typescript-eslint/eslint-plugin from 5.59.1 to 5.59.2 ✕ 0 / 19 checks OK javascript
#81 by dependabot bot was closed 5 days ago

Figure 1. Results from workflow jobs, showing as GitHub Checks on a PR

2.9. Visual Studio Code

The IDE Visual Studio Code (VSCode) was used in the process of developing Python scripts [21]. It is a free source code editor developed by Microsoft. It supports numerous programming languages and provides features like syntax highlighting, code completion, and debugging. For this work, the necessary scripts were written in Python using VSCode's Python extension.

3. Methodology

This chapter describes the process and methods used during the thesis work.

3.1. Thesis work process

The process of this thesis work has been iterative, however there have been three major phases, with separate steps as seen in Figure 2 [22]. The phases are:

1. Information gathering and planning-phase
2. Metric identification, and gathering-phase
3. Analysis phase

There has been much overlap of the phases due to the iterative approach. For instance, the comprehension of IKEA's team's software development practices has been a continuous learning experience that extended well into the third phase.

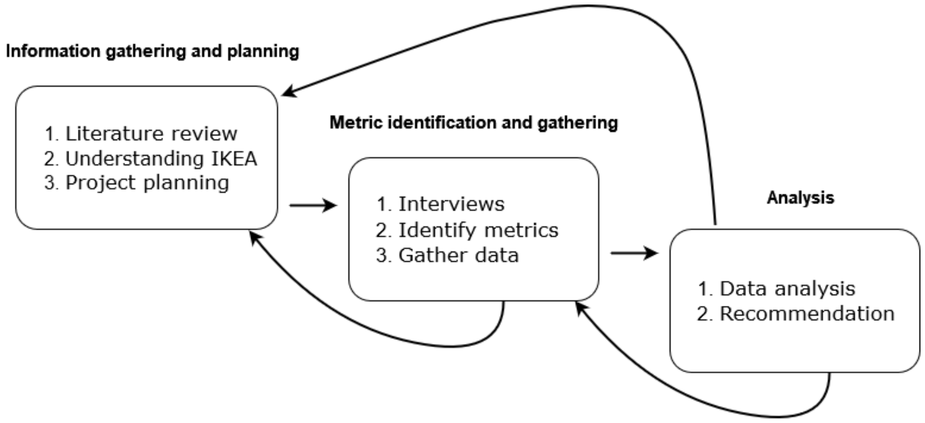


Figure 2. The iterative process for the thesis work

The first phase, information gathering and planning, can be further divided into three steps: Literature review on inner source, gaining an understanding of IKEA's software development practices, and thesis planning. The planning included choosing the methodology for identifying development metric.

The second phase, identifying and gathering metrics, consisted of three stages: interviews with developers, identifying the development metrics, and gathering the data by configuring GrimorieLab and writing custom scripts.

In the final phase, the analysis phase, the collected data was combined with an evaluation of the gathered research material to present the results of the teams, and finally determine recommendations for IKEAs continuous inner source initiatives.

3.2. Information gathering and planning

The first phase consisted of an information gathering on inner source, gaining an understanding of IKEA's software development processes and company culture, and lastly project planning, including methodology specification for metric identification.

3.2.1. Literature review

The initial stage encompassed gaining an understanding of the fundamental principles and practices of inner source. To achieve this, an information gathering was undertaken. As the concept of inner source is still new, no books were found. The criteria for choosing the research are explained in chapter 3.5. The literature review was conducted by gathering research papers found using Google Scholar and LUBsearch, or found in reference lists of other papers, 14 papers were selected based on the criteria. The search words were variations of "inner source" "inner source practices" and "organisational open source". Of the 14 papers, 5 were used for this thesis. While the other 9 fulfilled the criteria, the specific areas of research of the papers were not aligning with the purpose of this thesis. A significant insight from the information gathered revealed that although metrics and inner source have been discussed, such as [2][4], research on inner source metrics is not yet well-established. Also, no research found mentions the processes of individual developer teams. This posed a challenge during the thesis work, as limited guidance was available to determine the most suitable direction. This situation also granted the flexibility to explore and experiment with different approaches.

3.2.2. Understanding IKEA

This secondary step involved acquiring an understanding of IKEA's development process, which was partly obtained through accessing their internal platforms for documentation and communication. Access to Confluence, IKEA’s main platform for documentation, provided insight into their internal processes and development guidelines [23]. The company's Slack platform provided valuable insights into various internal communication channels [24]. This opportunity allowed the author to gain a deeper understanding of the organizational culture and dynamics. Moreover, the guidance provided by IKEA supervisors and others in the organisation greatly contributed to a better understanding of the day-to-day operations.

3.2.3. Thesis planning

With the initial stages concretised, a project plan was developed encompassing the major milestones of the thesis work and an estimation of the time required to complete each stage.

During this stage, the decision of using the GQM-methodology was also made and planned for, The GQM-approach described in chapter 2.1 was chosen partly because of the author’s familiarity with the methodology, but also because of practical examples of it being used for the purpose of identifying inner source metrics [25].

3.3. Metric identification and data gathering

The second phase involves utilizing IKEA’s inner source goals, to develop questions for the interviews, and using the developers’ answers to identify and gather development metrics. This includes configuring GrimoireLab, writing necessary scripts and ultimately collecting data from the development team's repositories.

3.3.1. Interviews with developers

Using the GQM-approach, 7 developers were interviewed. Typically, they were senior DevOps engineers, and all selected by the supervisors at IKEA. The interviewees were picked based on their experience in IKEA or open source-related development. Semi-structured interviews were chosen, which implies the interviews consist of a series of questions that can be flexibly adjusted based on the interviewee's responses, allowing for a more in-depth exploration of the participants' perspectives and experiences [26]. The semi-structured interview method was selected, in part, due to the author's limited experience with professional software development, which would make it challenging to ask relevant questions. Open-ended questions also allowed to prompt follow-up questions on tangents or topic that needed clarification. An initial interview guide was developed as seen in appendix A-1. For each of the inner source goals, one or more questions were asked, allowing developers to describe the essential aspects of their development in relation to the respective goal.

As outlined in chapter 1.1, the approach of this thesis work was partially a joint effort. Questions related to Mandana’s readiness assessment survey were asked during the same interviews. Although the primary purpose of these questions aimed at metric identification, they enhanced the understanding of IKEA's general processes and facilitated the improvement of the interview guide. After each interview the answers were reviewed, and the guide was continuously refined as this work’s author's comprehension of the subject deepened.

The interviews were mostly conducted with single developers face-to-face, online. In addition to providing crucial information for identifying metrics, the interviews revealed some vital practices that aided in determining how to gather the data, for example explanation of their use of automated tests in their CI-pipeline proved essential for the GitHub Checks-metric which will be described in chapter 4.

One of the interviewed developer’s team was technically part of one of IKEA's other subsidiaries, Inter IKEA. As a result of the organisational structure, access to this team's repositories were not granted and no repository data could be gathered and analysed from this team.

3.3.2. Identifying development metrics

To pinpoint the metrics, the open-source community CHAOSS's metrics were referenced to try to enable the use of industry-established standards [27]. This proved however not to always be possible after analysing the responses from the interviews, and customised metrics were needed as will be described in chapter 4. These custom metrics were based on the interviewed developer's descriptions of their processes and 2 internal development guidelines:

Engineering baseline, specifies that every repository should contain a readme-file, containing information and documentation about the repository. It also specifies that it should contain a link to additional documentation of the product.

Repository prerequisites, a document from The Open Source Program Office (OSPO), a department within IKEA responsible for overseeing open source projects as well as certain internal source initiatives. Specifies documents in repos teams should have if aiming to inner source their products.

3.3.3. Gathering repository metadata

The third step consists of setting up the necessary tools and gather the repository metadata. GrimorieLab, Docker, Docker Compose and WSL were used as described in chapter 2, as well as using VSCode to write scripts. 56 repositories belonging to 6 teams were used for the analysis, the process of sorting the repositories of the teams is described in 4.4.1. The repositories were obtained through a custom Python script.

3.4. Analysing

The last phase consisted of analysing the gathered data and combined with the findings from the literature review, conclude in a recommendation for how IKEA can proceed going forward.

3.4.1. Data analysis

The data gathered from the developer team's repositories were analysed in order to assess team's performance concerning the identified inner source goals. Both the average team's data as well as interesting observations from repositories will be used for this purpose. GrimoireLab provides gathering and visualisation capabilities of the visibility and time-to-close metrics used in this thesis, but to ensure consistency and not disclose the team and repository names, scripts were utilized for generating all graphs and tables shown in chapter 5.

3.4.2. Recommendation

The combination of data from the metrics, information about IKEA and the development processes, and research findings from the literature reviews, helps identify critical success factors, potential obstacles, and best practices for implementing inner source approaches for the development teams in IKEA.

3.5. Source criticism

The credibility of the sources was based on the following criteria: First, the credibility of the sources was assessed by assessing the publisher, prioritising articles that were peer-reviewed. Second, a preference was given to more recent articles to ensure up-to-date information.

Sources numbered [8]-[16], [18]-[21], [23]-[24], [27]-[28], and [32] are directly tied to the respective tools or official documentation websites, ensuring the accuracy of the information they provide. Sources [1]-[2], [5], [22], [26], [29] and [31] are valid considering the publication in peer-reviewed journals, ensuring scholarly merit. Sources [3]-[4], [6], [14], [17], [33]-[37] are published from conference papers. Source [7] and [25] are from a book published by a reputable scientific publisher and a practical, empirical guide. Finally, the methodologies in references [29]-[30] adhere to the descriptions provided by their respective originators.

4. Analysis

This chapter presents the outcomes and discoveries obtained through the employed methods, while also providing motivation for the decisions made throughout the process.

4.1. Inner source goals

The process of the GQM-survey involved determining the expectations IKEA had for inner source development, and how the goals could be used to determine development metrics. To pinpoint the goal, the supervisors in IKEA were questioned regarding their expectations for software development using inner source practices, yielding the goals as seen in table 1. The goals are numbered without inherent prioritization and will be used when motivating the metrics in following chapters.

Table 1. Description of IKEA's inner source goals

Goal	Description
1. Reduce silos	Removal of barriers that limit collaboration between teams
2. Reduce bottlenecks	Lessening limiting factors that impedes development
3. Reduce development time	Shorten the duration required to complete a project.
4. Improve quality	Increased quality of the product and the development process
5. Increase reusability	Reuse of code and components
6. Increase knowledge sharing	Increased communication, ideas, and expertise sharing among teams

Using the goals, an initial interview guide was developed, seen in appendix 1-A, with the aim to allow the developers to characterise their software development processes regarding the different goals.

4.2. Understanding IKEA

This chapters describes the effort of understanding the processes in IKEA and the developer teams. It presents the finding from the interviews and motivates the identification of the metrics.

4.2.1. IKEA’s software development

An understanding of the principles of IKEA’s software development was gained by taking part of internal documentation as described in 3.2.2, as well as information from interviews and informal discussions with people in the organisation. IKEA’s software development can be characterised as open and inclusive. An example of this is the use of an internally open GitHub Enterprise cloud, an organizational platform within GitHub with project management and team administration features [28]. The teams and their products as well as the corresponding source code and documentation is open and accessible for anyone within the organisation.

The developer teams are working very autonomously, with few restrictions on the tools, languages, and methods they deploy. The openness and transparency observed among IKEA's developers and inside the organisation is crucial for embracing inner source practices [2][3][5]. However, the variety of tools, processes, and the organisational structures also present difficulties for inner source development [3]. This diversity is a challenge considering the need to take into account the unique characteristics of each team and their preferred methodologies.

4.2.2. Interviews with developer teams

During the interviews, a few key discoveries appeared that were integral for the decision made during the thesis. One of these was that every team employed agile development processes, with some utilizing scrum, others kanban, and some a blend of both [29][30][31]. Furthermore, all teams used Jira, a project management tool, to implement the agile artifacts, such as planning using scrum boards or retrospectives at the end of sprints [32]. Another significant observation from the interviews was the widespread use of DevOps practices such as CI/CD-pipelines.

Generally, a single team has ownership and responsibility to maintain one or a few related products, often consisting of several different repositories to support the functionalities, as seen in Table 2. The repositories used in this thesis were filtered and gathered using the criteria mentioned in chapter 4.4.1. Some teams, such as team A and B, has ownership of repositories that are codeveloped with other teams.

Table 2. The number of members and repositories of the interviewed teams

Team	Repositories	Members
TEAM A	16	7
TEAM B	14	7
TEAM C	8	8
TEAM D	5	8
TEAM E	8	6
TEAM F	6	9

The different challenges the teams faced became apparent from their answers during the interviews. Some teams produce services for internal use while other caterers to other businesses, leading to different requirement. Development within the organisation was described by one developer to be able to not have the same emphasis on deadlines. Another challenge was technical dept, mentioned by two of the teams. This refers to previous shortcuts and compromises that hinders further maintainability and productivity. One teams, team E, recently inherited the ownership of their current product, and at the time of the interview described they were in the process of understanding the different functionalities and write new documentation since it was previously severely lacking.

One of the teams interviewed, here designated as Team F, stands out as being the biggest of IKEA's inner source initiatives. Among the six repositories identified for team F, one is inherently inner sourced and vigorously promoted within IKEA. It serves as a cornerstone of the company's initiative to inner source development.

Many developers shared similar objectives and faced comparable challenges, such as utilizing GitHub for version control and adopting many Agile and DevOps methodologies. Nevertheless, they employed a diverse array of tools and methods for testing, processing pull requests, and managing other aspects of the product development. Team C for example, used highly automated pipelines in many of their repositories. Notably, they were using bots to complete tasks such as automated updates of dependencies in several repositories, and in another, used a bot to push all commits as PR drafts. These differences made it challenging to compare the team’s different development processes and posed difficulties when determining how to gather the metrics that were not covered by GrimorieLab’s functionalities.

Currently, there are not much collaboration among developer teams across organizational units within IKEA, unless the team’s products had some level of interaction. Teams within the same domain did however collaborate to some extent or shared repositories, as seen in chapter 4.4.2. Another interesting finding from the interviews was the prevalent use of workshops for many of the teams, where they presented or were invited by others to try their product or some related technical challenge.

4.3. Identified metrics

While determining the metrics, open source metrics from the organization CHAOSS were used as inspiration as described in chapter 3.2.2. When these were not applicable, the internal specifications Engineering baseline and OSPO’s inner source documents specifications were used. The result of the developers' answers during the interviews, as illustrated in Figure 1, shows how IKEA’s inner source goal were used to correlate with the 5 identified metrics, which will be described in following chapters. One metric frequently corresponds to several of IKEA's inner source goals. The metric can be used to analyse individual repositories as well as the different teams.

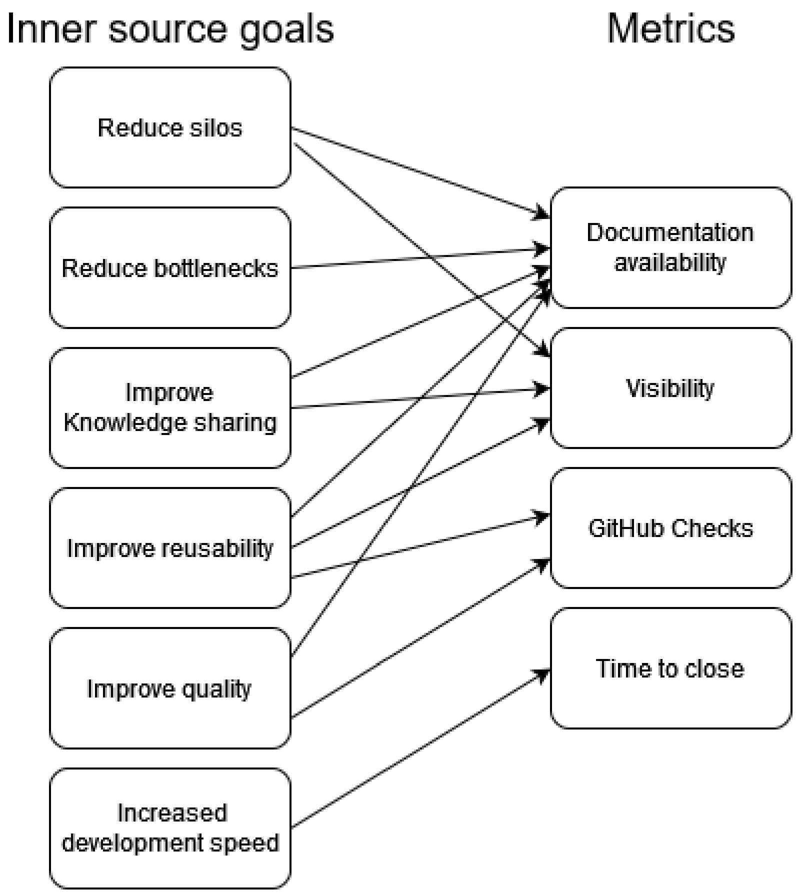


Figure 3. Identified metrics relating to IKEA's inner source goals

4.3.1. Document availability

During the interviews, developers emphasised the significance of available and up-to-date documentation in relation to questions concerning IKEA's inner source goals 1, 2, 5 and 6, seen in Table 1. This highlights the critical role that comprehensive documentation plays for the purpose of collaboration and promoting the reuse of code and resources. They did also mention that documentation is a part of the development that is often found lacking. Research also notes the problems with missing and low quality documentation for inner source initiatives, and the difficulties it brings for such initiatives [3].

Document usability was the CHAOSS-metric most closely correlating to the developers' answers in the interviews concerning the importance of documentation. For instance, when describing their approach of utilising APIs or services from another team, where the first step was to find and read the documentation to understand the functionalities. Documentation usability is however a well-defined metric, measuring requires qualitative methods for effective evaluation which entails conducting in-depth interviews or monitoring of task completion [33]. These processes can be time consuming and are beyond the scope of this thesis.

A model for the metric document availability has been proposed by Matulevičius et al. [34]. Although this model is not directly applicable due to differences to IKEA's specified documentation set and the proposed model's, it inspired the definition of the metric documentation availability as used in this thesis.

GrimoreLab lacks the functionality to provide metrics related to documentation, and a custom Python script was developed using the internal specifications mentioned in chapter 3.3.1, described below in Table 3.

Document availability is calculated by searching for the specified documents in the repositories, for the existence of links in the README as shown by (1), where d_n specifies the found documents and links.

$$\sum_{n=1}^{6} d_n \tag{1}$$

For the calculation of a team's documentation availability, the scores of all repositories are summed and divided by the total number of the team's repositories.

Table 3. Description of the documents used when assessing documentation availability.

File name	Description
Readme	Introduces a repository, often containing information such as project purpose, installation instructions or dependencies.
License	Outlines the legal rights and restrictions associated with the project.
Code owners	Specifies the individuals or teams responsible for maintaining and reviewing changes in specific within a project.
Contributing	Provides guidelines and instructions for potential contributors who wish to participate in the project's development.
Pull_request_template	Template for how a pull request shall me constructed.

The documentation available on GitHub does however not provide a comprehensive understanding of the team's documentation, as more and better detailed documentation usually can be found on the product’s Confluence page. It does offer an indication of the team's current prioritisation regarding documentation, and what can improve should the team want to inner source their development. Also worth mentioning is that even though the team’s repositories have been sorted, described in chapter 4.4.1, the result of the metric Document availability don’t consider the activity or importance of the different repositories. It can be assumed that repositories serving more core functionalities are more prioritised and valued for documentation purposes. Taking this into account, the individual repositories shown in chapter 5.1.1 are also analysed using the number of PRs for each repository. For the team’s score, the average of all the team’s repositories is used.

4.3.2. GitHub Check failrate on merge

The significance of different software tests for ensuring software quality emerged as one of the most evident findings from the interviews and was mentioned by every team, aligning mostly with goals 4 and 5.

Test coverage, a metric measuring the degree to which an application's source code is covered by test suites, initially emerged as a potential metric to address the goals. However, collecting this metric proved challenging due to the need to get access to the testing tools employed by the developer teams, which was made more complicated by the fact that many teams utilized multiple tools for various testing purposes.

The developer’s processes of ensuring quality of their products through automated pipeline testing using GitHub Actions, described in 2.9, was an alternative approach for assessing these goals. The use of GitHub Checks facilities a quick overview of the results from the workflow jobs. The ability to differentiate the teams based on the outcomes of jobs using GitHub Checks presents a new challenge: To evaluate the results of the individual commits in a PR or to use the final Check results of merged PRs. Using the results from checks on merged pull requests was chosen, and is motivated by two factors:

1) It shifts focus from the individual tests to the unofficial acceptance criteria established by the developer teams, as seen by the number of failed tests the allow to be merged with the source code.

2) The optimization and deployment of test suits is an extensive subject outside the scope of this thesis [35].

Although research on inner source and testing is limited [2]. Test results have been shown in open source projects to have a significant impact on the likelihood of a pull request getting merged [36]. Additionally, the size and scope of test suites affects the merge rate of PRs. Automated testing in pipelines increases both external and internal contribution. However, while a large test suite can increase the merge rate of PRs originating from within a team, it can also decrease the number of external PRs getting merged. This suggests that teams with extensive test suites in the pipelines have a higher barrier of entry, leading to reduced external contribution, and would not be optimal in an inner source context.

Another aspect to take into consideration when using this metric is that the workflows the teams use varies significantly. Many workflows are dedicated to CI/CD-pipeline maintenance tasks, such as automating pull request drafts or sending notifications for code reviews, which are not directly related to quality tests. Several teams incorporate quality-related testing jobs within the same workflows that facilitates pipeline maintenance, making it to distinguish the two. The results from the GitHub checks as used in this thesis will provide a measure of CI/CD-pipeline's effectiveness in terms of a team's failrate acceptance.

To calculate the metric GitHub Checks on merge of a single repository. The number of failed and cancelled jobs are divided by the total amount of jobs on merged PRs in the last 90 days, as seen in (2). Where j_f and j_c represents failed respectively cancelled jobs, and N is total number of jobs on the PRs.

$$\sum_{n=1}^{N} (j_{fn} + j_{cn}) \Big/ N \tag{2}$$

The calculation for an entire team is the sum of all cancelled and failed jobs in all repositories divided by the number of total jobs. As described in chapter 2.8, the reason both the number of failed and cancelled tests are used, is because the PR menu in the GitHub repositories shows them both as failed.

Merged PRs that have not passed all checks do not necessarily suggest issues with the code. Within a closed development environment, it's reasonable to assume that teams comprehend the tests they implement and their associated significance. A failed test could, for example, be a failed linting check. Linting, in this context, is a kind of automated software testing that checks code for stylistic or formatting errors, as well as certain types of programmatic errors [37]. Not all linting failures indicate functional problems with the code. For instance, a linting failure might be triggered by indentation inconsistencies or trailing spaces, which do not impact the actual execution or functionality of the code. Therefore, pull request are often merged even if linting checks have failed. A failed test can also be the result of the use workflow jobs not correctly covering the submitted commit.

Direct comparisons of the number of checks or failure rates is a complex task due to diverse team circumstances and requirements. Nevertheless, some factors may signal a team's aptitude for inner source initiatives. In adherence to inner source principles that emphasize transparency and efficient information sharing, repositories should ideally aim for a failure rate of zero or near-zero on merged pull requests [2][30]. In conclusion, teams with either an abnormally small or large number of jobs, or those with a high failrate acceptance, may not be the most suitable candidates for inner source development.

4.3.3. Time-to-close

Time-to-close specifies the amount of time between the creation of a PR until it is closed. A PR is closed when it either has been merged or it’s discarded by the team or the submitting developer. Time-to-close was identified based on the developers' descriptions of their development using GitHub pull requests, relating to goal 3. Because of outliners in the data, time-to-close is here calculated as the median time-to-close of the pull requests.

The metric lead time, the duration between specifying and designing a feature and putting it into production, was initially considered after analysing the answers from the interviews. However, similar to test coverage, collecting the data would require access to the individual teams' process and planning platform Jira. It can also be assumed that since team-specific activities such as planning and initial feature design occurs within a confined context of the team, lead time is not well-suited as an inner source measurement.

The different challenges and requirements faced by the teams makes it difficult to draw meaningful conclusions or comparisons using time-to-close alone. When a team's or repository’s time-to-close falls within a normal range, not much can be inferred. It can however serve as a benchmark for monitoring the outcomes of inner source practices over time. In cases when the time-to-close is exceptionally long or short, some assumptions can be made which highlights practices of the developers. For example, in some repositories the same developer both pushes and merges PRs after pipeline tests finished, the same team also had several repositories where the time-to-close was extremely small. This indicates code review happens face-to-face or not at all. For the purpose of inner source, not having a clear review process indicates lack of transparency. A very long time-to-close could hint towards a lack of engagement of the GitHub processes or other development process constraints which likewise would indicate less inner source compatibility.

The research on inner source mentions time-to-market as one of the established benefits of inner source development [2][30]. No research has however been conducted specifically regarding time-to-close or development increments of similar scale. It is unknown whether the median time-to-close time in regard to individual PRs would increase, decrease or stays the same when implementing inner source practices.

4.3.4. Visibility

During the interviews, for questions relating to goal 1, 3 and 5, developers explained the practice of repurposing code from other teams, usually by forking a repository with the desired functionality. Additionally, one of IKEA's internal guideline documentations, GitHub Guidelines, encourages developers to use GitHub's integrated functionality to interact with repositories as described in table 4.

Table 4. Description of component of the metric visibility

Repository action	Description
Fork	Creates a personal copy of a repository, enabling experimentation
Star	Serves as a bookmark for easy access and shows appreciation for a project
Watch	Allows users to receive notifications about a repository's updates and activities

The number of forks, stars and watches for a repository can provide insights into the visibility and activity surrounding the repository and the team and indicates knowledge sharing in the organisation. A diversity of stakeholders is necessary for a product to be successfully inner sourced [3]. These three significators for repositories of the teams will be equally valued for the calculation of the visibility metric. However, different aspects of a repository can be inferred depending on which one that's analysed. A high number of forks indicates that the product might have functionality reused in other team's products as previously described, pointing to the reusability, while a star and watch more closely aligns with knowledge sharing.

4.4. Gathering and visualising the data

With the metrics identified the next step was to gather and visualise the data from the repositories. Grimorielab was the main method for this purpose, complemented with scripts when the functionalities were lacking.

4.4.1. Repository and data selection criteria

The repositories belonging to the teams could easily be found on IKEA's Enterprise GitHub. However, many repositories belonging to teams were not used for the functionalities of their products and thus not suitable for gathering when analysis the metrics. In order to account for this, two criteria were applied when selecting the repositories. To only select repositories that had been updates in the past year, and to only get repositories that were not forks. A script was created that uses the team's name to produce a JSON file, which was formatted to be directly utilized for the configuration of GrimorieLab described in appendix A. While a better selection of repositories could be made by asking the developer teams what repositories to choose, the decision was made to filter the repositories in order to get as much data as possible and not to be influenced by the developers, for example, if they knew they had repositories they did not used according to set procedures, there is a possibility the developers would not want this to be a part of the analysis.

For the metrics Time-to-close and GitHub Checks failrate on merge, the data that will be analysed follows GrimorieLab's default time range of 90 days.

4.4.2. Using GrimorieLab

In the early stages of the thesis, the intention was to solely employ Python scripts and GitHub REST API for gathering metrics data from the team's repositories. During the information gathering, GrimoireLab was discovered, and its extensive capabilities seemed valuable considering the insights it could provide about the team's development processes. Particularly such as the network graph created and displayed using Kibana, Figure 3. In the graph, the bots pushing PRs have been removed in order to show the cooperation between developers. The graph shows pull request collaboration among the developer teams. The nodes are the highest producing developers based on the amount of pull requests made, and the size of the edges signify collaboration between two developers on pull requests. The graph is not team-member specific, as it also shows other developers who have contributed to the team's repositories. Some of the insights gained in the interviews can also be inferred from the graph. Team D who are using bots for most of their PRs, teal in the graph, is because of this seen as a single node. While there is some collaboration between the teams, they mostly develop independently of each other. Worth noting is that while Team F, here in tan, expectedly from an inner source project have more collaboration with other developers than many teams. Team B, green, has a lot of cooperation with members originating from other teams, judging by the number of nodes. Looking into the data shows this is the result of team B having ownership of a shared repository with many contributors.

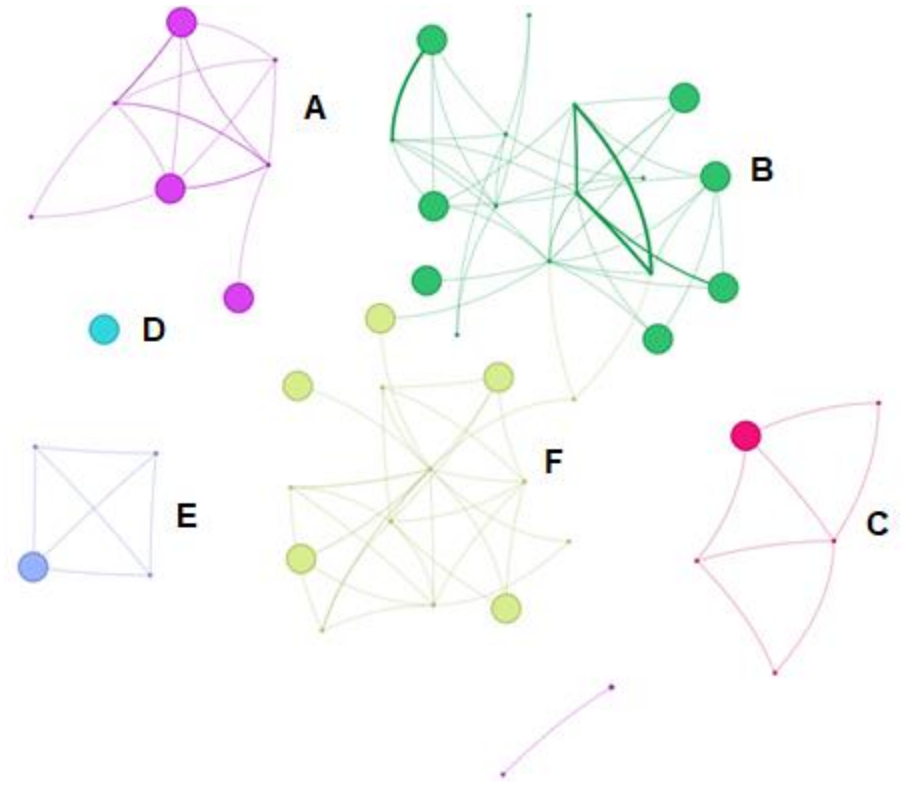


Figure 4. Pull request collaboration of the interviewed teams

When installing and configuring GrimorieLab, the docker-compose images were used as described in 2.3-2.4. And are further specified in Appendix 1-B.

For the metadata which could not be gathered using GrimoreLab, the metrics GitHub Check failrate on merge ad document availability, custom Python scripts were developed using VSCode. The graphs and tables seen in chapter 5 are not from GrimorieLab, for the purpose of keeping the name of the teams and repositories confidential in accordance with a signed nondisclosure agreement. Instead, data was downloaded from GrimorieLab or accessed directly from Elastiscearch and visualized with development Python scripts.

5. Results

In this chapter, the analysis of the team’s gathered metrics will be presented and conclude in a recommendation for what steps IKEA may consider going forward with inner source initiatives.

5.1. Results from identified metrics

As discussed in 4.2.2, the GQM-methodology and interviews with 7 developers in IKEA resulted in the identification of the 4 metrics seen in Table 5.

Table 5. Description of the identified metrics

Metric	Description
Document availability	The presence of documents and links in repository based on IKEA’s internal guidelines
GitHub Check failcheck on merge	The average amount of failed workflow jobs, seen at GitHub Checks, in merged PRs
Visibility	Number or forks, stars and watchers
Time-to-close	Median time a PR remains open until it is closed

5.1.1. Document availability

Figure 5 plots the average rating of the team's repositories’ document availability according to the criteria detailed in chapter 4.2.1. Looking at the team's overall performance, it is apparent that most have not incorporated the described inner source-related documents. This is not unexpected, given that most teams don’t use inner source methodologies in their development processes. However, ReadMe and associated links were present in most repositories.

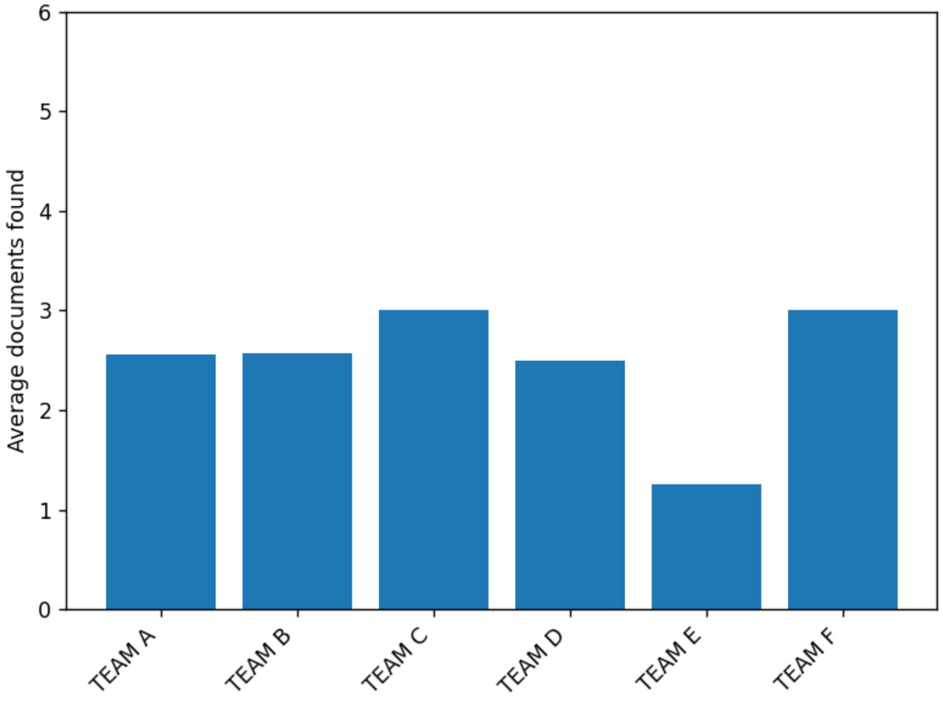


Figure 5. Documentation availability scores of the teams

Worth noting is however in team F, the inner source initiative, can be seen to have slightly higher Documentation availability than most of the other teams. Especially in repository 1 shown below in Figure 6. Colour coding the numbers enables efficient overview of the importance the repositories have when considering the team's focus of development.

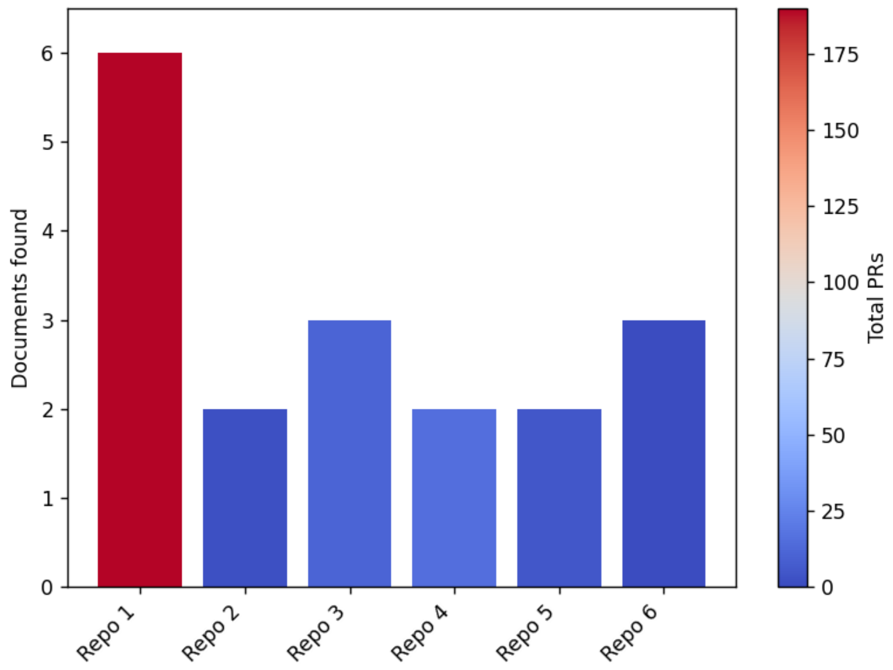


Figure 6. Documentation availability of team F

It's worth mentioning that Team F's Documentation availability spread of the repositories is an outliner among the teams. Team B, shown in Figure 7, is a more representative view of the distribution of the activity in the repositories of the different teams.

For teams wanting to adopt inner source practices, available and up-to-date documentation is a key factor as described in chapter 4.3.1. From this point it can be seen in the data that almost all team have potential for improvement considering the documentation they provide on GitHub.

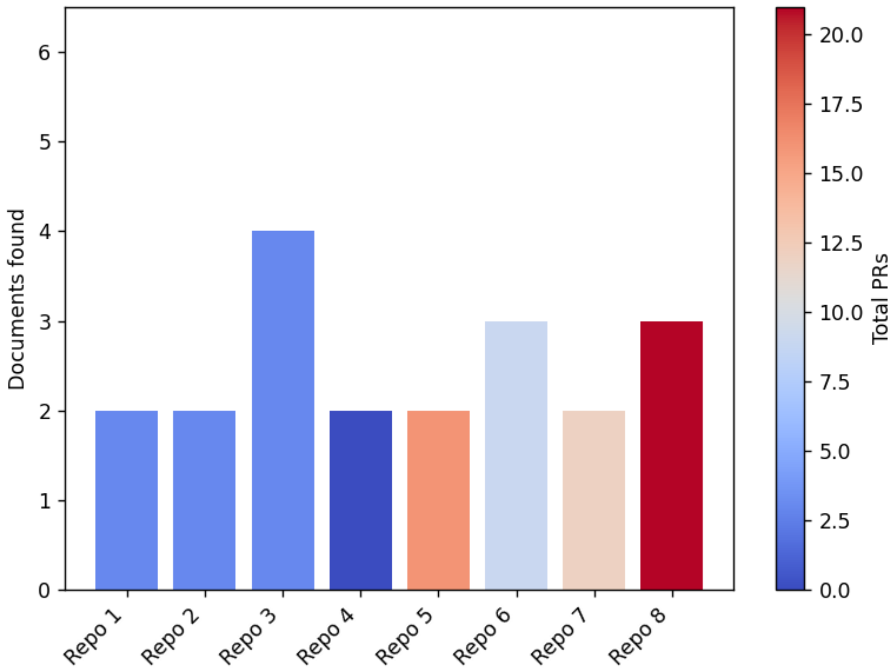


Figure 7. Document availability scores team C

5.1.2. GitHub Checks failrate on merge

Figure 8 presents the results for the team's Checks on merged PRs and average number of Check. As discussed in section 4.3.2, merges that fail tests do not necessarily indicate faulty code. However, they do reflect a lack of regard or prioritisation to the importance of clear and transparent test results. While this might be acceptable within a closed development team who knows the inherent characteristics of their product and tests, an inner source project should aspire to maintain a low failure rate, as exemplified by Team F.

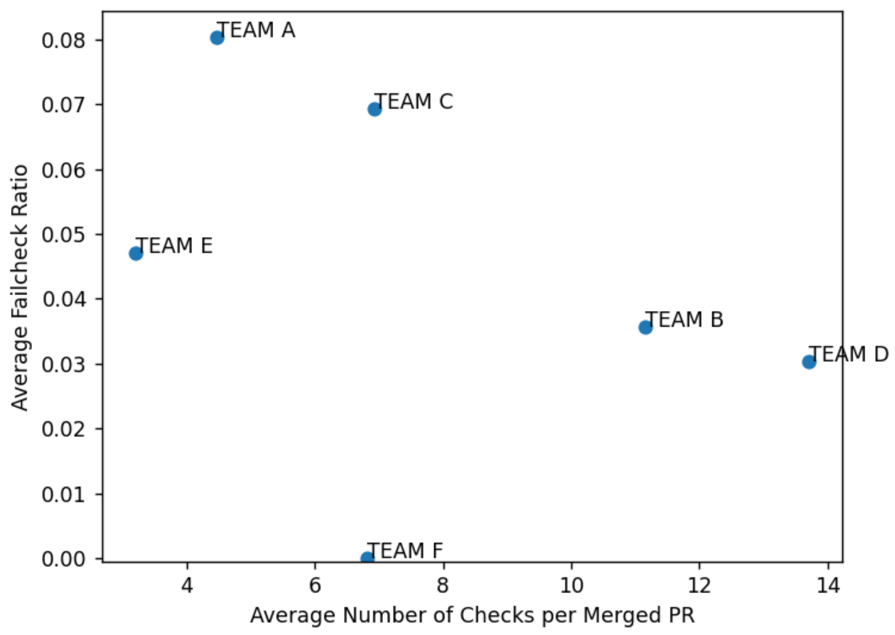


Figure 8. Average number of checks per PR and failcheck ratio of the teams

Using only the average number of failrate and jobs, displayed as Checks, as a comparison across teams will as discussed not provide a clear view of the team's processes. The number of jobs do however point towards certain characteristics of the teams. Team A, C and E for example, shows a relatively high acceptance to failed tests when merging PRs, while also employing fewer tests than the other teams.

Figure 9 highlights the results from the repositories of Team D, which proved to be particularly intriguing. As stated in chapter 4.2.2, the developer outlined their extensive pipeline, which is clearly reflected in the data. Repository 3 has a significantly high number of jobs in workflows within their pipeline, which by far exceed any of the other repositories in any of the teams. If cancelled jobs were not categorised as fails, as described in section 2.8, the failure check ratio would be even lower in this particular repository. As discussed in chapter 4.3.2, a comprehensive test suite in the CI pipeline could potentially deter external contributions.

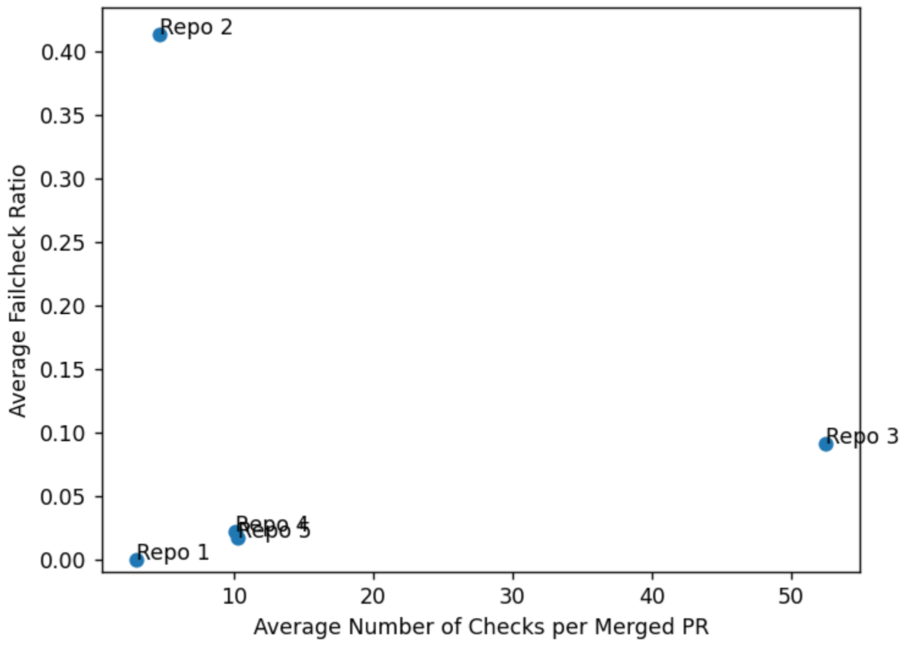


Figure 9. Average number of checks per PR and failcheck ratio for team D

In conclusion, the suitable number of tests can as discussed not be generalised simply to assess the teams. An assumption can however be made that in the current state, teams A, D, and E might may need to reconsider the tests they deploy or how they evaluate test acceptance, particularly with regards to the number of workflow jobs, would they implement inner source methodologies in their development.

5.1.3. Time-to-close

Figure 10 shows the median time-to-close for the different teams, based on the PRs from the last 90 days. As mentioned in 4.3.3, without knowing the specific context of the team's development characteristics, these number do not signify much about the projects. Most pull requests get closed the same day and except for team C, there is generally no big difference between the teams.

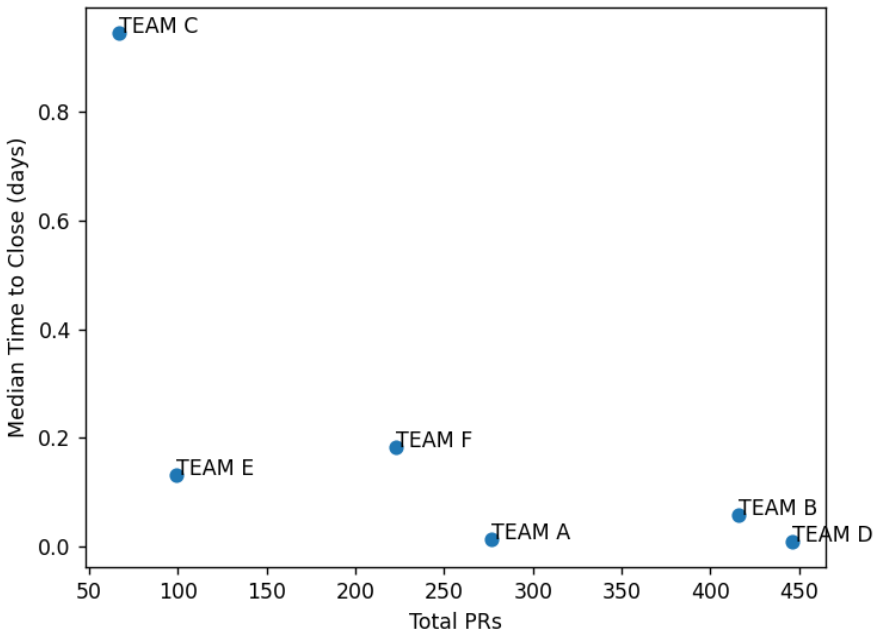


Figure 10. Median time-to-close of the teams

Looking into specific repositories sheds some more lights on the practices of the teams which is of interest. Most teams' repositories have a division among their time-to-close as shown by team C, Figure 11. While there is a repository with very high time-to-close, it doesn't affect the median value considering the few PRs. However, the over-all time-to-close would suggest that there may possibly be some hinderance in the team's overall development processes.

Looking into the repositories of team A, B, and D highlights some practices of these teams. Team D uses bots to automatically update dependencies in the repositories, leading to many PRs with an exceptionally quick time-to-close. Team A and D have a few repositories with particularly short time-to-close, which in some repositories are the results of a single developer pushing PRs without review. In other repositories there's only two or very few developers cooperating on the development and sometimes merges without review having been done. None of these occurrences are as described in 4.3.3 ideal for inner source purposes.

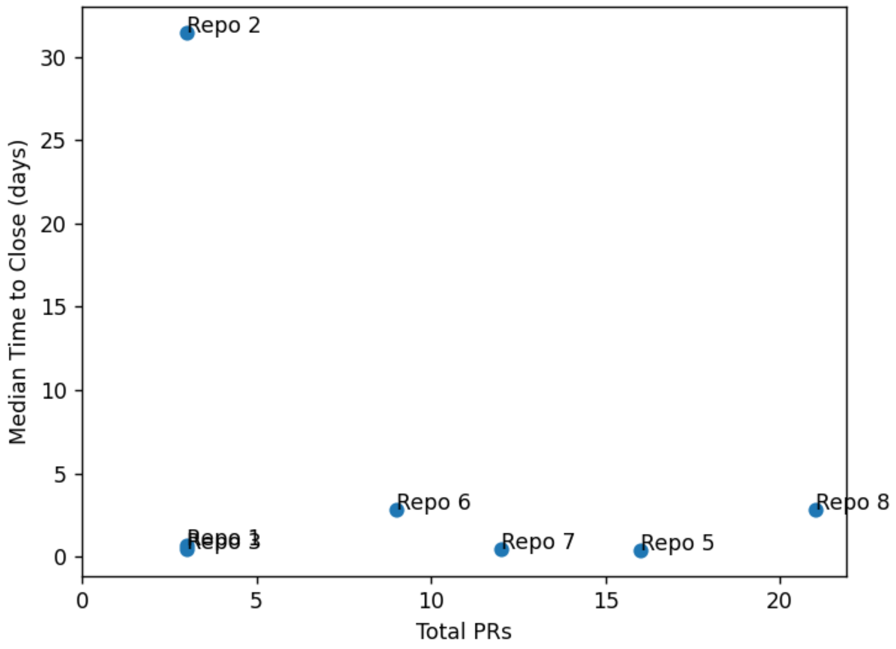


Figure 11. Median time-to-close of team C

In conclusion, the teams with very low time-to-close exhibit practices in some parts of their development which might not be suitable for inner source purposes, and in the case of not having code reviews, not in accordance with IKEA's requirements. Should these teams want to inner source their development, they should reevaluate the development processes in these repositories, which can start by increasing the cooperation and contribution of the developers within the team.

5.1.4. Visibility

As table 6 shows, there's a large difference in the number of forks, stars, and watchers for the different teams. Notably here is team F, the inner source initiative who scores highly in all three categories. Team F having a high visibility goes is line with the previous assumption that increased visibility as defined by the forks, stars and watchers can signify a project's inner source potential. Worth mentioning here is that a large part of the watchers for the team is a result of a repository used for a coding event, with the purpose of spreading awareness of the initiative and increase the collaboration.

Table 6. Visibility metric of the teams

Team	Forks	Stars	Watchers
TEAM A	31	20	31
TEAM B	1	11	38
TEAM C	1	14	2
TEAM D	17	7	16
TEAM E	2	0	34
TEAM F	45	20	169

Generally, the data shows that a repository with a high number of watchers also has many stars and forks. Examining a team's visibility can yield different insights. A high fork count, as detailed in Chapter 4.2.4, implies potential functionality reuse across different teams' products. Conversely, a notable number of stars and watches suggests knowledge sharing of the repositories. Another point previously discussed is the importance of a range of stakeholder, and as such, increasing the overall visibility of the teams is something the teams can try to achieve if they want to inner source their product. This can be achieved by events such as the one used by team F, or workshops as described in 4.2.2.

5.2. Recommendation for IKEA

Based on the findings of in this thesis, IKEA will be given a recommendation for ways to increase the probability of future inner source initiatives. The scripts for gathering the data as well as an instruction for configuring GrimorieLab in accordance with this thesis will be presented to faciliate the analysis of more teams as well as follow-up of the teams interviewed in this thesis work. Following are a few guidelines, derived from the thesis' findings, which IKEA can adopt as a framework when determining the teams to be selected for future inner sourcing:

1. **Improve documentation**: The data gathered from the repositories showed that the Documentation availability in the repositories was found to be inconsistent across the teams. Documentation is essential for new team members, or contributors from other teams, to quickly understand the project and start contributing. Therefore, IKEA should ensure that documentation is regularly updated to accurately reflect the state of the project.
2. **Review and adjust test acceptance practices**: The findings suggest discrepancies in the accepted Check failrate on merge across different teams. Some teams might require a large number of tests due to the requirements and characteristics of their products. It is recommended to ensure that results of the workflow jobs are appropriately addressed, and that the need for a large number of tests in the pipeline is adequately justified. Other teams appear to have a higher acceptance towards test failures, or showcases low numbers of pipeline tests overall, both of which should be reconsidered.
3. **Optimize the PR management process**: The data shows discrepancies in the median time-to-close for different teams. Teams with very low time-to-close have been seen in some cases to be skipping important processes such as code reviews. Almost all teams had repositories with very high time-to-close, possible indicating lack of engagement or prioritisation. Therefore, it is suggested that IKEA needs to further promote proper PR management processes, including code reviews, to ensure code quality and collaborative practices.
4. **Increase project Visibility**: The data and previous research suggests that teams with higher Visibility, as defined in this thesis, have a higher potential to be successfully inner sourced. Teams can increase their project's Visibility by organizing coding events, workshops, and other activities that encourage collaboration, knowledge sharing and information

about their product and technical solutions. These activities will not only promote their projects but also provide opportunities for developers in different teams to interact and learn from each other.

In conclusion, IKEA has a strong foundation for adopting inner source practices due to the transparent and open development environment. On a developer team level, the key to successful implementation of inner source strategies lies partly in addressing the mentioned areas of improvement discovered in this thesis, documentation, test acceptance practices, PR management, and project visibility. By focusing on these areas, IKEA will be able to increase the success rate of developer teams wanting to inner source their development.

6. Conclusion

This thesis has addressed the problem definition described in 1.4, outlining 4 questions to better understand the software development processes within IKEA, and by interviewing 7 developer teams, determine how inner source practices can be assessed and potentially implemented.

1. *What are the general software development processes within IKEA?*

   The investigation into IKEA's general software development processes revealed a transparent and open development environment, with an emphasis on agile practices and DevOps methodologies. The teams are operating very much autonomously and are free to choose the languages, tools and practices when developing their product. This does however also lead to many different approaches, making it harder to compare the teams in a structured manner. The diversity also presents challenges for inner source adoption.

2. *What metrics from the repositories' metadata can be used to evaluate teams in regard to inner source?*

   Several metrics were identified through interviews with developers as potential indicators of inner source potential: (a) document availability, (b) GitHub Check failrate on merge, (c) visibility, and (d) time-to-close. These metrics can serve as a useful starting point for evaluating the interviewed teams and other of IKEA's developer teams' potential for adopting inner source practices.

3. *What conclusion can be drawn from the gathered metrics regarding the current state of software development?*

The analysis of the gathered metrics suggests that the current level of collaboration and co-development between teams at IKEA is limited. This observation implies that development teams may not currently be as open to external contributions as they could be. For instance internal acceptance criteria, as reflected by the GitHub Check failrate on merge metric, appear to be well-understood within the teams, but may not be as transparent to outsiders. Nevertheless, it's evident from the visibility metric that projects are being acknowledged by others within IKEA, indicating that there are ongoing efforts to enhance knowledge sharing across the organization.

4. *What recommendations can be given to IKEA consider future inner source initiatives?*

   This is answered in chapter 5.2.

6.1. Ethical aspects

A non-disclosure agreement was signed during the start of this thesis. To comply with the agreement, the supervisor of IKEA was made aware and agreed to the use of description of the organisation, teams and internal workings as presented in this report. The developers were made aware of this and accepted the premise that their answers during the interview were only to be used for the purpose of this thesis and within IKEA.

During the interviews, one developer with previous experience with open source talked about some of the challenges she perceived with inner sourced development. Not all developers are comfortable to share and invite others to co-develop their code. It can be because of fear of scrutiny, or that the developer is ashamed to share the code because it might have had to be written very quickly to keep up with project schedule. An implemented inner sourced development needs to take these aspects and the well-being of the developers into consideration.

7. Future work

This thesis work offers insights into IKEA's software development processes and potential for inner source practices, but also highlights areas for further exploration.

The focus was largely on repositories, thereby missing out on the communication and collaboration occurring on other platforms such as Slack and Google cloud platform (GCP)-groups, where much discussion also takes place. Also, IKEA used the documentation platform Confluence, which in this these could be used for more in-depth research about the state of the documentation of the teams. Future work could provide a more comprehensive view by integrating these platforms into the analysis. The broad filtering of the repositories can be improved to better reflect the functionalities of the teams.

The advanced functionalities of GrimoireLab could be harnessed in future work to expand upon the current metrics used. Grimorielab contains the functionality for integrations the API’s of Slack, Confluence and Jira, and could as a result be used as a platform for diverse continuous analysis of selected teams.

The Documentation availability metric could be improved upon by applying weights depending on the importance of the repositories, providing a slightly easier overview of the results on a team level.

For the visibility metric, getting the traffic and especially number of clones of repositories would be another variable to take into consideration when assessing reusability, similar to forks. Getting access to the traffic requires a personal access token with write-access, which wasn’t feasible in for the purpose of this thesis.

8. Terminology

Metadata - In the context of this thesis, metadata refers to information about the data in the repository, which could include details about commit history, contributors, pull requests, test results,

DevOps - DevOps is an approach to software development that integrates development (Dev) and IT operations (Ops), with much focus on the CI/CD (Continuous Integration/Continuous Deployment) for small, frequent updates to production.

GitHub Actions - Facilitates CI/CD-pipeline tasks in repositories

GitHub Workflow – Specifies autonomation jobs directly in the repository, can highly customised, often used to build, test and deploy code.

GitHub Checks – Feature that integrates with workflow to provide feedback from jobs.

9. References

[1] A. di Vaio, R. Palladino, A. Pezzi, and D. E. Kalisz, "The role of digital innovation in knowledge management systems: A systematic literature review" Journal of Business Research, vol. 128, pp. 220-231, Feb. 2021.

[2] H. Edison, N. Carroll, L. Morgan, and K. Conboy, "Inner source software development: Current thinking and an agenda for future research," Journal of Systems and Software, vol. 163, May 2020.

[3] Stol, K.-J., Avgeriou, P., Babar, M. A., Lucas, Y., & Fitzgerald, B. “A comparative study of challenges in integrating Open Source Software and Inner Source Software”. Information and Software Technology, pp. 1319-1336, Dec. 2011

[4] M. Capraro and D. Riehle, "Inner Source: Adopting Open Source Development Practices in Organizations”, Proceedings of the 38th International Conference on Software Engineering Companion, ICSE '16, pp. 472-475, May 2016.

[5] M. Capraro and D. Riehle, "Inner source definition, benefits, and challenge," ACM Computing Survey, vol. 49, no. 4, pp. 1-36, 2017

[6] Wan, Z., Xia, X., Zhang, Y., Lo, D., Zhou, D., Chen, Q., & Hassan, A. E. “What motivates software practitioners to contribute to inner source?” In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering pp. 132-144, Nov. 2022.

[7] R. van Solingen et al., "Goal question metric (gqm) approach," in Encyclopedia of Software Engineering, John Wiley & Sons Inc., 2002

[8] S. Dueñas et al., "GrimoireLab: A toolset for software development analytics," in Proceedings of the IEEE International Conference on Software Maintenance and Evolution, ,pp. 679-682, Sep. 2018.

[9] Elastic NV. (2023). Elasticsearch. Available: https://www.elastic.co/elasticsearch/ [Accessed: April 18, 2023]

[10] Elastic NV. (2023). Kibana. Available: https://www.elastic.co/kibana/ [Accessed: April 18, 2023]

[11] Docker, Inc. (2023). Docker. Available: https://www.docker.com/ [Accessed: April 16, 2023]

[12] Docker, Inc. (2023). Docker Compose. Available: https://docs.docker.com/compose/ [Accessed: April 19, 2023]

[13] GrimoireLab. (2023). GrimoireLab. Available: https://hub.docker.com/r/grimoirelab/grimoirelab/ [Accessed: April 19, 2023]

[14] Microsoft Corporation. (2023). Windows Subsystem for Linux. Available: https://docs.microsoft.com/en-us/windows/wsl/ [Accessed: April 19, 2023]

[15] GitHub, Inc., "GitHub REST API," GitHub Developer, 2021. [Online]. Available: https://docs.github.com/en/rest/. [Accessed: April 16, 2023].

[16] GitHub, Inc., "About Pull Requests," GitHub Docs, 2023. Available: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests. [Accessed: Apr. 25, 2023].

[17] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? A systematic mapping study on definitions and practices", Proceedings of the Scientific Workshop Proceedings of XP2016, New York, USA, 2016, pp. 1-11

[18] GitHub, "GitHub Actions Documentation," GitHub Docs, 2021. Available: https://docs.github.com/en/actions. [Accessed: April 13, 2023].

[19] GitHub, "GitHub Checks Documentation," GitHub Docs, 2021. Available: https://docs.github.com/en/rest/reference/checks. [Accessed: Apr. 13, 2023].

[20] L. Brunner, "GitHub Checks." GitHub. Available: https://github.com/LouisBrunner/checks-action/pulls?q=is%3Apr+is%3Aclosed, [Accessed: May 20, 2023]

[21] Microsoft Corporation, "Visual Studio Code Documentation," Visual Studio Code Docs, 2021. Available: https://code.visualstudio.com/docs. [Accessed: Apr. 19, 2023]

[22] T. Dybå and T. Dingsøyr, "What Do We Know about Agile Software Development?," in IEEE Software, vol. 26, no. 5, pp. 6-9, Sept. 2009.

[23] Atlassian Corporation Plc, "Confluence: Team collaboration software," Atlassian, 2021. Available: https://www.atlassian.com/software/confluence. [Accessed: 16-Apr-2023]

[24] Slack, Slack Technologies, San Francisco, CA, 2023. Available: https://slack.com/. [Accessed: May. 10, 2023]

[25] Izquierdo, D., & López, J. M., Managing InnerSource Project, InnerSource Commons 2018

[26] H. Kallio, A-M. Pietilä, M. Johnson, and M. Kangasniemi, "Systematic methodological review: developing a framework for a qualitative semi-structured interview guide" Journal of Advanced Nursing, May 2016

[27] CHAOSS, "About CHAOSS," 2023. Available: https://chaoss.community/about-chaoss/. [Accessed: May 14, 2023]

[28] "About GitHub Enterprise Cloud." GitHub Docs. Available: https://docs.github.com/en/enterprise-cloud@latest/admin/overview/about-github-enterprise-cloud. [Accessed: May 19, 2023]

[29] K. Beck et al., "Manifesto for Agile Software Development," Agile Alliance, 2001. Available: http://agilemanifesto.org/. [Accessed: May 10, 2023].

[30] K. Schwaber and J. Sutherland, "The Scrum Guide," Scrum Guides, 2020. Available: https://www.scrumguides.org/. [Accessed: May 10, 2023].

[31] M. Lage Junior and M. Godinho Filho, "Variations of the kanban system: Literature review and classification," Int. J. Prod. Econ., vol. 125, no. 1, pp. 13-21, May 2010.

[32] Atlassian Corporation Plc, "Jira Software," Atlassian, 2023. Available: https://www.atlassian.com/software/jira. [Accessed: May 14, 2023]

[33] J. Dumas, "Software Usability: Appropriate Methods for Evaluating Online Systems and Documentation," in Proceedings of the ACM SIGDOC Annual International Conference on Systems Documentation, pp. 69-77, Oct. 1990.

[34] R. Matulevicius, F. Kamseu, and N. Habra, "Measuring Open Source Documentation Availability", Proceedings of the International Conference on Quality Engineering in Software Technology. pp. 83-102. 2009

[35] M. Kreitz, “Security by design in software engineering,” SIGSOFT Softw. Eng. Notes, vol. 44, no. 3, p. 23, Nov. 2019.

[36] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub”, Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 805-816, Aug. 2015

[37] C. Vassallo, S. Proksch, A. Jancso, H. C. Gall, and M. Di Penta, "Configuration smells in continuous delivery pipelines: a linter and a six-month study on GitLab," in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 327–337. Nov. 2020

Appendix A: Extended Material

A1 Table showing initial interview guide

Goal	Question
1. Reduce silos	Q1: How do you handle external ideas or collaboration?
2. Reduce bottlenecks	Q1: What are the most common bottlenecks you encounter in your development? Q2: What steps do you take to avoid bottlenecks?
3. Improve knowledge sharing	Q1 How do you help new developers get acquainted with your repository/software?
4. Improve quality	Q1 What steps are you taking into consideration to make the process reusable?
5. Increased reusability	Q1 What steps are you taking into consideration to make the process reusable?
6. Increase development speed	Q1 What are the biggest parts of your work that slows down development? Q2 How do you make sure the tasks are on schedule?

A2) Configuration specification for GrimorieLab docker-compose

setup.cfg Specifies the configuration for the back-end data retrieval by Percival. It also configures the inclusion of necessary API tokens, and the default panels for the Kibana dashboards. In this thesis, following default back-end scripts were configured: Git, Github, Github2:issues, Github2:pull, and Github:repositories

projects.json Specifies desired repositories to be used. The repositories belonging to a single team and product are grouped together to facilitate easier Kibana visualization and analysis. For Git data, the GitHub token needed to be appended to the URL in the format "username:token@".