

Institute of Software Engineering
Software Quality and Architecture

University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Bachelorarbeit

Concept and Implementation of a Browser Extension for Gropius

Pia Wippermann

| | |
|-------------------------|---|
| Course of Study: | Informatik |
| Examiner: | Prof. Dr. Steffen Becker |
| Supervisor: | Dr.rer.nat. Uwe Breitenbücher, Sandro Speth, M.Sc. |
| Commenced: | May 3, 2021 |
| Completed: | November 3, 2021 |

Abstract

Context. Today, modern software systems are often component-based. In addition to numerous advantages, this also brings some disadvantages. One of them is that the teams involved often work in different issue management systems. Therefore, it is a challenge to communicate related issues across components in the different teams.

Problem. Although Gropius, as a cross-component issue management system, offers a solution to this problem, no company is currently using it. Therefore, it must be made easier for them to get started with Gropius.

Objective. Some of the functions of Gropius are to be integrated directly into the user's issue management system, with which the user is already familiar. The objective here is to create and evaluate the concept as well as a prototype of this idea.

Method. To do this, a browser extension is used to manipulate the display of webpages from issue management systems such as GitHub and Jira, and to integrate additional functionality from Gropius into it. The creation of a concept, as well as the evaluation of it and the prototype is carried out in close cooperation with stakeholders.

Result. In the evaluation of the elaborated concept, as well as the implemented prototype, it became clear that the idea is evaluated as helpful by the experts and that an integration of the Gropius functions into the issue management system used by the experts is desired.

Conclusion. The main contributions of this thesis are the detailed requirements, a concept and a prototype for the Gropius Browser Extension as well as the evaluation of the concept and the prototype through an expert survey.

Kurzfassung

Kontext. Moderne Software Systeme sind häufig komponentenbasiert, was viele Vorteile mit sich bringt. Dennoch sind damit auch einige Nachteile verbunden, wie zum Beispiel das Cross-Component Issue Management. Üblicherweise sind die integrierten Komponenten von verschiedenen Teams in unterschiedlichen Issues Management Systemen gemanaged, daher ist es oft herausvordernd abhängige Issues zwischen den Komponenten zu kommunizieren.

Problem. Obwohl Gropius als Cross-Component Issue-Management-System eine Lösung für dieses Problem bietet, wird es derzeit von keinem Unternehmen eingesetzt. Der Einsatz neuer Systeme in der Industrie, insbesondere wenn damit die Verwendung eines neuen Tools zusammenhängt, ist vor allem zu Beginn eher schwer zu realisieren. Daher muss den Unternehmen der Einstieg in Gropius erleichtert werden.

Ziele. Einige der Funktionen von Gropius sollen direkt in die Issue Management Systeme der Nutzer integriert werden, denn damit sind die Softwareentwickler vertraut. Ziel ist es hierbei vor allem das Konzept dieser Integration zu erstellen und im Anschluss daran zu evaluieren.

Methode. Es soll eine Browsererweiterung genutzt werden, um die Anzeige von Webseiten der Issue Management Systeme wie GitHub und Jira zu manipulieren und zusätzliche Funktionalitäten von Gropius in die jeweiligen Webseiten zu integrieren. Beim Ausarbeiten des Konzepts, sowie bei der Evaluation soll eng mit Personen zusammengearbeitet werden, die von Problemen mit Cross-Component Issues betroffen sind, und somit Hauptzielgruppe der geplanten Browsererweiterung sind.

Resultat. Bei der Evaluierung des erstellten Konzepts, sowie des Prototyps der Browsererweiterung für Cross-Component Issue Management wurde deutlich, dass die Idee von Experten als sehr hilfreich bewertet wird und eine Integration der Gropius-Funktionen in die von den Experten verwendeten Issue Management Systeme gewünscht wird.

Schlussfolgerung. Zusammengefasst ist der Beitrag dieser Bachelorarbeit zum einen detaillierte Anforderungen an die Browsererweiterung für Cross-Component Issue Management und zum anderen ein Konzept, sowie ein Prototyp der Browsererweiterung. Das Konzept und der Prototyp sind im Anschluss daran mit Experten durch eine Umfrage evaluiert.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 2 | Foundations and Related Work | 3 |
| 2.1 | Foundations | 3 |
| 2.2 | Related Work | 6 |
| 3 | Overview | 9 |
| 4 | Requirements Engineering | 11 |
| 4.1 | Planning of the Investigation | 11 |
| 4.2 | Ideation Phase | 13 |
| 4.3 | Prototyping Phase | 26 |
| 4.4 | Transformation Phase | 27 |
| 5 | Concept | 29 |
| 5.1 | Concept Overview | 29 |
| 5.2 | Concept Regarding the Collected Requirements | 30 |
| 5.3 | Reusable Components | 35 |
| 6 | Implementation | 39 |
| 6.1 | Architecture and Design of the Browser Extension | 39 |
| 6.2 | Implementation of the Gropius Browser Extension | 39 |
| 7 | Evaluation | 43 |
| 7.1 | Study Design | 43 |
| 7.2 | Results | 45 |
| 7.3 | Discussion | 49 |
| 7.4 | Threats to Validity | 50 |
| 8 | Conclusion | 53 |
| 8.1 | Summary | 53 |
| 8.2 | Benefits | 53 |
| 8.3 | Limitations | 54 |
| 8.4 | Lessons Learned | 54 |
| 8.5 | Future Work | 54 |
| | Bibliography | 57 |

List of Figures

| | | |
|------|---|----|
| 4.1 | The Living Lab approach [RSH09]. | 11 |
| 4.2 | Adapted Living Lab approach. | 12 |
| 4.3 | Component diagram of scenario application PhotoPia. | 16 |
| 4.4 | Mockup for sub-goal (SG3) | 16 |
| 4.5 | Rating scale for (RE4) | 17 |
| 4.6 | Companies of the interviewed employees. | 19 |
| 4.7 | Overview of the frequency the interviewed employees are confronted with such a problem of cross-component issues. | 21 |
| 4.8 | Overview of the rating of the individual sub-goals by the Gropius experts. | 22 |
| 4.9 | Overview of the rating of the individual sub-goals by the employees from the industry. | 22 |
| 4.10 | Rated goals by the employees from the industry. | 23 |
| | | |
| 5.1 | Overview of the concept for the Gropius Browser Extension. | 30 |
| 5.2 | First approach to add a component to a Gropius project. | 31 |
| 5.3 | First approach of the Gropius project list. | 31 |
| 5.4 | Second approach of the Gropius project list. | 32 |
| 5.5 | Creating a component for a Gropius project. | 32 |
| 5.6 | Component is added successfully to a Gropius project. | 32 |
| 5.7 | First approach of the related issue list. | 33 |
| 5.8 | Active Gropius projects and by which the related issues shown are filtered. | 34 |
| 5.9 | Option to select the active Gropius projects by which the related issues shown are filtered. | 34 |
| 5.10 | Mockup for selecting a Gropius project. | 35 |
| 5.11 | Mockup for selecting a component for a Gropius project that was selected before. | 36 |
| 5.12 | Mockup for selecting an issue. | 37 |
| | | |
| 6.1 | Standard cross-browser architecture. | 40 |
| 6.2 | Messaging between the components of a browser extension based on [Chra]. | 40 |
| | | |
| 7.1 | Average rating of (Q1) | 45 |
| 7.2 | Average rating of (Q2) | 45 |
| 7.3 | Average rating of (Q3) | 45 |
| 7.4 | Average rating of (Q4) | 46 |
| 7.5 | Average rating of (Q5) | 46 |
| 7.6 | Average rating of (Q6) | 46 |
| 7.7 | Average rating of (Q7) | 47 |
| 7.8 | Average rating of (Q8) | 47 |
| 7.9 | Average rating of (Q9) | 47 |
| 7.10 | Average rating of (Q10) | 48 |
| 7.11 | Average rating of (Q11) | 48 |

7.12 Average rating of (Q12). 48

Acronyms

- API** Application Programming Interface. 3
- CCIM** Cross-Component Issue Management. 2
- CCIMS** Cross-Component Issue Management System. 3
- CSS** Cascading Style Sheets. 5
- GQM** Goal Question Metric. 43
- HTML** Hypertext Markup Language. 5
- HTTP** HyperText Transfer Protocol. 5
- IDE** Integrated Development Environment. 4
- IMS** Issue Management System. 1
- UI** User Interface. 1
- UML** Unified Modeling Language. 4
- URI** Uniform Resource Identifier. 5

1 Introduction

Distributed component-based architectures are a very common way of structuring software systems nowadays. The system is then usually developed by several independent teams, which apart from the advantages can also lead to various problems, as identified by Mahmood et al. [MNH15]. Among other things, one problem is that these teams often use different Issue Management System (IMS) which only work on a project specific scope. Consequently, it is very difficult to communicate cross-component issues effectively. For example, bugs that occur in the interface of a component spread to the components that consume that interface. Such issues must somehow be traced back to its original component, because fixing that bug depends on fixing the bug in the original component.

Speth et al. provided a solution for it which is presented through the work of Gropius [SBB20]. Gropius is an integrated cross-component issue management system. The purpose of this system is that all issues of the components of a project can be modeled and managed using a graphical notation in Gropius in a web application. Furthermore, issues can be linked across components and changes to issues are passed on to the original IMS and are synchronized with them. The problem is that currently companies and development teams do not use Gropius. The introduction of a new system is a lengthy process, especially in large companies, and getting them to use Gropius will be probably difficult at first. Another problem that may arise with Gropius as a separate tool is that the developers, product owners, and all other users of IMSs are used to GitHub, Jira, GitLab, etc. Experience shows that the openness towards the use of a new tool is rather low, especially if you are currently using a working one. Therefore, we need a solution that serves as a kind of bridge between Gropius and the IMSs and facilitates the start with Gropius.

To make it easier to start using Gropius, some selected functions of Gropius should be integrated directly into the user's IMS. This means that those affected do not have to get used to a completely new system straight away, but rather their familiar system is enhanced with some Gropius functions. To achieve this, the integration of Gropius must be as natural as possible and the use must disrupt the user's workflow as little as possible. The integrated functions should be selected in such a way that, in the context of the component, the user is provided with insight into systems in which the respective component is used. This also makes it possible to reduce the view offered in Gropius to a complete project, to a component and its dependencies, and to display these in the corresponding IMS. One objective of this thesis is to find out whether these stated goals are seen as useful by stakeholders who are to be identified. In addition, further requirements are gathered for what functions of Gropius should be integrated into the User Interface (UI) of the IMSs. This then leads to the next objective, which is to design a concept and implement a prototype based on the gathered requirements.

When integrating additional functionality into an existing website that is displayed in a web browser, it is necessary to manipulate the way the website is displayed and its functionalities. For this purpose, a browser extension is suitable that allows you to just do that. In our case, this means

changing the way IMSs like GitHub and Jira display their webpages in web browsers to add additional functionality from Gropius using a browser extension. Furthermore, A detailed plan is being developed on how to collect and evaluate requirements for the Gropius Browser Extension. After a concept and a prototype have been designed on this basis, this is evaluated with the help of an expert survey. Here, the main focus is on the usefulness of the integrated Cross-Component Issue Management (CCIM) in the IMS in general, but also specifically on the evaluation of the implemented prototype.

A large part of the interviewed employees from the industry, already had problems with cross-component issues that are managed in different IMS. From the results of the interviews in requirements engineering as well as from those in the evaluation it can be seen that the concept of the Gropius Browser Extension is evaluated as very useful and that besides the separate UI of Gropius an integration of the functions into the used IMS is desired.

In summary, the contribution of this thesis is the elaboration of a concept for an integration of selected functions regarding the management of cross-component issues managed within different IMS. This concept is developed in close cooperation with stakeholders and therefore detailed requirements are gathered. Furthermore, a prototype for the Gropius Browser Extension will be implemented and the concept as well as the prototype will be evaluated by an expert survey.

Thesis Structure

This thesis is structured as follows:

Chapter 2 – Foundations and Related Work: This chapter is on the one hand about methods and concepts that need to be understood for this thesis and that are re-used and on the other hand existing work in this area is presented.

Chapter 4 – Requirements Engineering: The entire requirements engineering process is covered here. This includes both the planning and conduct, but also the evaluation thereof.

Chapter 6 – Implementation This chapter elaborates the design and architecture of prototype for the Gropius Browser Extension and describes its implementation.

Chapter 7 – Evaluation: In the last phase of this thesis, the concept and the prototype of the Gropius Browser Extension is evaluated. The planning, conduct, and the results of it are presented in this chapter.

Chapter 8 – Conclusion Here the summary and future work of this thesis is shown.

2 Foundations and Related Work

In this chapter, the foundations and the related work of this thesis are shown. For this purpose, the foundations are first listed and explained in Section 2.1, followed by the related work in Section 2.2 which is divided into the literature research methodology in Section 2.2.1 and existing browser extensions for IMSs.

2.1 Foundations

In this section, the foundations of this thesis are listed and explained one after the other, which are relevant to understand this thesis. On the one hand, software issues in general are discussed in Section 2.1.1 while IMSs are described in Section 2.1.2. On the other hand, the Cross-Component Issue Management System (CCIMS) Gropius is explained in Section 2.1.3. Furthermore, information on the query language for Application Programming Interface (API) can be found in Section 2.1.4, followed by the foundations regarding web browsers and web browser extensions in Section 2.1.5 as well as the design of user interfaces in Section 2.1.6.

2.1.1 Software Issues

Software issues can be used to track ideas, feedback, bugs, etc. for work in a software project [Atl][Gita]. Such an issue consists of a title, a description, an author and optional team members who are responsible for the issue. The structure of an issue varies depending on the IMS in which it is managed. It can also contain additional information regarding the label or the importance of the issue for example. Software issues can be created and managed within an IMS by different people such as developers, project owners, or stakeholders of the project.

2.1.2 Issue Management Systems

Software issues that are described in Section 2.1.1 are managed within so called issue management systems which are client-server applications. There are a variety of IMSs such as Github¹, GitLab², Jira³ or Redmine⁴. IMSs „(...) help to manage issue reporting, assignment, tracking, resolution, and archieving“ [BVGW10]. Stakeholders can communicate via the IMS, including not only the developers of the software, but also for example customers, project managers, and quality assurance

¹<https://github.com/>

²<https://about.gitlab.com/>

³<https://www.atlassian.com/software/jira>

⁴<https://www.redmine.org/>

personnel. Johnson et al. [JD03] summarized the basic features of IMSs, which include, for example, the classification of issues in terms of priority and category of the issue and the modification of issues by users.

2.1.3 Gropius

Gropius, introduced by Speth et al. [SBB20], is a solution approach for managing cross-component issue relationships within complex software systems that consist of several independent components. Hence, it is a CCIMS that allows developers to model components and their dependencies of a software system and manage cross-component issues. A metamodel for cross-component issues has been defined by Speth et al. [SBB21].

Initially, a Gropius project must be created for a software system that is to be managed in Gropius, to which the individual components must then be added. These components need a link to the IMS in which the component's issues are managed as well as a link to their repository system. To show the inter-dependencies of the components of a software system, a graphical notation similar to a Unified Modeling Language (UML) component diagram is used. After a project manager or software developer creates a Gropius project or has been added to one, components can be created and then linked to each other through interfaces. It is also possible to create issues for a component and link them to issues of other components or interfaces that are affected by the created issue. Gropius forwards these issue changes to the original IMSs and synchronizes with them and thus acts as a wrapper over the IMSs used by users to manage issues that affect multiple components or interfaces.

Gropius is divided into front-end, back-end and databases and there are adapters for the individual IMSs that connect the back-end of Gropius to the original IMSs of the components. Back-end and front-end communicate via GraphQL which is described in Section 2.1.4. In addition to the Gropius front-end, an Integrated Development Environment (IDE) extension that is called Gropius-VSC has been developed by Speth et al. The extension Gropius-VSC „(...)“ aims to provide a focused component-specific view of cross-component issues while keeping the cross-component issue management features(...)“ [SKBB21] in contrast to the Gropius front-end.

2.1.4 GraphQL

GraphQL is a query language for APIs. Customers can request exactly what they need there. Therefore, applications that use GraphQL are fast and stable because they can control exactly what data is retrieved. Types are used „(...)“ to ensure that apps only ask for what's possible and provide clear and helpful errors⁵.

⁵<https://graphql.org/>

2.1.5 Web Browser and Web Browser Extensions

A web browser „(...) is an application running on top of the operating system of the computer“ [Aie18]. Web browsers allow to request resources on the web which are identified by a unique Uniform Resource Identifier (URI). This interaction is done via the file exchange protocol HyperText Transfer Protocol (HTTP). The web browser receives Hypertext Markup Language (HTML) documents, which are then displayed to the user in a human-readable form. HTML is a markup language that is rendered by the web browser to display the web page. It describes the content of a web page. Cascading Style Sheets (CSS) describe how the content of a web page should look and thus describes the design of the page. Commands can be embedded in HTML files via JavaScript, which „(...) is a loosely typed, prototype-based, object-oriented programming language“ [Aie18]. These commands are interpreted by the web browser and instructions can be executed by it at page interpretation time.

With browser extensions it is possible to modify and enhance the functionality of a web browser and thus change the display of a web page in the corresponding web browser to add additional functions [DG09][Moz]. They are available for a wide range of popular web browsers and run within the context of the web browser [Meh16]. Such browser extensions consist of different cohesive components and they „(...) can include background scripts, content scripts, an options page, UI elements and various logic files“ [Chrb]. Browser extensions are based on the web technologies HTML, JavaScript, and CSS. More technical details can be found in ?? where the implementation of the Gropius Browser Extension is described.

2.1.6 User Interface Design

For the Gropius Browser Extension it is essential that the inserted elements integrate smoothly into the UI of the IMSs. The relevance of a good UI and the principles of UI design are addressed in [BEZ08]. Two examples of a good user experience are therefore easy-to-use and intuitive. The paper mentions some design principles, one of which is to recognize and consider the effects of known interface conventions. Furthermore, the eight golden rules of UI design by Shneiderman et al. [SP10] are also relevant for this bachelor's thesis. When designing a UI the following rules should be considered:

- Strive for consistency.
- Seek universal usability.
- Offer informative feedback.
- Design dialogs to yield closure.
- Prevent errors.
- Permit easy reversal of actions.
- Keep users in control.
- Reduce short-term memory load.

2.2 Related Work

This section addresses the related work of this thesis. Initially, the research methodology is described in Section 2.2.1, followed by the related work found in Section 2.2.2 which is about existing browser extensions for IMSs.

2.2.1 Literature Research Methodology

When searching for related work for this thesis, the search engine Google Scholar⁶ was used. First, keywords are collected which were related to the topic and then these were entered into the search engine, first individually and then in suitable combinations. For all results, which were then delivered, the first ten entries were considered. The individual entries were checked for their relevance to this bachelor's thesis by first looking at the title, whereby some entries could already be excluded. The abstract was then read for each of the remaining entries, whereupon some entries could be excluded again. After that, the results classified as relevant were analyzed in detail and, following the principle of snowballing, the sources referenced there were analyzed according to the same principle.

When collecting the keywords relevant for this bachelor thesis, they were divided into categories and within these categories all possible combinations were then entered into the search engine. The first category was about software issues and their management and included the following keywords:

- software issues
- issue management system
- issue tracking
- cross component

The second category was then about browser extensions for IMS and cross-component issues.

- browser extension
- cross component issues
- software issues
- chrome extension

Regarding browser extensions for IMS and cross-component issues, no relevant work could be found in this area except for Gropius [SBB20], the metamodel for cross-component issues [SBB21], and the IDE extension Gropius-VSC [SKBB21], all of them by Speth et al., which was already mentioned in the foundations in Section 2.1. Consequently, research was subsequently conducted in the search engine Google⁷. As in the scientific search engine Google Scholar the first ten entries

⁶<https://scholar.google.com/>

⁷<https://www.google.com/>

were considered when searching for the keywords. Often, overviews of existing browser extensions were offered by the websites, whereupon the suggestions listed there were researched separately in the Google search engine. Partly browser extensions were also linked within the website, this link was then followed. The following keywords were used for this research:

- browser extension
- chrome extension
- GitHub
- cross component
- issue

2.2.2 Browser Extensions for Issue Management Systems

There are already a large number of browser extensions for IMSs. In the following, some of the found browser extensions for the IMSs GitHub as well as Jira are discussed.

GitHub

GitHub itself offers an overview of browser extensions that can be installed and thus the functionalities regarding GitHub can be personalized [Gitb]. As this list is very long it was also filtered by the keywords „cross component“ and „issue“. While the filter „cross component“ had none hits, the filter by „issue“ resulted in three hits. The first is the „github-mention-highlighter“⁸ which highlights the current user’s username when mentioned in an issue, for example. The next one is „hubnav“⁹ that enables navigating GitHub via keyboard shortcuts. This includes opening issues of a repository via a keyboard shortcut. The last one that was found is „github-show-avatars“¹⁰. This extension shows large avatars for example in issue lists to recognize quickly who created the issue. Obviously, none of the extensions mentioned address the problem of managing cross-component issues that are managed in different IMSs.

One of the most popular extensions for GitHub is ZenHub ¹¹. ZenHub is integrated directly into GitHub’s interface and it offers some more project management solutions right on top of GitHub. For example, just like GitHub, ZenHub offers a board that displays all of a developer’s issues, but unlike GitHub, they don’t have to be tied to a repository or organization. Shared workspaces from different repositories can be created so that all participants have a complete overview of the workspace. Nevertheless, only GitHub repositories can be added to these workspaces [Zena; Zenb] and thus it does not provide a solution for cross-component issues managed in different IMS.

⁸<https://github.com/benbalter/github-mention-highlighter>

⁹<https://github.com/cheshire137/hubnav>

¹⁰<https://github.com/matthizou/github-show-avatars>

¹¹<https://www.zenhub.com/>

There are a number of other browser extensions for GitHub, but they are not presented here. A large part of the browser extensions do not affect the issues managed in GitHub at all, and if so, then only in terms that are not relevant for this thesis. No browser extensions could be found that provide a solution approach for cross-component issues managed in different IMS.

Jira

As for GitHub, numerous browser extensions already exist for the IMS Jira and they are mainly aimed at saving time. An example of a Chrome Extension for Jira is JIRA Assistant¹². This extension allows you to manage issues within the Google Chrome browser with just one click. Nevertheless, this does not address linking issues within Jira, nor does it address linking issues between independent components managed in different IMSs.

In [LLm+21], some helpful issue creation extensions for Jira are presented. These include Zephyr Capture¹³, which allows you to create and annotate screenshots and share them. Another extension that is listed there is the JIRA Template Injector¹⁴ that provides an issue description template for each type of issue that can be created. Finally, Google to Jira is mentioned, which allows users to create, edit and comment on Jira issues via Gmail and Calendar. None of the found browser extensions for Jira address the problem of CCIM.

¹²<http://www.jiraassistant.com/>

¹³<https://www.atlassian.com/de/software/jira/capture>

¹⁴<https://jiratemplate.com/>

3 Overview

This chapter describes the chronological sequence of this bachelor's thesis and thus provides an overview of the following chapters.

At the beginning of this thesis, the requirements engineering was conducted. A lot of time was invested in the preparation of the requirements engineering. This primarily involves selecting a method for the rough procedure in this process and then planning the individual phases in requirements engineering. As described in detail in Section 4.1, which deals with the planning of the investigation, the principle of the Living Lab was chosen. It follows that requirements engineering consists of the ideation phase, followed by the prototyping phase, and the transformation phase. Once this selection was made, the individual phases could first be adapted to our needs and then planned in detail. As part of the requirements engineering, several interviews were conducted with employees of established, medium-sized, as well as smaller companies to gather requirements for the Gropius Browser Extension. More precise details are described in Section 4.2, Section 4.3 as well as in Section 4.4.

Chapter 5 then goes into more detail about the concept of the Gropius Browser Extension. The concept was designed step by step and further developed with the influence of the interviewed stakeholders. This was realized with the help of a first prototype and was accordingly carried out at the same time as the prototyping phase began. The prototype was again part of the requirements engineering and was implemented coinciding with the ideation phase and subsequently included in the stakeholder interviews. Thus, the interviews, the prototyping phase, and the creation of the concept were conducted simultaneously.

After all interviews were finished and a concept was designed, the next step was to implement the actual prototype. This prototype differs from the first prototype in that functionalities have been integrated. In fact, the first prototype was only a conceptual design and therefore only a superficial prototype without any functionality. More on the first prototype in Section 4.3. It also served to validate the collected requirements. The implementation of the second prototype is described in Chapter 6.

The last phase of this thesis, which started after the implementation phase, is about the evaluation of the second prototype and the concept of the Gropius Browser Extension which is described in Chapter 7. Here, interviews were again conducted with some of the employees interviewed in requirements engineering to gather feedback.

4 Requirements Engineering

The first phase of this thesis is to collect the requirements for the planned Gropius Browser Extension. This chapter describes the exact process and results of the requirements engineering. First of all, the planning of the investigation is discussed in Section 4.1. Subsequently, the individual phases of requirements engineering, as they were envisaged in the planning, are described. In the ideation phase, this includes the preparation of the survey described in Section 4.2.1, the conduct of the interview described in Section 4.2.2, and the evaluation of the interviews described in Section 4.2.3. Afterwards, the prototyping in Section 4.3 and the transfer phase in Section 4.4 will be discussed.

4.1 Planning of the Investigation

First of all, we had to plan how to proceed to gather the requirements for the Gropius Browser Extension. This was planned based „Requirementsengineering und -management“ by Rupp [RSH09]. There are many different planning approaches to requirements engineering, such as the Living Lab. Living Lab is especially useful when it comes to collaboratively driven inquiry and is particularly useful when designing a creative new product. A setting is created that comes as close as possible to the reality of life in which the later system is to be used. Stakeholders are involved in the development process and are therefore fully-fledged development partners. Figure 4.1 depicts a model of the Living Lab approach, which is divided into three phases, the ideation phase, the prototyping phase and the transfer phase.

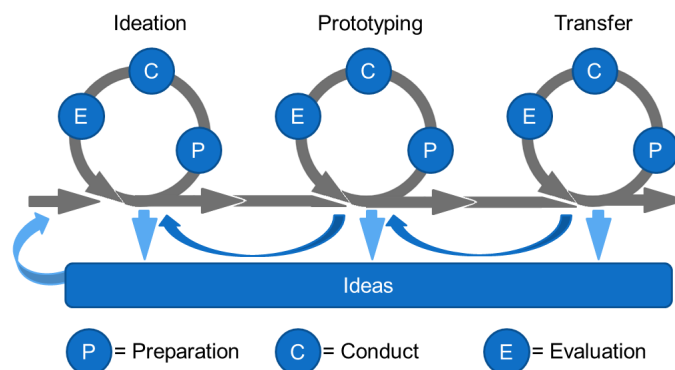


Figure 4.1: The Living Lab approach [RSH09].

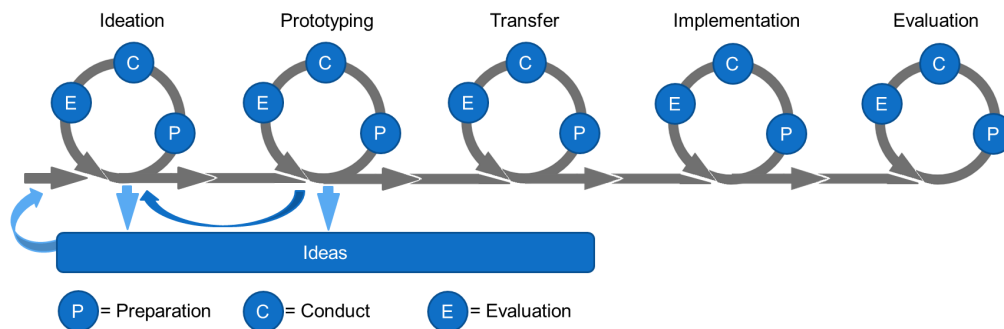


Figure 4.2: Adapted Living Lab approach.

There are already several modified versions of the Living Lab, such as the light and medium variants, in which phases are omitted and also for our case it has to be modified a little bit. We do not want to create a completely new creative end product, but it still makes a lot of sense to work closely with a wide range of stakeholders. The exact selection of stakeholders can be found in Section 4.2.1. Due to the fact that we want to integrate new functionalities into existing websites, thus modifying their UI, it is very possible to create a realistic environment without much effort. This makes it possible to show stakeholders concrete examples of integrated CCIM into the used IMS as early as possible. Thus, the first three phases can be adopted pretty much as they are envisioned in the Living Lab. However, since in this bachelor's thesis the final product will also be a prototype of the Gropius Browser Extension, we decided to add two more phases. Figure 4.2 depicts the model of our approach.

The first phase is the ideation phase that is described in Section 4.2. It is divided into three subphases. Subphase number one, described in Section 4.2.1, is about the planning, subphase number two, described in Section 4.2.2, about the conduction, and subphase number three, described in Section 4.2.3, about evaluating the gathering of requirements. One part of the ideation phase is determining the objects of investigation. These are, on the one hand, the sources of requirements, which is defined in the first subphase. The techniques of determination of requirements as another part of the objects of investigation are also determined in this subphase, as well as the target and product vision. Last but not least, we have the requirements themselves, which are determined in the third subphase of the ideation phase when the gathering of requirements is evaluated. This is done by deriving detailed requirements from the original requirements collected in the second subphase. After performing the three subphases, the first phase is finished and the second phase, called the prototyping phase, begins. Within the second phase we have implemented an initial design prototype to validate the requirements. How exactly this first prototype looks like and how it is developed can be found in Section 4.3. The third phase is then intended to evaluate the requirements collected in the ideation phase in terms of technical feasibility. The current status of the Gropius back-end is there of particular relevance to us, as it determines which requirements can be implemented in a fully functional way at this time. Last but not least, the two further phases to be carried out as part of this bachelor's thesis are the implementation and the evaluation. In the implementation

phase, the previously elaborated concept for the Gropius Browser Extension is to be implemented as far as possible. More on this phase in Chapter 6. After this is done, the concept as well as the implemented second prototype is to be evaluated in the last phase as described in Chapter 7.

After planning the general procedure in requirements engineering and the determining the objects of investigation was completed, a meeting was held with Prof. Dr. Steffen Becker. The described method as well as the objects of investigation elaborated in the section, including the detailed planning of the interviews, were presented in the meeting in order to obtain feedback on them and make possible improvements before the interviews began. To ensure that the presentation, which is intended to introduce the topic before the questions are asked, is understood by all, it was decided at this meeting to first conduct a test survey with a student worker from a company. Possible suggestions for improvement should then be discussed again and possibly adjustments made. The result of the test interview is presented in Section 4.2.2.

4.2 Ideation Phase

Phase one of three of the requirements engineering is discussed in this section. It deals with the ideation phase and its three subphases. These include the preparation in Section 4.2.1, the conduct in Section 4.2.2, and the evaluation in Section 4.2.3.

4.2.1 Preparation

In this subsection, a distinction is made between the elaboration of the objects of investigation and the elaboration of the interviews. Therefore, it is divided into two parts, firstly the elaboration of the objects of investigation which are the sources of requirements, the investigation techniques, and the product vision as well as the goals of the Gropius Browser Extension and secondly the elaboration of the interviews.

Objects of Investigation

In the preparation of the gathering of requirements, the first step was to identify the sources of requirements. Thus, it was a matter of identifying the stakeholders who should be involved in the requirements elicitation process. This involved Gropius experts as well as employees from the industry, since they are the main target group for the planned Gropius Browser Extension. In addition to the stakeholders, Gropius with its functionalities is another a source of requirements.

After the sources of requirements were worked out, the next step was to plan the investigation techniques. Following Rupp [RSH09], identification techniques were selected to determine stakeholder requirements. According to that, one should select a main investigative technique for this purpose, which can then be combined with supporting techniques. The investigative techniques mentioned are interview techniques, observation techniques, artifact-based techniques, and creativity techniques. In the first step, we chose to use interviewing techniques for this purpose, since the artifact-based and observation techniques in particular are not suitable in our context. Creativity techniques are also less suitable as a main investigative technique in our context, but rather as a supporting one. Therefore, we have decided to go with an interview. Initially, it was

planned to conduct the interviews as smaller group interviews, on the one hand the group consisting of the Gropius experts and on the other hand the group consisting of employees from the industry. In the end, the employees from the industry were interviewed individually, but more on this in Section 4.2.2. Another investigative technique used based on the analysis of the existing Gropius system is requirements anticipation. The principle of anticipating requirements is to generate them based on information about the system. In our case, some requirements were already guessed before the interviews with the stakeholders started, in order to validate them during the interviews. These anticipated requirements also had a significant influence on the formation of goals and visions, which are described below. But first to the anticipated requirements, which are the following user stories:

As a developer I want ...

- (US1) to see the meta data graphically displayed in the description of an issue created by Gropius.
- (US2) to have displayed a list of related issues for the issues in my component.
- (US3) to be able to see the type of relation between two issues.
- (US4) to link the issue I created with existing issues of other components.
- (US5) to create the issue I created for my component for another component as well and link those issues with each other.
- (US6) to add an issue that I find and that affects my component to my component within the IMS of the original issue.
- (US7) stored test cases to be executed automatically when a related issue is marked as solved and I am notified about the result.

The next step was to define the main objectives as another part of the objects of investigation. As noted above, the ones we have worked out are based primarily on the anticipated requirements. Our product vision was thus created which is „**Integrated context aware issue relationship management for independently managed software systems**“. We have further worked out three sub-goals, which are:

- (SG1) Discover and react
- (SG2) Create and link directly
- (SG3) Display existing issue relations

Sub-goal (SG1) refers to the user story (US6). When discovering an issue that affects another component reacting within that IMS should be possible and add this issue to another component. Sub-goal (SG2) refers to (US4) and (US5). When creating an issue it should be possible to link this issue directly. The third and last sub-goal refers to (US1), (US2), and (US3).

Elaboration of the Interview

Now that we had identified the objects of investigation, the product vision, as well as decided on the investigation techniques, the interviews had to be planned in detail. The rough procedure is that the stakeholders are first introduced to the topic with the help of a presentation and then asked questions. The elaboration of the presentation as well as the questionnaire is described below.

At the beginning of the presentation, before the stakeholders were presented with the developed product vision and the sub-goals, they were first introduced to the topic with two scenarios. This should also facilitate understanding of the vision and sub-goals. With the creation of the scenarios, another investigative technique was incorporated into the interview that was mentioned by Rupp [RSH09]. The scenarios we have worked out for this purpose are based on a component-based system of a photo app. Figure 4.3 shows the component diagram on which the scenarios are built. The scenario application is called PhotoPia. For the scenarios the component's issues should be managed in different IMS. We chose to have the Photo App component's issues managed in GitHub, the User Management component's issues in Jira, and the Image Filter Service 1 component's issues in GitLab. For the two following scenarios, only the three components mentioned were relevant, which is why the issue management of the other components is not discussed. The first scenario was based on a failed login of a user of the Photo App. As a developer of the photo app, you look for a bug that triggers the failed login in Jira, where the User Management component's issues are managed. It should be pointed out then that the current situation is that one would have to manually add the found bug in the IMS GitHub and possibly add a link to the original bug in the description. The problem is that the issues are not really linked to each other and that this way is very inconvenient. The solution to this problem is that we provide the ability to add this issue directly to the component Photo App in the IMS Jira. This scenario is intended to illustrate sub-goal **(SG1)** to the stakeholders. The second scenario was based on a feature request on an image filter. The approach here is to create a feature request for the component Photo App in GitHub that is known to be dependent on a feature request in, for example, Image Filter 1 Service. So currently you would also need to create a feature request in GitLab for the Image Filter 1 Service and add the corresponding link to GitHub. Here, too, we are dealing with the same problem. Our second sub-goal **(SG2)** is presented to stakeholders as a solution to this. In this case, it means that when we create the feature request, we want to be able to create it for another selectable component as well. To introduce sub-goal **(SG3)**, Figure 4.4 was included in the presentation, which is a mockup of a section of the component diagram that illustrates the issue relationship and is integrated into the GitHub interface. After the introduction by the scenarios, the sub-goals were then finally presented and assigned to the scenarios. Another introduction point, which was to be presented to the stakeholders, are the Gropius projects. For this purpose, it should be explained on the basis of the Photo App that first a project must be created, in which the components are added, and only then the linking of issues is possible.

After the presentation, which lasted about ten minutes and introduced the stakeholders to the problem and the initial goals, the actual interview was to begin. Therefore, we have developed a questionnaire consisting of seven questions or prompts, which on the one hand deal with the topic of CCIM in general and on the other hand ask for the desired features of the Gropius Browser Extension. Several types of questions were included in the creation of the questions, which are discussed in detail below. The following questions or prompts were developed for the questionnaire, with the seventh and last question **(RE7)** serving to prioritize the features collected in **(RE6)** and thus could not be formulated exactly in advance.

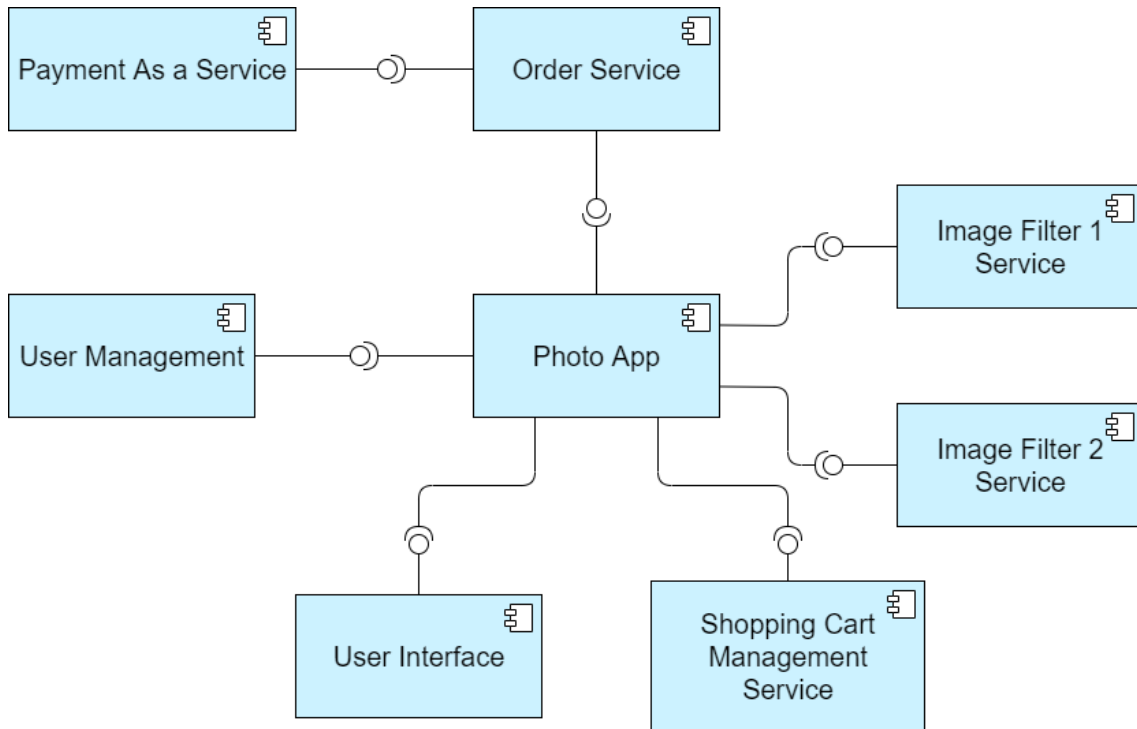


Figure 4.3: Component diagram of scenario application PhotoPia.

Failed to login #1

Open PiaWippermann opened this issue 4 days ago · 0 comments

PiaWippermann commented 4 days ago · edited ▾ Owner

No description provided.

PiaWippermann added the bug label 4 days ago

The mockup shows a simplified component diagram with 'Photo App' and 'User Management' components connected by a line. Below each component is a folder icon containing a red gear with a white lightning bolt, representing a bug.

Figure 4.4: Mockup for sub-goal (SG3).

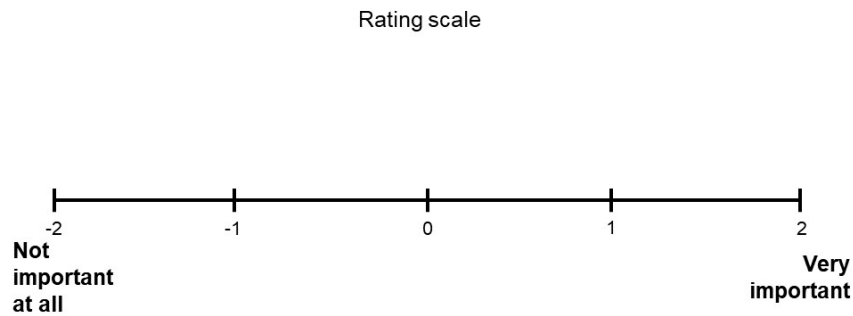


Figure 4.5: Rating scale for **(RE4)**.

(RE1) Can you think of a situation in which you had a similar problem with cross-component issues as it was shown in the scenarios?

(RE2) How often do they encounter such a problem with cross-component issues?

- never
- rarely
- quarterly
- monthly
- weekly
- daily

(RE3) Are there any special cases or problems that need to be dealt with explicitly?

(RE4) Please rate the sub-goals **(SG1)**, **(SG2)**, and **(SG3)** that were presented with the scale shown in Figure 4.5.

(RE5) Do you think it makes sense to have a separate interface for the CCIM, or do you prefer to have this tied into the IMS that you use?

(RE6) Which features should be integrated by the Gropius Browser Extension into the UI of the IMS used?

(RE7) Prioritization of the collected features.

(RE1) is an open-ended question that asked for another scenario from the stakeholders and should cause the stakeholders to really get into the topic of cross-component issues and because of that, be able to better answer later questions about desired features. **(RE2)** is related to the first question and as there are six offered response options it is a closed question while **(RE3)** is again an open-ended question. In the next question which is **(RE4)** the principle of ranking was included. Based on Porst [Por13], an appropriate scale was selected to evaluate the goals. The purpose of the scale is to use it to reasonably evaluate the importance of the individual sub-goals. Hence, the characteristics of the scale are in a relational relationship to each other and the distances between the scale points are equal when evaluating the goals. For this reason, we decided to use an interval scale with numeric interval points. The interval points are not described verbally, but have numbers and only the end

points are labeled „not important at all“ and „very important“, this is also called an endpoint-named scale. Endpoint-named scales should have between five and nine scale points and as only three goals should be assessed, we chose five for this. Thus, with five interval points, we have a scale zero point, so respondents have the option to rate a goal neutrally. Such a scale zero point is considered critical when it is feared that the respondents do not have a clear opinion on the topic and the neutral rating thus becomes an escape category. Since this is not a concern, the option to consciously choose neutral evaluation was included. In summary, an endpoint-named scale was chosen with five interval points that can be seen in Figure 4.5. The next question which is **(RE5)** refers to the problem we suspect that Gropius will not be accepted as a separate tool at first. For this reason, we decided to add a question to support this statement. Since it was assumed that stakeholders would not be able to answer this question with a clear yes and no, but would want to differentiate between scenarios, this is an open-ended question. The following question **(RE6)** is the essential question in requirements engineering, as it asks for the desired features that the Gropius Browser Extension should be capable of, and is therefore again an open question. Finally, the requirements just mentioned should be prioritized as it is mentioned in **(RE7)**. For this purpose, the technique of topological sorting was used. Requirements should always be compared in pairs and thus a ranking of the requirements in terms of their importance gradually emerges. Since different requirements are mentioned by the various stakeholders, this cannot be formulated as a fixed question. The questions on this arise from the answers to **(RE6)**.

As mentioned above, a distinction was made between two different groups of stakeholders, the Gropius experts and the employees from the industry. Although there are obvious knowledge differences regarding Gropius and cross-component issue management, there was little difference between the presentations of the two groups. Only the presentation for the Gropius experts should briefly point out the current problem that it will be difficult, especially at the beginning, to convince companies and developers to use a completely new tool for CCIM. As mentioned above, this assertion is verified with the help of **(RE5)** that is asked to the stakeholders.

4.2.2 Conduct

In the second sub-phase of the ideation phase, the planned interviews are conducted with the stakeholder groups. In these interviews the original requirements are collected which can then be processed to detailed requirements that are described in Section 4.2.3. This subsection only describes the rough procedure of the interviews while in Section 4.2.3 the answers to the questionnaire are summarized.

First of all, a test interview was conducted with a working student of the company Novatec Consulting GmbH to obtain feedback on the presentation and the structure of the questions as it was decided in the meeting with Prof. Dr. Steffen Becker. The result was that the presentation conveys the problem in an understandable way and thus provides a good introduction to the subsequent questions, which were also evaluated as clearly formulated.

After the successful test interview, the first group interviewed was the Gropius experts. Part of this group was Prof. Dr. Steffen Becker, one of the supervisors Sandro Speth, Niklas Krieger who is one of the back-end developers of Gropius, and Sarah Stieß. Here, the first two questions about a situation similar to the one in the scenario and the frequency of occurrence of such a problem were

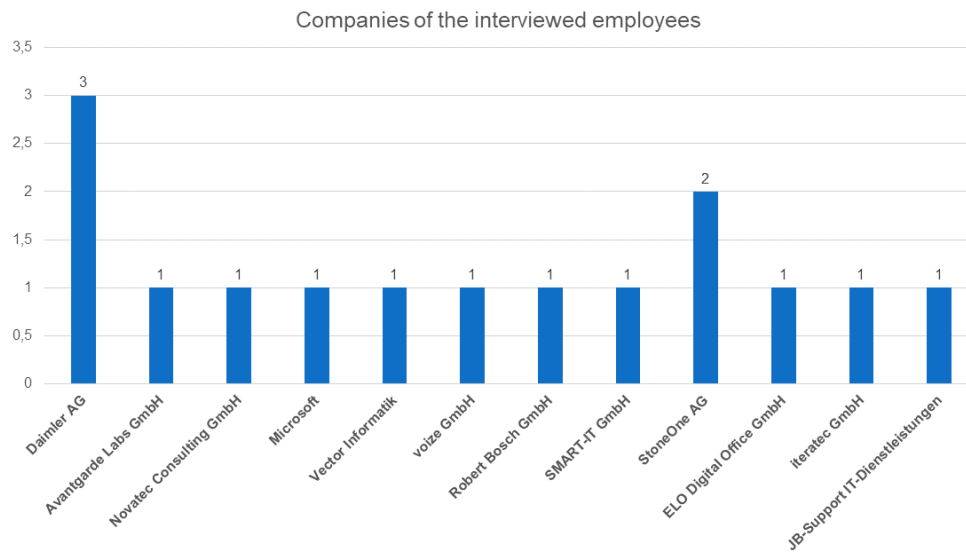


Figure 4.6: Companies of the interviewed employees.

answered primarily by putting oneself in the role of another developer, so that the questions were answered from different perspectives. In contrast, these questions were answered employees from the industry from their own perspective as developers.

In order to find employees from the industry to participate in the interviews, seventeen employees from different companies were contacted. Fifteen of them responded and agreed to be interviewed. They belong to the companies Daimler AG, StoneOne AG, Microsoft Corporation, Vector Informatik GmbH, Robert Bosch GmbH, SMART-IT GmbH, Avantgarde Labs GmbH, Novatec Consulting GmbH, voize GmbH, ELO Digital Office GmbH, and JB-Support GmbH. Figure 4.6 shows the exact distribution of company affiliations. As described in Section 4.2.1, the interviews with the employees of the companies were planned to be conducted as group interviews. In the end, however, these were conducted as individual interviews, as this allowed individual opinions and suggestions to be addressed more intensively. This also made it easier to ensure that the subject matter was understood and that developers who had not yet dealt with the issue were able to give their opinions and feedback. The first four surveys were conducted exactly as planned to gather additional requirements. Simultaneously with the interviews, we already started to develop the first prototype, which was used to test the requirements as described in Section 4.3. A detailed description of how the prototype was further developed based on this can be found in Section 4.3. This prototype was used in the next interviews before the question of what other features the Gropius Browser Extension should cover. Requirements that had already been collected could thus be prioritized and processed before the developers could then contribute their own ideas. This was an important step in refining the requirements, which are discussed in more detail in Section 4.2.3, and in figuring out how to integrate the collected features into the IMS GitHub.

4.2.3 Evaluation

The evaluation phase involved summarizing the original requirements, which were directly mentioned by the stakeholders, deriving the detailed requirements from the requirements originally elicited in the interviews, as well as evaluating the stakeholders' responses to the questions. The answers to the questions are illustrated with diagrams, and only the answers of the companies' employees are included in them. Therefore, the subsection is divided into two parts, namely the analysis of the answers to the questionnaire and the derivation of the detailed requirements.

Analysis of the answers

Starting with the evaluation of the answers to the questions posed to the stakeholders. In answer to the first question (**RE1**) it has already been predicted by the Gropius experts that a large proportion of developers will have to deal with such a problem very frequently. Especially when it comes to front-end and back-end components, to open source components or in general in microservice architectures. Depending on this, such a problem may occur more or less frequently. When it comes to external libraries, you can probably expect such a problem about every month, but with microservices it might happen much more frequently. You could expect daily or even hourly problems here. These statements were also confirmed by the employees from the industry. Thirteen of the fifteen stakeholders interviewed reported having experienced such a problem with cross-component issue management. Various scenarios were mentioned that happened in the same way or similar to the scenario presented. Two people generally indicated that this problem occurs primarily in the context of front-end and back-end components, as well as in the microservices environment, and that they have experienced this frequently. Another stated that this was mainly the case with regard to support and development tickets. As one approach to solving the problem, one respondent named that they have implemented a unified IMS within the company that is used without direct repository assignment to work around this problem. However, this does not solve the problem that open source-components that are used are managed in other IMS. In addition to that, new companies with which one cooperates seldom use the same IMS, so it is not possible to determine exactly which uniform IMS would make the most sense. In other companies, different IMS are used in different business units of a company. One respondent indicated that there is a central IMS and individual components derive issues from the central IMS and manage them in a separate IMS. There is one employee who is only responsible for the synchronization between the IMS. But not only between different IMS such problems with cross-component issues occur, but also within an IMS as one respondent indicated. In his case it is about GitLab. Figure 4.7 shows the results of question (**RE2**), the distribution of how often such a problem occurs among the interviewed employees of the companies. It is clear to see that more than half of the respondents have such a problem at least monthly and in the interviews it became clear that all of them who are confronted with it urgently desire a solution to this problem. The evaluation of the following question (**RE3**) starts with the group of Gropius experts. The focus was on possible problems and special cases seen by the interviewees. Firstly, it was mentioned that components managed in non-public IMS cannot be added to Gropius projects. Another problem that regards to browser extensions are changes in the interface of the IMS websites. With such changes, the elements that are integrated by the extension can no longer be included in the previously planned location. Thus, in any case, you must first check whether the element you want to edit exists. In addition, it must be regularly reviewed whether there are changes to the interfaces of the IMS and, if necessary, the

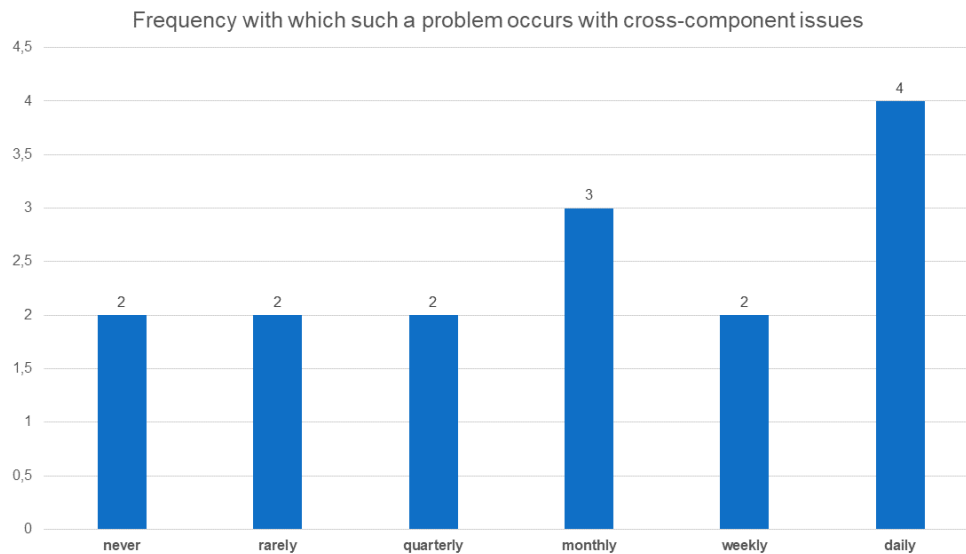


Figure 4.7: Overview of the frequency the interviewed employees are confronted with such a problem of cross-component issues.

integration of the features by the extension must be edited. For the employees from the industry, the problems identified differed depending on how frequently the problem occurs for each respondent and which IMS is used. The two respondents who had never experienced this problem said that they doubted that the effort required to integrate this functionality would be disproportionate to its use. However, the percentage of respondents who have this problem frequently, and the enthusiasm in finding a solution, is very high, with thirteen out of fifteen respondents. One problem that was mentioned by four of the respondents, especially in relation to the IMS Jira, is that it is already very complex to use and you have a lot of options when you create a new issue. Integrating new features would only make it more complex to use Jira and this would lead to many not using the features. This is also the reason why they felt that only the most necessary should be included in the IMS and that integration should be as fluid as possible. How exactly this should look then was covered in the question regarding desired features. There is one more problem, which was mentioned in the interview that is the rights management. The question here is who is allowed to access which IMS and specifically who is allowed to access which issues. It should be possible to specify this for safety-critical components in particular. The subsequent point (**RE4**) was the request to rank the presented sub-goals (**SG1**), (**SG2**), and (**SG3**) on the scale described in Section 4.2.1. First, Figure 4.8 shows how the sub-goals were ranked by the group of Gropius experts. The rating of each sub-goal was discussed by the respondents and a collective rating was given at the end for each sub-goal. In Figure 4.9, the rating of the individual sub-goals by the interviewed employees can be seen. 60% of them rated (**SG1**) and (**SG3**) as 2 which means that these sub-goals are very important to the majority of them. (**SG2**) on the other hand, was rated as 1 by 60% of the employees which in turn means that this sub-goal is important to the majority. The average rating of each sub-goal is shown in Figure 4.10. It is clearly evident that the average rating of the three sub-goals is very similar. The average rating of (**SG1**) and (**SG3**) is exactly the same at 1.333, (**SG2**) is rated as slightly less important on average at 2.367. The next question posed to the employees

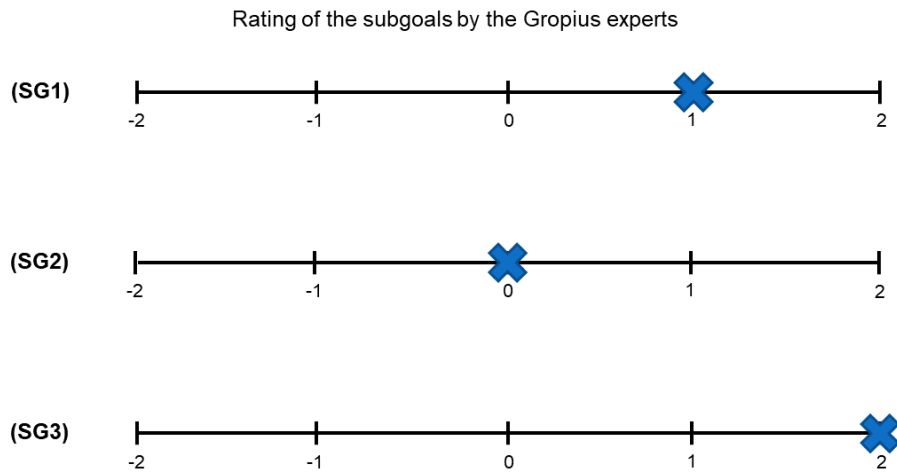


Figure 4.8: Overview of the rating of the individual sub-goals by the Gropius experts.

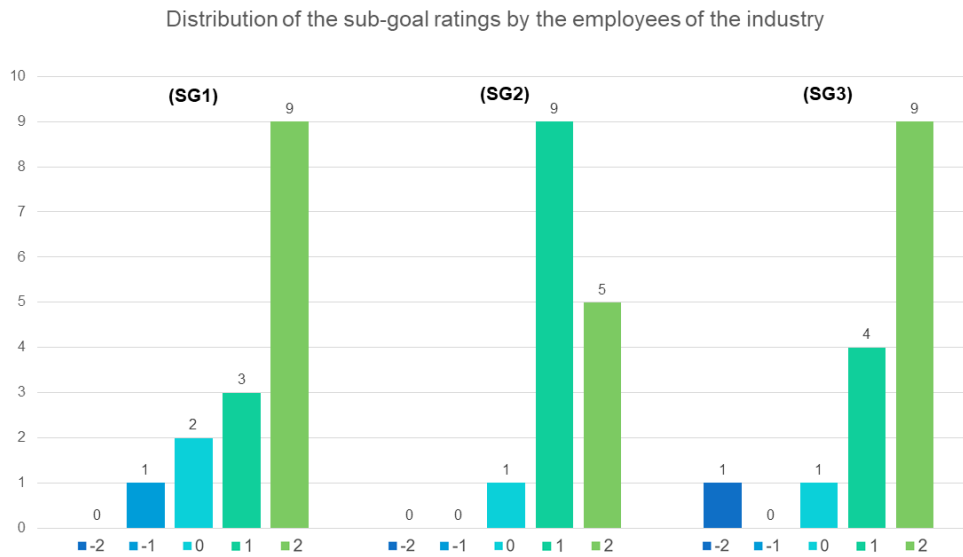


Figure 4.9: Overview of the rating of the individual sub-goals by the employees from the industry.

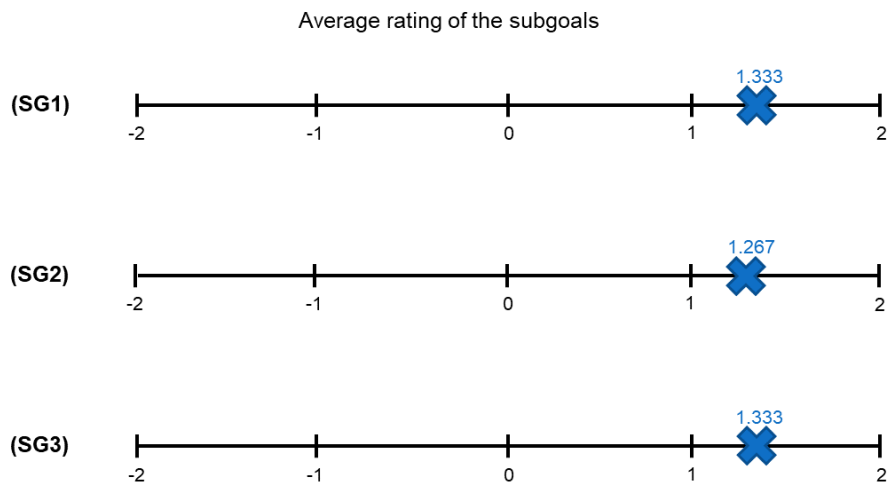


Figure 4.10: Rated goals by the employees from the industry.

from the industry (**RE5**) was whether or not a separate interface for Gropius made sense. There were different opinions on this. The majority, eleven of the fifteen respondents, believe that the integration in the IMS is a priority and that a separate interface is an additional „nice to have“. Acceptance of a new tool is rather difficult, they said, because training for the employees is needed and people are familiar with using IMSs like GitHub, Jira, etc. But a separate tool is useful for team leaders and product managers, five of them said, or as a kind of dashboard so you have a top-down view, which three others commented on. Another said that full integration of features into the IMS would be optimal, but that this would make the interfaces too complex. For this reason, he said, a separate interface makes sense as an add-on, yet as much as possible should be integrated into the IMS. One respondent was specific about what should be integrated into the IMS and what should be managed through a separate interface. Accordingly, the creation of issues and issue relations as well as the viewing of relations should be integrated into the IMS, while the creation of projects and components as well as their interfaces should be regulated via the separate interface. Yet another is of the opinion that only existing issue relations should be displayed in the IMS, everything else should be regulated via the separate interface, because this makes the interfaces of the IMS too complex. Only one of the respondents believes that the integration of features into the IMS is not useful and everything should be handled through the separate interface. Next, the question regarding the desired features of the Gropius Browser Extension that is asked for in (**RE6**) is evaluated. As described in Section 4.2.2, the group of Gropius experts, as well as the first four employees of the companies, were asked directly about desired features. The remaining respondents were presented with the prototype with some implemented features and asked to rate them and after that they had the chance to propose their own ideas. The following is a summary of all respondents' statements about desired features. On the one hand, it was mentioned that a fast login to Gropius should be made possible. Then there should be a button in the IMS that allows to add the component managed in this IMS to a Gropius project, as well as a list showing in which Gropius projects the component is included. In addition, the dependencies of an issue should be displayed in the corresponding

page of this issue. Whenever an external issue is displayed, the website of this issue should be called when clicking on such an issue. For each issue, a list of artifacts as well as SLOs should also be presented in the IMS as well as the labels of an issue should be imported. Requirements were also mentioned regarding the graphical representation of component dependencies as well as issue relations. Firstly, the entire component diagram of a Gropius project is to be displayed, as it is done in the Gropius Front-End. Clicking on a component should then link to the component's respective IMS. Secondly, a section of the component diagram is to be shown to illustrate the dependencies of a single issue. Only components that are included in the issue relationship of this issue should be shown. Another requirement regarding the component diagram is reusability. If possible, a library should be selected that can also be reused in the Gropius front-end. In terms of design, this is to be adapted to that of the respective IMS, for example, there should also be a dark and light mode for GitHub. Icons are also to be set to match those in the Gropius front-end. Since a component can be contained in several Gropius projects, you must be able to set an active Gropius project. Concerning this active project, the issue relations should then be displayed. Switching this active project back and forth must be as simple and fast as possible. The prioritization that is subsequently requested in (RE7) is incorporated into the prioritization of the detailed requirements, which are discussed below.

Derivation of the detailed requirements

Thus, the evaluation of the questions from the interviews is now complete and the next step is deriving detailed requirements from the original requirements that have just been pointed out. This was achieved in several steps. First, the requirements were separated, necessary ones were extracted, these in turn were abstracted and supplemented, and subsequently refined and improved. The method of deriving the detailed requirements from the collected original requirements also comes from Rupp [RSH09]. The result of this process was a tree of requirements, with the leaves representing the detailed requirements for the system. The starting point was therefore the original requirements collected in the interviews. The gathered requirements were subsequently separated and abstracted to create the nodes of the requirements tree. After that these abstracted requirements were refined and improved. Within this process also some of the anticipated requirements that were not mentioned in the interviews were inserted. In the following, the requirements tree is presented as a list of user stories. The detailed requirements presented are prioritized from top to bottom based on the last question asked of respondents.

As a developer I want to ...

- manage issue relationships within the IMS
 - link issues within the IMS
 - * select project first and then component to link my issue to one of the component's issues
 - * add an issue to another component and link these issues with each other
 - * link an issue to an existing issue of another component when creating this issue
 - * create an issue also for a selected component and link those issues when creating this issue

-
- * be automatically reminded to link the issue when inserting a URL link of another issue
 - see the dependencies of an issue in the IMS
 - * have a list of related issues
 - * see the descriptions of linked issues
 - * see the status of linked issues
 - * have the ticket number of the linked issue displayed, if available
 - * be able to view the types of the issue relationships
 - * have matching icons for the issues
 - * see the meta data graphically displayed in the description of an issue created by Gropius
 - be able to set a filter for projects for which I want to manage the issue relationships at the moment
 - end issue relationships within the IMS
 - have an ease of use and fluidity of integration into the IMS used
 - an easy and fast authentication via the popup
 - have the design adapted to the IMS used
 - have as much information as necessary but as little as possible integrated into the IMS
 - regulate as little as possible via the popup, but have all functionalities directly integrated into the IMS
 - I would like to manage Gropius projects within the used IMS
 - add a repository as a component to a Gropius project
 - * select a project of a list of all my projects I want to add the component
 - * filter the project list via the input
 - * create a new project
 - * define the interfaces to other components
 - see in which Gropius projects the component is included
 - have component diagrams included in the IMS
 - see the complete component diagram of the project the component is in
 - be able to zoom in on the component diagram to refine the view
 - see an excerpt of the component diagram corresponding to the issue which I can show and hide via a dropdown

- open the corresponding IMS when clicking on a component in the shown component diagram
- certain features when the status of a dependent issue is changed
 - be notified if the status of a depending issue is changed
 - automatically stored test cases to be executed when the dependent issue is marked as solved and I am notified about the result
- set additional information for an issue
 - specify plannedTime and spendTime to an issue
 - set the prioritization of an issue
 - set a list of SLO's for every issue
 - set a list of artifacts for every issue
 - set the version number from which the change of the status of the issue is included
- have the additional information for an issue displayed in the IMS
 - see plannedTime and spendTime to an issue
 - see the prioritization of an issue
 - see the list of SLO's for every issue
 - see the list of artifacts for every issue
 - see if a changed status already affects the current live version
- have a dashboard integrated into the IMS showing the percentage of issues in relation to their prioritization

4.3 Prototyping Phase

In this phase, the first prototype of the planned extension for Gropius was to be implemented. The purpose of this phase was it to examine the elaborated requirements and find out the best way to integrate the new functionalities into the IMS GitHub. Therefore we have developed a first horizontal prototype [Zdr00], that is one that should represent the wide range of functionalities that the Gropius Browser Extension should implement, but without back-end support. It was focused on the fact that the design is already adapted to the interface of GitHub in order to get detailed feedback on the integration from the stakeholders. The prototype is thus non-functional but high fidelity.

This first prototype was used for the interviews to visualize some of the already gathered requirements. Stakeholders were asked to provide their feedback regarding the feature itself, as well as how this was integrated into the GitHub interface. From this, conclusions could be drawn regarding usability and design, thus advancing the development of the concept of the Gropius Browser Extension. Therefore, this first prototype was implemented in several iterations. Stakeholder feedback was implemented in each iteration, both improving existing features and adding new ones. The final version was then used to develop a general concept that would apply not only to the IMS GitHub,

but also to other IMS. The elaboration of this concept can be read in Chapter 5. In general, however, it can be said that all features implemented in the prototype were rated as useful by all respondents to whom it was presented.

4.4 Transformation Phase

Now that the interviews have been completed, the next step is to analyze the requirements in terms of technical feasibility. The progress in the back-end of Gropius plays the decisive role here, which is why meetings were held with Niklas Krieger, one of the back-end developers of Gropius, to clarify which requirements can already be implemented in a fully functional way. Some of the requirements cannot be implemented as it is desired at the moment due to the current status of the Gropius back-end. Nevertheless, all the requirements collected are technically feasible in principle.

5 Concept

In this chapter, the elaboration of a concept for the Gropius Browser Extension is explained. Initially, an overview over the concept for the Gropius Browser Extension is given in Section 5.1. After that, in Section 5.2 the concept regarding the individual features the Gropius Browser Extension should be capable of regarding the requirements engineering is presented. The detailed concept has gradually evolved with the development of the first prototype. The first prototype was developed for the IMS GitHub, so the figures illustrating the development of the concept are based on GitHub. Nevertheless, the concept is also always generalized so that this is also valid for any IMS.

5.1 Concept Overview

First of all, an overview of the concept for the Gropius Browser Extension is given in this section. It will be explained on the basis of the scenario presented in requirements engineering that was about the „PhotoPia“ project. For the sake of simplicity, the project is reduced to the three components User Interface, User Management and Photo App. The Gropius Browser Extension is primarily intended to enable the management of cross-component issues within the UI of the common IMS. Linking issues between independently managed components of a software system is handled by the Gropius back-end. Nevertheless, the goal of the Gropius Browser Extension is that stakeholders managing cross-component issues feel that everything is running in the context of the IMS they are using. As an example, we have the three components from the Gropius project PhotoPia. In all three components an issue is stored, which are related to each other. The issue of User Management is related to the issue of the Photo App, and this in turn is related to the issue of the User Interface. For each component managed in an IMS, a component is created in the Gropius back-end, and the associated issues also exist in the Gropius back-end and are synchronized with the original issues. With a browser extension it is possible that the web browser, in which the website of the IMSs is displayed, is extended by additional functionalities and the way a website is displayed can be modified. The dependencies of the issues, which actually only exist in the Gropius back-end, can therefore be displayed and managed in the IMS UI. Figure 5.1 depicts the graphical representation of the described concept. Abstractly, this means that dependencies of issues of one component to issues of other components managed in different IMSs can be displayed and edited via the UI of the corresponding IMS. Dependencies exist only in the Gropius back-end, but with the help of browser extensions that can access the functionalities in the Gropius back-end, they can also be managed via the IMS UI.

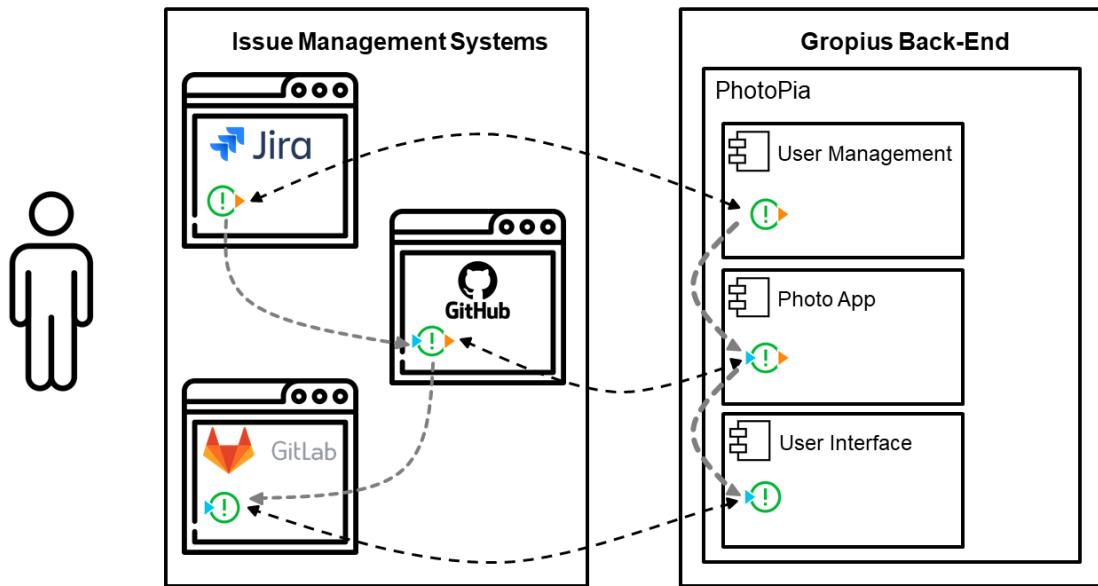


Figure 5.1: Overview of the concept for the Gropius Browser Extension.

5.2 Concept Regarding the Collected Requirements

In general, fourteen of the fifteen interviewed employees of the companies believe that the integration of the new functionalities into the IMS through the Gropius Browser Extension should be designed in the style of the respective IMS. Only one thought that this should be conspicuously in a different style, so that it would be clear that an additional function was integrated here. Accordingly, it is decided to adopt the style of the respective IMS. This also makes more sense in terms of the eight golden rules of design by Shneiderman et al. [SP10], so striving for consistency is one of them. A large part of the features that will be integrated into the IMSs are actions that the user can trigger, which consist of several steps. Following to Shneiderman et al., it is particularly important that the user is provided with informative feedback that makes it clear where the user currently is. Therefore, a brief but informative description should be given for each step. Another rule of the eight golden rules, which was particularly taken into account when creating the concept, is the design of a dialog to yield closure. Every action that is triggered by the user has a goal and this should be communicated to the user when the goal is reached. As the users go through the steps to perform the action they triggered, an easy reverse of the actions should be permitted. For this reason, in almost every step there should be a button in the dialog header that allows you to return to the previous step. Thus, another rule according to Shneiderman et al. was considered.

The first feature that has already been implemented within the first iteration of the prototyping phase is a button to add a component to a project that one can select and a list of projects in which the component is integrated. In Figure 5.2 you can see the first approach on how a component can be added to a project and in Figure 5.3 the first one of displaying the project list. Eleven of the fifteen respondents were asked to rate this feature integration. All of them found the function very helpful. However, the second person who was shown the prototype noted that the button for adding the component to a selectable project should be better placed under the list of projects.

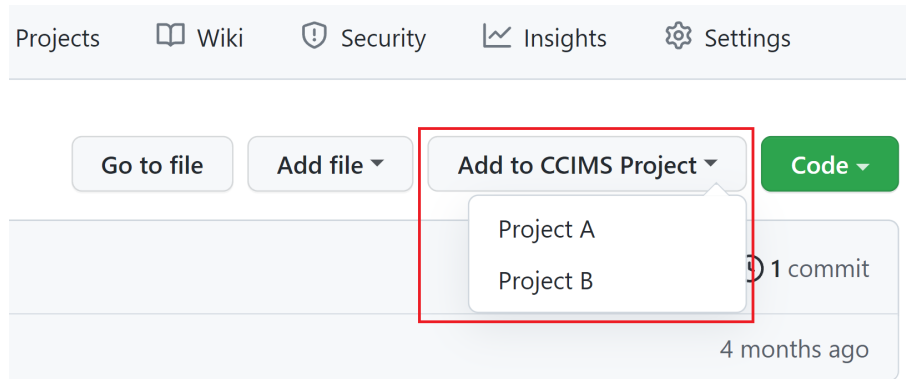


Figure 5.2: First approach to add a component to a Gropius project.

Gropius Projects

PhotoPia
AlbumPia

Figure 5.3: First approach of the Gropius project list.

Of the nine respondents who were presented with both approaches, all felt that the button below the list of Gropius projects was better. In Figure 5.4 you can see the final version of the project list and the option to add the component to another Gropius project. Generally, this means that the list of Gropius projects and the button for adding a component to a Gropius project should be implemented in one place. Clicking the button then opens a popup that guides you step-by-step through adding the components to a Gropius project. Step one is the selection of the project and step two is the specification of the interfaces to other components. Since selecting a project is relevant to multiple features, the concept of that is described in Section 5.3.1. Step two is then creating a new component in Gropius for the selected project. When creating a new component name, repository URL, and description can be inserted by the user. Here, the default name is the name of the current repository and the repository URL is the current URL. In Figure 5.5, the mockup for this is shown in the IMS GitHub. After the component is then created, the dependencies to other components in the project must be specified by defining interfaces. The most user-friendly way to do this here is via a graphical interface. Components and their interfaces of the project should be displayed in form of a component diagram and the interfaces to the newly added component should be able to be added graphically. When the component is now finally added to the project a success message is shown to the user. A mockup for that is shown in Figure 5.6.

In addition to managing cross-component issues, the graphical representation of component dependencies within a system in the IMS used is seen as a major advantage of the Gropius Browser Extension, as confirmed by the Gropius experts as well as by the companies' employees interviewed. Thematically, this fits best with the overview of Gropius projects, in which the current component is

Gropius Projects

PhotoPia

[Add Component to a Gropius Project](#)

Figure 5.4: Second approach of the Gropius project list.

Figure 5.5: Creating a component for a Gropius project.

included. All eleven respondents who were shown the Gropius project list considered the approach to be good that when clicking on a displayed Gropius project, a popup should open showing the component diagram of the respective system. This in turn should support zooming in and out, and when clicking on a component, the IMS associated with the component should open in a new tab.

The second feature that was implemented within the first iteration of the prototyping phase is a list of related issues for an issue in the IMS. With the first iteration, only the list was displayed and there was no functionality when clicking on the edit button as you can see it in Figure 5.7. The feedback on this was very clearly positive from all nine respondents who were shown the prototype. Editing and adding new issue relationships should be possible via the edit button in the case of GitHub, as confirmed by all of them. In general, this means that the management of issue relationships in the style of the respective IMS should be integrated with the list of related issues. When adding a

Figure 5.6: Component is added successfully to a Gropius project.

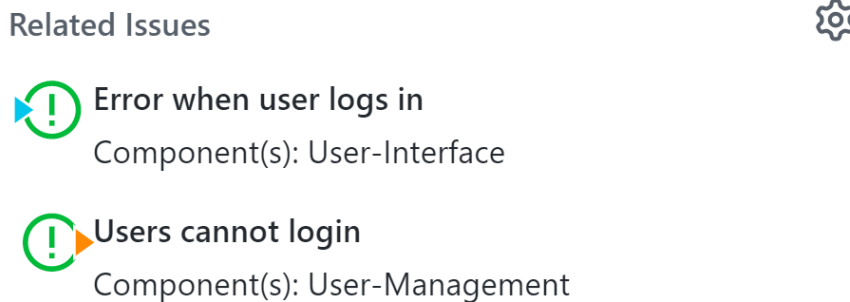


Figure 5.7: First approach of the related issue list.

new issue, you must first select the project in which you will select a component for which you will create the issue. The concept of selecting a project is described in Section 5.3.2 and the one for selecting a component in Section 5.3.2. After these two steps are done the issue can be added or selected to the corresponding component and the issue relationship can be specified which is described in Section 5.3.3. This makes it possible to insert issue relations after an issue is created and thus sub-goal (SG1) is addressed. Another feature that should be included in the list of related issues is the ability for the user to specify for which issues they would like to be notified when their status is changed. One of the interviewed employees who mentioned this feature also suggested adding an icon in the form of an eye behind each issue. These icons should initially be dim gray, which means the notification for this issue is not active, and when you click on this icon, it should turn dark gray, which means the notification is active.

Another feature that should illustrate the issue relation of the current issue is the section of the component diagram. This was already shown to the respondents in the form of a mockup that is shown in Figure 4.4 in the presentation and should be evaluated again in the prototyping phase. Feedback on this was also consistently positive, although one respondent noted that this should be added to a drop-down menu so that people can optionally hide the chart. However, the position below the description of the issue was felt to be appropriate by all. When clicking on a component, the corresponding IMS should be opened and when clicking on the displayed issue, the corresponding website of the issue of the component should be opened.

Last but not least, the first iteration also tackled the design of the Gropius Browser Extension popup. On the one hand, the login to Gropius should be possible via the popup. After the user logged in, the first approach in the first iteration of the prototyping phase was to be able to set active projects via the popup. Since it is possible that a component is contained in several Gropius projects, it must be possible to specify with respect to which Gropius project the information about the relationships of an issue is to be seen. The initial idea was that only one project can be active at a time and you can simply switch back and forth between projects. Immediately from the first to whom the feature was shown in the prototype, it was noted that it would be better if more than one project could be active. This assumption was also confirmed by all other ten respondents. Another noted that it is very inconvenient to be able to set active projects only through the popup. As this is outside the actual environment this could be quickly neglected. His suggestion for improvement was to add

Active Gropius Projects



PhotoPia

Figure 5.8: Active Gropius projects and by which the related issues shown are filtered.

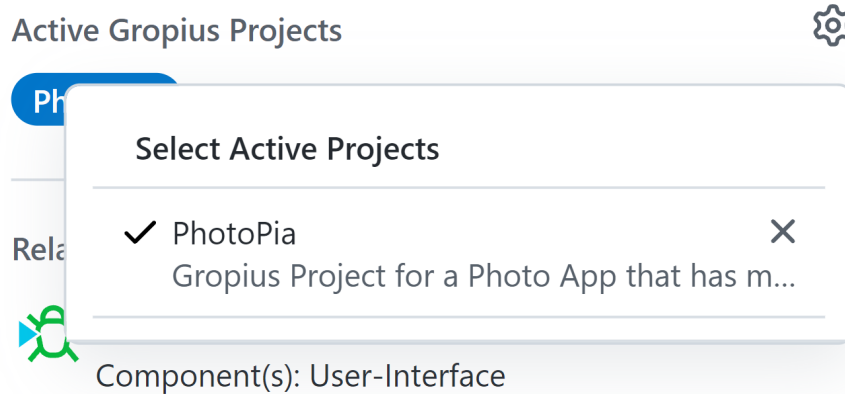


Figure 5.9: Option to select the active Gropius projects by which the related issues shown are filtered.

a new item in the sidebar of GitHub to set the active projects. In concrete terms, this means that a filter for Gropius projects should be included directly in the UI of the respective IMS, which would then apply both to the display of the sections of the component diagrams and to the displayed related issues list. All other seven respondents to whom this alternative was suggested also felt that this was better. Figure 5.8 shows the item in the sidebar of GitHub that shows which Gropius projects are active at the moment. For GitHub, it was decided that the selection of active Gropius projects would be implemented in GitHub style to maintain consistency in the UI, as this is one of the principles in designing a UI according to Shneiderman et al. [SP10]. A mockup of this is shown in Figure 5.9. Since it has now been decided that filtering of active Gropius projects will not be done via the popup, general information will instead be shown in the popup when the user is logged in. This includes a list of all Gropius projects that the user has created or added to and their associated components. For each component, a list of issues associated with that component should be displayed, which the users can then filter by the issues to which they are assigned to.

Another requirement to be considered in the concept is our second sub-goal (SG2) from requirements engineering, namely „create and link directly“. For GitHub, this case does not need to be considered separately, as the GitHub UI looks almost the same when creating a new entry as it does when viewing an existing entry. This means that when an issue is created, the item in the sidebar that shows the issue relationships can already be displayed and in this process the issue can be created and linked directly for another component. In general, however, the same concept applies here, that first a project must be selected, then the corresponding component and then you can select an existing issue of this component or create a new issue. Afterwards, only the issue relation has

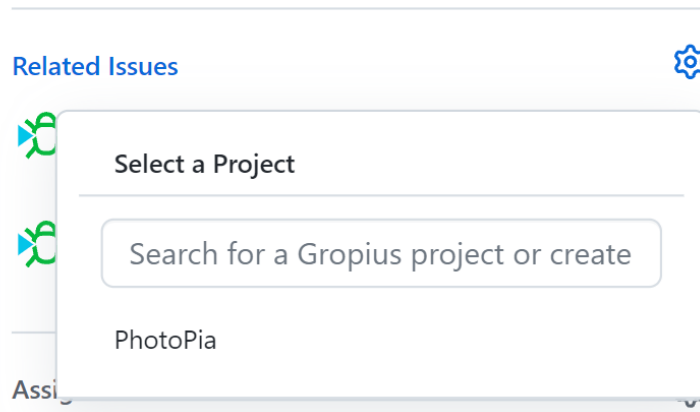


Figure 5.10: Mockup for selecting a Gropius project.

to be specified. The concepts of these steps are described in Section 5.3.1, Section 5.3.2, and Section 5.3.3. The exact integration into the different IMS has to be decided individually so that the feature is fluently integrated into the UI.

5.3 Reusable Components

Some functionalities are required by several of the features described above, so a concept for each of these functionalities is described below.

5.3.1 Selecting and Creating a Project

When selecting a project, the user is presented with a list of all his projects from which he can choose one. Through an input field the user can filter this list. Figure 5.10 shows how this can look like. There should also be the option to create a new project, which is why this option is displayed to the user as soon as he enters something in the input field. If a new project is to be created, the user will be prompted to specify the name and the description of the project. After confirming the addition there, he will come back to confirm the selection of the new project.

5.3.2 Selecting a Component

The concept of selecting a component is similar to that of adding a project. Again, the user is presented with a list of available components that he can filter via the input. However, it should not be possible to create components here. The list should be sorted so that the components of the project that interface with the component managed in the current IMS are listed first, followed by the remaining components of the Gropius project. A mockup for selecting a component for the IMS GitHub is shown in Figure 5.11.

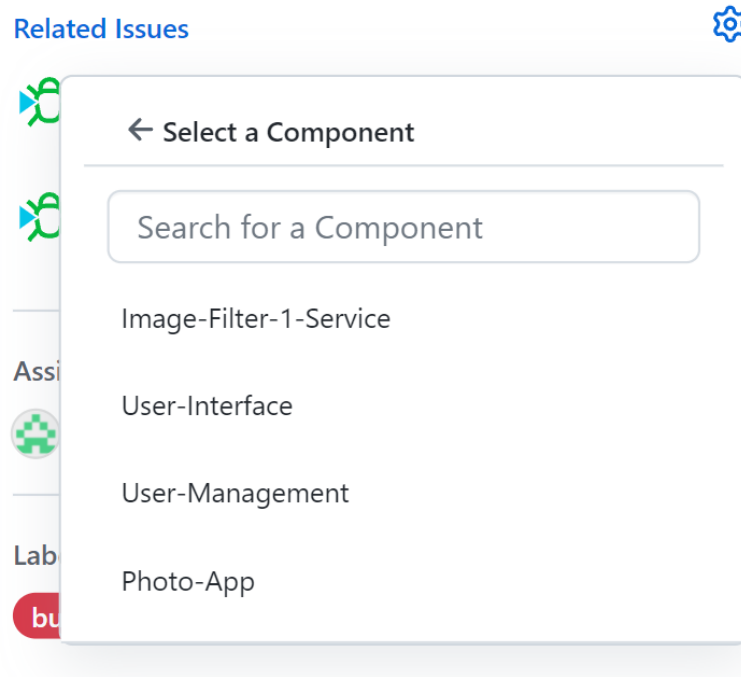


Figure 5.11: Mockup for selecting a component for a Gropius project that was selected before.

5.3.3 Issue Selection and Linking

The principle of selecting an issue is the same as selecting a project and a component. Associated to a given component all issues are listed, which can be filtered by an input field. Once there is user input, there is also the possibility to create a new issue, as is the case when selecting a project described in Section 5.3.1. Figure 5.12 shows a mockup for selecting an issue. When creating an issue, it should be possible to specify title, body, start date, end date, estimated time, as well as the type of issue. After the issue is selected or created, this issue can be linked to the issue whose page in the IMS the user is currently on. For this purpose, source and target issue should be defined, and the type of relation should be specified. After the issue is selected and linked a success message is shown to the user.

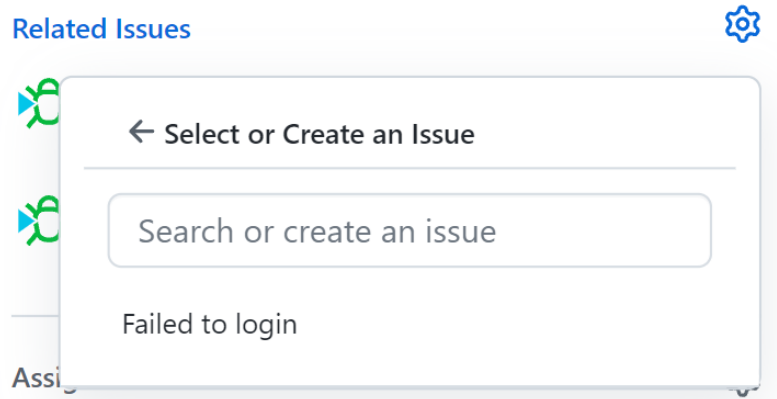


Figure 5.12: Mockup for selecting an issue.

6 Implementation

In this chapter, the architecture and design of the Gropius Browser Extension is discussed in Section 6.1, as well as the implementation of the second prototype for the Gropius Browser Extension in Section 6.2.

6.1 Architecture and Design of the Browser Extension

This section deals with the architecture and the design of the Gropius Browser Extension. The architecture for the Gropius Browser Extension is based on the web extensions framework that is a W3C standard cross-browser architecture [Bro] and is modeled in Figure 6.1. This framework is supported in all popular web browsers like Google Chrome, Firefox, and Opera with the exception of Safari. According to that, the web extension consists of three components which are the background pages, the UI pages, as well as content scripts. The UI pages of the browser extensions are some UI options that can be offered to the user. This includes, for example, a button in the browser toolbar, whereby a popup opens when you click on it. The background pages on the other side are executed as soon as the extension is loaded. „Background scripts are the place to put code that needs to maintain long-term state, or perform long-term operations, independently of the lifetime of any particular web pages or browser windows“ [Moz]. The background and UI pages have access to the browser APIs while the content scripts interact with the webpages. In contrast, background and UI pages can not interact with webpages and content scripts only have limited access to the browser APIs. Thus, the content scripts run in the context of a webpage and the background and UI pages run in the context of the extension. The content scripts can further communicate with the background and UI page via messages [Chrc]. The components listen to other components' messages and can then respond to them via the same channel. Additionally, values can be stored by using the storage API [Chra]. Content scripts, the UI and the background pages also interact with the Gropius back-end via the query language GraphQL. Figure 6.2 shows the messaging between the components as well as the interaction with the Gropius back-end.

6.2 Implementation of the Gropius Browser Extension

In this section the implementation of second prototype of the Gropius Browser Extension is described.

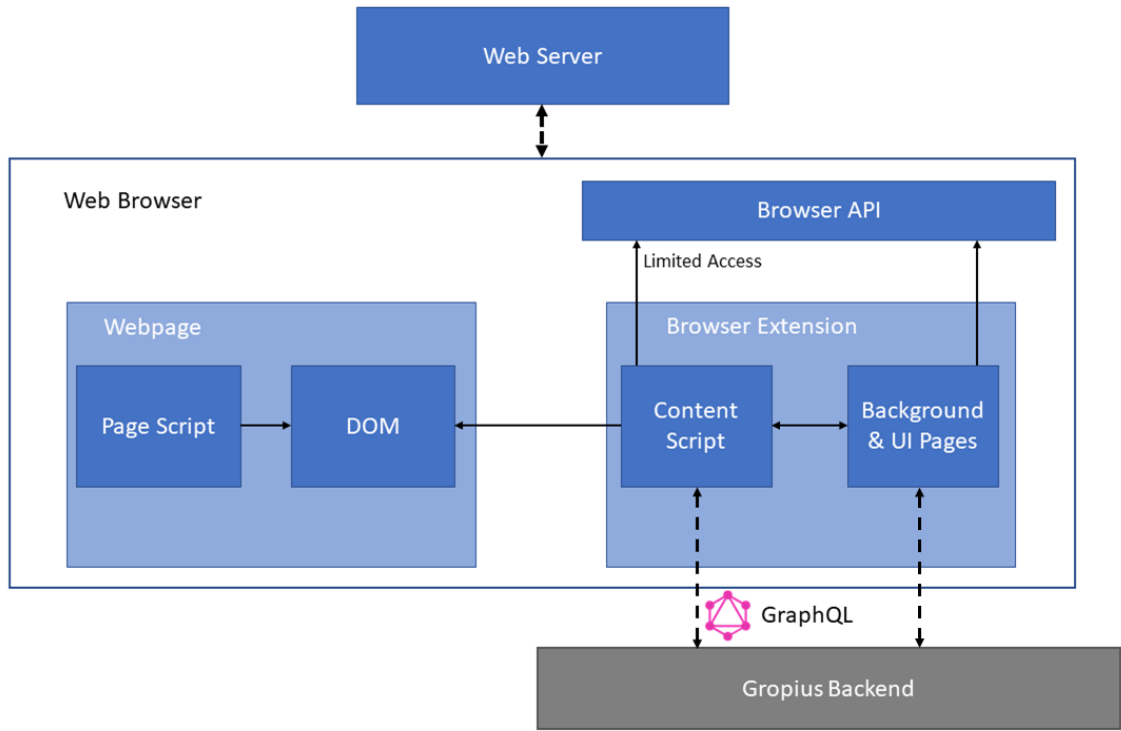


Figure 6.1: Standard cross-browser architecture.

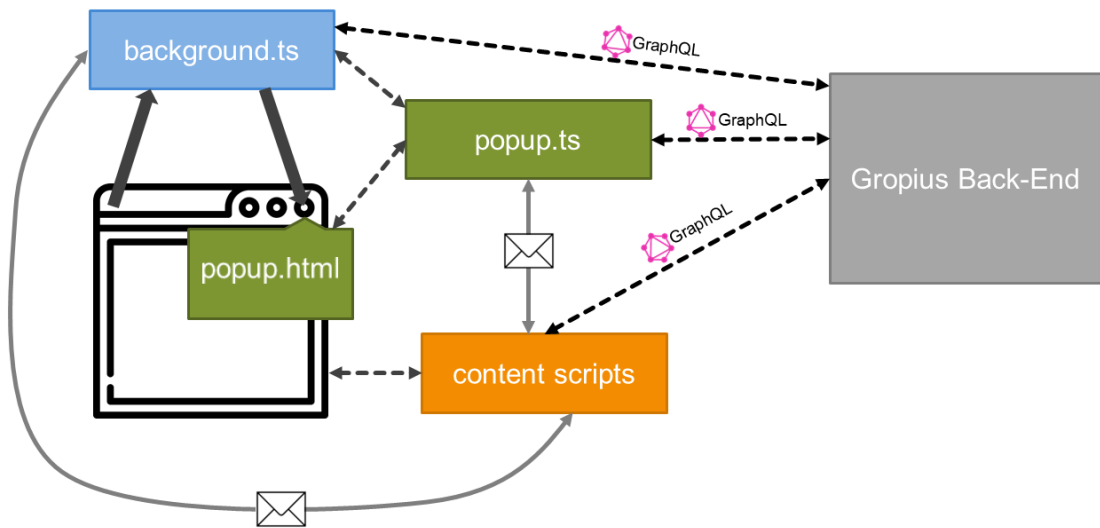


Figure 6.2: Messaging between the components of a browser extension based on [Chra].

First of all, it was decided to use Vue.js¹. Vue.js is a progressive JavaScript framework and with Vue CLI², the standard tooling for Vue.js development, the plugin „vue-cli-plugin-browser-extension“ is used to implement the Gropius Browser Extension. With this plugin it is supported to create the extension for popular browsers like Google Chrome and Firefox. Nevertheless, this prototype for the Gropius Browser Extension is adapted for the Google Chrome browser and should not be used in other browsers for the time being.

In the following, the features implemented in the second prototype are discussed. As described in Section 4.3, a first prototype was implemented to validate the requirements and to design a concept for the functions that the Gropius Browser Extension should cover. This prototype is now to be extended with functionalities. First of all, it is possible, according to the presented concept in Section 5.2, to add a new component to a Gropius project that can be selected or created. Moreover, the list of Gropius projects in which the component is used is also displayed. The corresponding concept is explained in Section 5.2 as well. All further features which were made functional in the second prototype concern the issue management. For an existing issue displayed in the related IMS website, a list of related issues is displayed, which can also be edited. Section 5.2 describes how to proceed when adding an related issue. The concept worked out there is followed here. When the mouse cursor is hovered over a related issue, a small popup with information about the related issue is displayed. The last feature, which is implemented in the second prototype, is the filter for active Gropius projects. According to the active Gropius projects, the related issue list is filtered. Due to time constraints, the planned component diagrams that are described in Chapter 5 have not been implemented. It was discussed with the Gropius front-end developers if there is a way to import the graph component into the Vue.js project without much effort, but no solution was found for this.

The code for the prototype of the Gropius Browser Extension can be found at <https://github.com/ccims/ccims-browser-extension>.

¹<https://vuejs.org/>

²<https://cli.vuejs.org/>

7 Evaluation

This chapter describes the final stage of this thesis, the evaluation of the implemented prototype as well as the concept of the Gropius Browser Extension. For the evaluation of the results, an expert survey was carried out, whereby the questions, which were asked to the experts in the process, were created based on the principle of Goal Question Metric (GQM) introduced by Caldiera et al. For this purpose, some employees from the industry who were interviewed in the requirements engineering phase were asked again whether they would participate in the survey.

7.1 Study Design

In order to evaluate the results of this thesis, first the survey has to be worked out. The rough flow of the expert survey is that first one of the scenarios, which was already introduced in the requirements engineering interview, is simulated using the developed prototype and then questions are asked regarding the prototype, as well as the concept of the Gropius Browser Extension.

Since the same employees from the industry were to be interviewed again for the survey as in the requirements engineering phase, the survey is based on the interview conducted there regarding the requirements for the Gropius Browser Extension. There, the respondents were introduced to the topic of CCIM using two scenarios based on a component-based system of a photo app. This software system is called PhotoPia. The first scenario will now be revisited in the expert survey for the evaluation. For this purpose, only the three components User Management, Photo App and User Interface are considered. A user cannot log in to the app through the User Interface, so as a developer of the Photo App, you look into the User Management's IMS to see if there is a bug that is the trigger for this issue. This bug should then be created and linked for the Photo App component within the UI of the User Management's IMS. The resulting bug in the Photo App component is then to be linked in the UI of the Photo App's IMS to an existing bug in the User Interface component. Since the prototype implemented in this thesis was only implemented for the IMS GitHub, all three components are managed in GitHub. It is then explained to the survey participants that the purpose of this system is actually that the components are managed in different IMS and this scenario is only performed as a demonstration in the IMS GitHub. The first feature then presented to survey participants is adding a component to a Gropius project. For this purpose, the Gropius project PhotoPia was created in advance and the components User Management and User Interface were already added. Together with the participants, the PhotoApp is added to this Gropius project and the list of Gropius projects in which a component is already used is presented. After that, the scenario described above should be performed in IMS GitHub with the implemented prototype. This way, the majority of the implemented features in the prototype are presented to the survey participants, so that it can be evaluated afterwards with appropriate questions.

After the demonstrated scenario is elaborated, the next step is the formulation of appropriate questions, which was carried out based on the GQM principle. Based on the objective that was presented to the respondents in the requirements engineering interviews, the following goal was adopted for as the goal for the evaluation:

(G1) Enabling integrated context aware issue relationship management for independently managed software systems.

Twelve questions were then elaborated, aiming on the one hand to evaluate the implemented prototype, which realizes the formulated goal **(G1)**, but also generally to evaluate the idea of integrated CCIM into the used IMS. The questions are as follows:

- (Q1)** How helpful is the main feature of the Gropius Browser Extension to see related issues of other components?
- (Q2)** How quickly can you recognize that the issue currently viewed has dependencies on issues of other components?
- (Q3)** How big do you estimate the time savings enabled by the Gropius Browser Extension to find related issues of other components (compared to the common approach without the extension)?
- (Q4)** How important do you consider the presented concept of the Gropius Browser Extension for the development process of software and its operation?
- (Q5)** How well do you find the Gropius Browser Extension integrated into the look and feel of the actual website?
- (Q6)** How clear do you think GitHub's interface is with the new features built in?
- (Q7)** How user friendly is editing and adding related issues?
- (Q8)** How helpful is the search function for Gropius projects and components?
- (Q9)** How helpful is the search function for the issues?
- (Q10)** How quickly is it possible to see in which Gropius projects a component is used with the help of the Gropius Browser Extension?
- (Q11)** How high do you rate the chance that the Gropius Browser Extension will be accepted in the industry?
- (Q12)** How important do you consider the concept of cross-component issues for the software engineering process of complex systems that consist of several independent components?

It was decided to use an expert survey as a metric, as this is appropriate in the scope of this thesis. A rating scale was also developed for each question, which was used to evaluate each question by the respondent employees of the companies. As with the creation of the scale for question **(RE4)** in the requirements engineering, the approach here was also based on Porst [Por13]. Thereby again an endpoint-named scale was chosen with five interval points. The endpoints were then named to match the associated question. The rating of -2 for each question corresponds to a completely negative evaluation of the question, whereas the rating of 2 is assigned to a completely positive or

approving evaluation. What exactly the scales for the individual questions look like can be found in the evaluation of the results in Section 7.2. In summary, an expert survey is to be used to validate the objective (G1) of this thesis. For this purpose, questions are posed to software developers, which are to be answered using an endpoint-named scale. Thus, the point of view of the GQM plan is that of a software developer.

7.2 Results

This section presents the results of the expert survey. Eleven of the fifteen employees from the industry interviewed in requirements engineering participated in the expert survey for the evaluation. For each of the questions, a figure was created showing the distribution of the ratings on the one hand, and the average rating on the other, based on the rating scale used. These are first listed one after the other and then a conclusion is drawn about them.

Rating How helpful is the main feature of the Gropius Browser Extension to see related issues of other components?

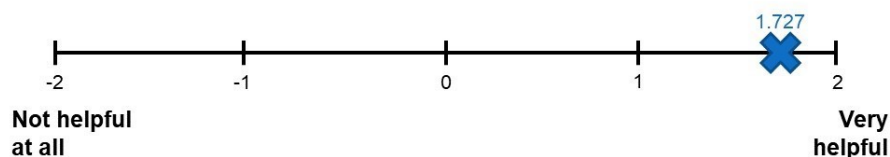


Figure 7.1: Average rating of (Q1).

How quickly can you recognize that the issue currently viewed has dependencies on issues of other components?

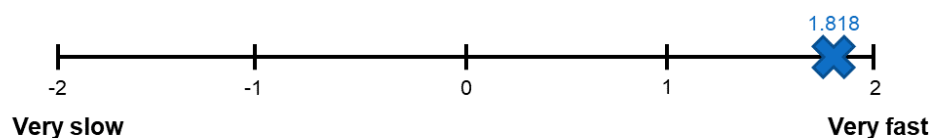


Figure 7.2: Average rating of (Q2).

How big do you estimate the time savings enabled by the Gropius Browser Extension to find related issues of other components (compared to the common approach without the extension)?

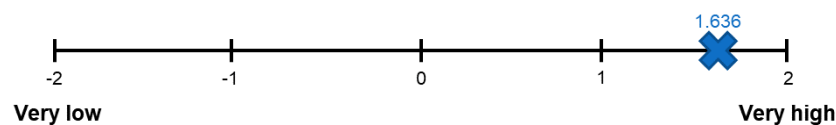


Figure 7.3: Average rating of (Q3).

How important do you consider the presented concept of the Gropius Browser Extension for the development process of software and its operation?

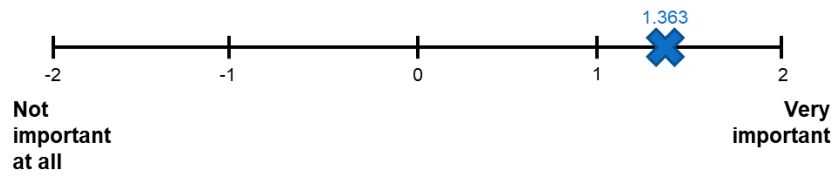


Figure 7.4: Average rating of (Q4).

How well do you find the Gropius Browser Extension integrated into the look and feel of the actual website?

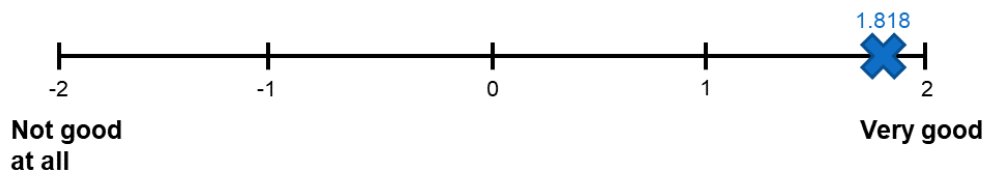


Figure 7.5: Average rating of (Q5).

How clear do you think GitHub's interface is with the new features built in?

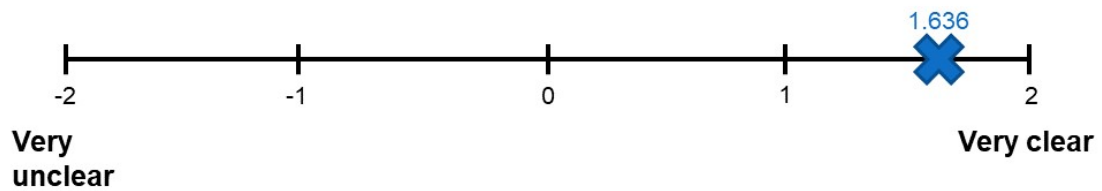


Figure 7.6: Average rating of (Q6).

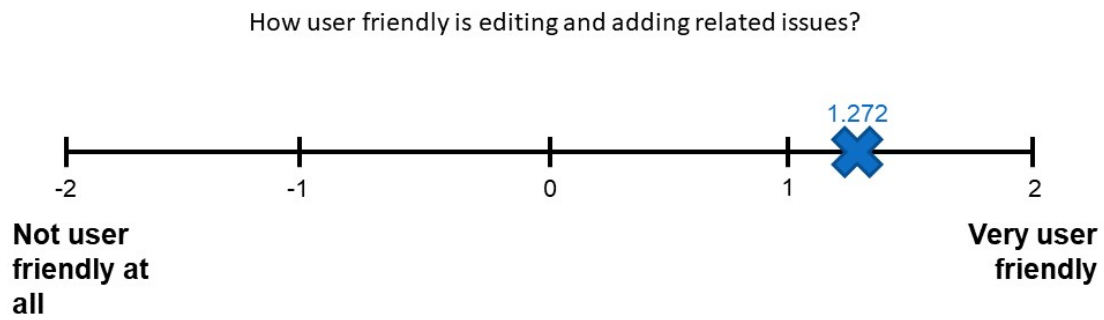


Figure 7.7: Average rating of (Q7).

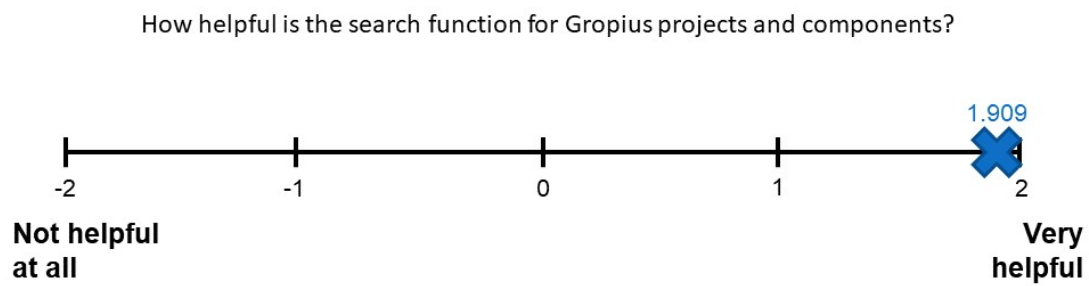


Figure 7.8: Average rating of (Q8).

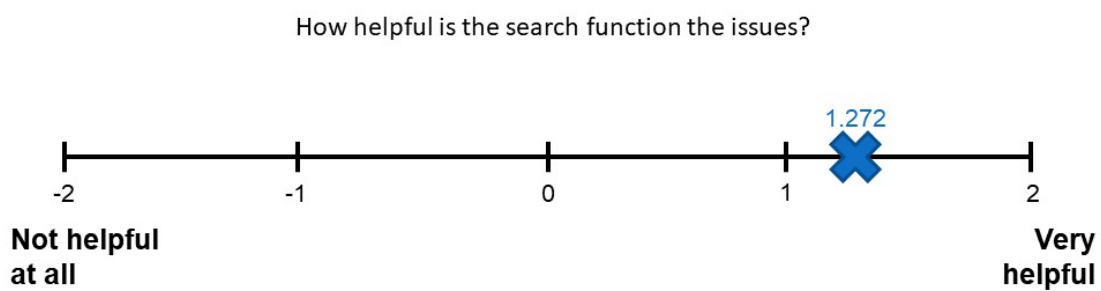


Figure 7.9: Average rating of (Q9).

How quickly is it possible to see in which Gropius projects a component is used with the help of the Gropius Browser Extension?

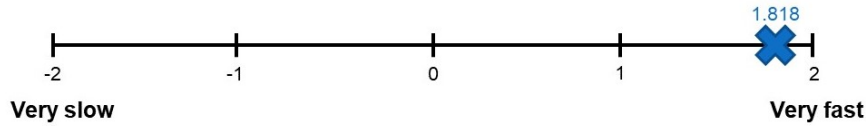


Figure 7.10: Average rating of (Q10).

How high do you rate the chance that the Gropius Browser Extension will be accepted in the industry?

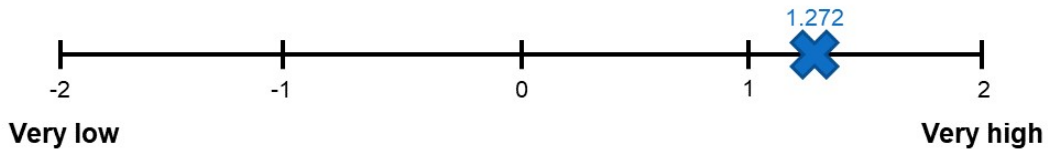


Figure 7.11: Average rating of (Q11).

How important do you consider the concept of cross-component issues for the software engineering process of complex systems that consist of several independent components?

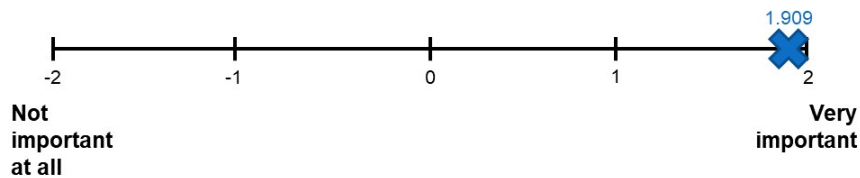


Figure 7.12: Average rating of (Q12).

Basically, the feedback regarding the prototype as well as the concept of the Gropius Browser Extension is very positive. First, the experts' evaluation of the Gropius Browser Extension concept is discussed. Questions (Q1), (Q3), (Q4), as well as (Q12), which are primarily focused on evaluating the concept of the Gropius Browser Extension, received an average score of approximately 1.5. In the expert survey, as in the requirements engineering interviews, it became clear that a problem is seen in the management of cross-component issues in complex software systems. The experts' ratings were between one and two for all four questions. Ratings with one were mainly justified by the fact that there are more important things to do, especially with regard to the entire development process and operation.

In the next step, the evaluation of the questions regarding the implementation in the prototype is presented. Questions (Q2), (Q5), (Q6), (Q7), (Q8), (Q9), as well as (Q10) are primarily related to the evaluation of the prototype. These questions received an average score of approximately

1.6. Basically, the implementation of the prototype was rated really good. In particular, the fluid integration and the adapted design to the UI of GitHub were frequently emphasized here. Some valuable suggestions were also made on how to make the implementation even more user-friendly and useful. First, it became very clear that the search for issues should not be filtered by name alone. Nine of the eleven respondents indicated that searching by issue number should also be possible here. One of them also added that a search based on semantic similarity would also be very helpful here. Furthermore, in terms of usability, the popups for creating issues and projects were criticized for being too small. Here the suggestion was brought that there is an option to have this popup extend across the entire screen by clicking a button. It should then also be possible to provide more detailed information about the issue or project that has been created.

7.3 Discussion

In the following, three hypotheses are first created, which are either accepted or rejected based on the answers to the expert questions. For this purpose, the results of the interviews from the requirements engineering are also included. The following hypotheses were created:

- (H1) In complex software systems consisting of several independent components, the problem of managing cross-component issues is faced by software developers.
- (H2) When such a problem occurs, the Gropius Browser Extension's concept of integrated CCIM provides a solution for that.
- (H3) If the problem of managing cross-component issues between different IMSs arises, it is conceivable that the Gropius Browser Extension will be used in the industry.

Several validation questions are relevant to the first hypothesis (H1). The most important are (RE1) and (RE2) from the interview in requirements engineering. As described in Section 4.2.3, when asked if stakeholders had ever had a problem managing cross-component issues managed in different IMSs, a variety of scenarios were described. In addition, more than half of the respondents reported having such a problem at least monthly. Even those respondents who had never experienced the problem could imagine that such a problem occurs very often, especially in complex systems consisting of several independent components, and that a solution would be desirable. Another question, which serves to validate this hypothesis, is from evaluation and that is (Q12). The question of how important the concept of cross-component issues is rated in the software engineering process of complex systems that consist of several independent components was given an average score of 1.909. This means that the concept is considered very important by the surveyed employees from the industry. Based on these answers to the questions, it can be said very clearly that hypothesis (H1) is true and thus (H1) is accepted.

Following to the discussion regarding the second hypothesis (H2). Several questions can be included in this regard as well. Starting with the questions from requirements engineering, where on the one hand the sub-goals worked out in Section 4.2.1 should be evaluated. In Section 4.2.3, the results are presented in this regard, where it can be seen that all three goals were rated on average between 1 and 2, meaning that the interviewed employees from the industry rate all three goals as important to even very important. The core of the presented concept of the Gropius Browser Extension is the integration of the features covered by Gropius into the IMS used by the software developers.

Accordingly, the question (**RE5**) from the requirements engineering is another question which is relevant for the validation of (**H2**). The majority of the industrial employees interviewed prefer integration within the IMS they use to the introduction of a separate UI for Gropius, as outlined in more detail in Section 4.2.3. Finally, the relevant questions from the evaluation's expert survey are considered in relation to (**H2**). As summarized in Section 7.2, the evaluations by the experts regarding the concept of the Gropius Browser Extension are consistently positive. There were no negative outliers to any of the questions asked in this regard. When problems arise with CCIM, they can of course only be solved to a certain extent. No extension can solve the problem in its core, nevertheless it becomes clear that the presented concept of the Gropius Browser Extension makes the management of these cross-component issues much easier and thus (**H2**) is accepted.

A separate question was asked about the last hypothesis (**H3**) in the evaluation's expert survey. The average rating of how likely it is that the Gropius Browser Extension will be adopted in the industry is 1.272, as depicted in Figure 7.11. The majority of respondents rated this question at 1, stating that it is always difficult to introduce a new system in the industry. Many companies would have the attitude of „never change a running system“ which would lead to a limited openness towards such new systems. Nevertheless, the barriers to using the Gropius Browser Extension were considered quite low, which is why the respondents thought it was likely that it would be accepted by the industry. In any case, the barriers of integrating a browser extension into the IMS used is significantly lower than introducing a completely new tool. Thus, chances are high that the Gropius Browser Extension will make it easier to start using Gropius in the industry. This leads to the third and last hypothesis (**H3**) being accepted as well.

7.4 Threats to Validity

When discussing the threats to validity, four aspects can be distinguished according to Runeson et al. [RH09]. These are construct, internal, and external validity, as well as reliability.

Commencing with the internal validity. Since the evaluation was conducted as an expert survey, it is subjective. The answers depend on personal opinions, but also possibly on external influences on the respondents.

Furthermore, there are also threats to external validity. Again, this is due to the fact that the evaluation was conducted as an expert survey. With surveys, there is always the question of how representative the group surveyed actually is. Accordingly, some findings from the survey may not be generalizable. In the scope of this thesis, care was taken to interview different employees from industry whose work experience varies and who are employed in companies of different sizes. Nevertheless, further larger studies should be conducted to allow for a more statistically significant validation.

In addition to internal and external threats to validity, there are also threats to construct validity. On the one hand, it is possible that questions were misunderstood by the surveyed employees from the industry. However, because the surveys were conducted in person, it was possible to directly answer the questions of the respondents and directly address any misunderstandings. Thus, this threat is minimal. Another threat to construct validity is the fact that the prototype was implemented only for the IMS GitHub. But the core of the concept of the Gropius Browser Extension is to enable the

management of related issues that are managed in different IMS. During the survey, it was therefore made clear right at the beginning that the demonstration of the scenario would only be carried out as an example using the IMS GitHub.

The last aspect regarding threats to validity is regarding reliability. This is about the extent to which the results of the analysis depend on the researcher. In order to keep this threat as small as possible, the questions as well as the procedure of the survey were first discussed with the supervisors of this thesis and thus their comprehensibility was ensured as much as possible. In the same way, the documentation and discussion of the results of the evaluation was clarified.

8 Conclusion

This chapter aims to conclude this thesis which includes on the one hand a short summary in Section 8.1. On the other hand, in Section 8.2 it is outlined who benefits from this thesis and in which way. Subsequently, the limitations are presented in Section 8.3, the main lessons learned in Section 8.4, and finally future work based on this in Section 8.5.

8.1 Summary

The main objective of this thesis was the proof-of-concept of the Gropius Browser Extension. For this purpose, detailed requirements were first collected, some of which were directly implemented in a first prototype. A lot of time was spent gathering requirements for the Gropius Browser Extension from a wide variety of stakeholders which are on the one hand Gropius experts and on the other hand employees of very large, medium-sized, as well as small companies. These requirements were gradually implemented in a first prototype, which has already been used in some interviews to validate collected requirements. Based on the gathered requirements, as well as the first prototype, a detailed concept was then created. In addition to that, a second prototype has been implemented, which functionally implements some selected requirements. This prototype, as well as the created concept of the Gropius Browser Extension, then had to be evaluated. In the expert survey conducted during the evaluation, it became clear that the concept is generally rated very positively. In summary, the integration of CCIM for independently managed components into the IMS used by software developers is considered very useful by the interviewed employees from the industry. Especially when it comes to the Gropius system being accepted by the industry, it is necessary to give it an easy start, especially at the beginning. The Gropius Browser Extension is very well suited for this, as the barriers here are lower than introducing a completely new tool. The developers are familiar with their currently used IMS and most of them do not like to change a running system.

8.2 Benefits

The final product addresses especially developers as well as product owners who work with complex software systems consisting of several independent components. Through the interviews it became very clear that a solution for the problem regarding the management of cross-component issues is urgently desired by the target group. During the interviews in requirements engineering, as well as in the ones of the evaluation, the feedback regarding the concept of the Gropius Browser Extension was clearly positive. The target group benefits from the Gropius Browser Extension mainly in terms of time savings, but also from having a larger overview of complex software systems.

But also the Gropius experts benefit from the result of this bachelor's thesis. The interviews made it possible to gather further requirements that are generally relevant to the Gropius system. In addition, the interviews showed that the general idea behind Gropius is rated as very useful by most of interviewed employees from industry.

8.3 Limitations

As also mentioned in Section 7.4, a limiting factor in gathering requirements, as well as evaluating the concept and prototypes of the Gropius Browser Extension, is that it is possible that the surveyed group is not representative enough.

In addition, the evaluation of the concept, as well as the prototype is limited to the IMS GitHub. Though it was made clear that such a Gropius Browser Extension should be implemented for other common IMSs, nevertheless, some questions of the evaluation have now been answered only with regard to the implementation for GitHub.

8.4 Lessons Learned

Since a lot of time was spent on requirements engineering, this is also where I learned the most. I have never done requirements engineering on this scale before. As with many things, gaining experience is of great importance here. Through this thesis, I have learned in particular what to pay special attention to in requirements engineering and how best to proceed in planning it. I also gained a lot of experience by conducting the numerous interviews with employees from the various companies. Through the different points of view, I was able to look at the topic from different perspectives, which would not have been possible without the interviews.

Furthermore, I learned what really matters in scientific work. On the one hand, I would not have thought at the beginning that the conceptual part would take up such a large part of this thesis, compared to the actual implementation. But also in terms of evaluation, I learned what to look for and how best to proceed with the evaluation. Last but not least, of course, I learned a lot about scientific writing itself. Learning to research foundations and related work, and how to properly include sources, those are just other things I have learned.

8.5 Future Work

In this section, the future work based on this thesis is suggested.

First of all, more of the collected requirements should be implemented in the Gropius Browser Extension. First and foremost, this includes the section of the component diagram, which should show the related issues with the associated components for an issue. In addition, a component diagram for displaying the entire Gropius project with the associated components and their dependencies should be implemented. However, the Gropius Browser Extension should also be enhanced with some of the other requirements collected, in particular, when they are then fully supported by the Gropius back-end.

Furthermore, the Gropius Browser Extension must also be implemented for other IMSs. The concept of the individual features has already been formulated in general terms within the scope of this thesis. Nevertheless, specially adapted concepts must be designed for the individual IMSs which also offer a fluid integration of the Gropius functionalities here.

Bibliography


- [Aie18] M. Aiello. “The Web Was Done by Amateurs”. In: *The Web Was Done by Amateurs: A Reflection on One of the Largest Collective Systems Ever Engineered*. Cham: Springer International Publishing, 2018, pp. 1–6. ISBN: 978-3-319-90008-7. DOI: 10.1007/978-3-319-90008-7_1. URL: https://doi.org/10.1007/978-3-319-90008-7_1 (cit. on p. 5).
- [Atl] *What is an issue?: Jira software cloud*. 2020. URL: <https://support.atlassian.com/jira-software-cloud/docs/what-is-an-issue/> (cit. on p. 3).
- [BEZ08] A. Blair-Early, M. Zender. “User interface design principles for interaction design”. In: *Design Issues* 24.3 (2008), pp. 85–107 (cit. on p. 5).
- [Bro] URL: <https://browserext.github.io/browserext/#availability-csp-content> (cit. on p. 39).
- [BVGW10] D. Bertram, A. Voids, S. Greenberg, R. Walker. “Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams”. In: *Proceedings of the 2010 ACM conference on Computer supported cooperative work*. 2010, pp. 291–300 (cit. on p. 3).
- [Chra] *Architecture overview*. URL: <https://developer.chrome.com/docs/extensions/mv3/architecture-overview/#apis> (cit. on pp. 39, 40).
- [Chrb] *Getting started*. URL: <https://developer.chrome.com/docs/extensions/mv3/getstarted/> (cit. on p. 5).
- [Chrc] *Message passing*. URL: <https://developer.chrome.com/docs/extensions/mv3/messaging/> (cit. on p. 39).
- [DG09] M. Dhawan, V. Ganapathy. “Analyzing information flow in JavaScript-based browser extensions”. In: *2009 Annual Computer Security Applications Conference*. IEEE. 2009, pp. 382–391 (cit. on p. 5).
- [Gita] *About issues*. URL: <https://docs.github.com/en/issues/tracking-your-work-with-issues/creating-issues/about-issues> (cit. on p. 3).
- [Gitb] *Build software better, together*. URL: <https://github.com/collections/github-browser-extensions> (cit. on p. 7).
- [JD03] J. N. Johnson, P. F. Dubois. “Issue tracking”. In: *Computing in Science & Engineering* 5.6 (2003), pp. 71–77 (cit. on p. 4).

- [LLm+21] W. b. E. LePage, E. LePage, W. by Evan LePage Evan is Unito's Senior Content Manager. A journalist-turned marketer, E. L. E. is Unito's Senior Content Manager. A journalist-turned marketer, E. is Unito's Senior Content Manager. A journalist-turned marketer, A. a. b. E. LePage, B. G. Michaels, B. N. Bouchard, B. M. Wurm. *10 of the Best Jira Chrome Extensions (and They're All Free!)* 2021. URL: <https://unito.io/blog/best-jira-extensions-chrome/#h-workflow-extensions-for-jira> (cit. on p. 8).
- [Meh16] P. Mehta. "Introduction to Google Chrome Extensions". In: *Creating Google Chrome Extensions*. Berkeley, CA: Apress, 2016, pp. 1–33. ISBN: 978-1-4842-1775-7. DOI: [10.1007/978-1-4842-1775-7_1](https://doi.org/10.1007/978-1-4842-1775-7_1). URL: https://doi.org/10.1007/978-1-4842-1775-7_1 (cit. on p. 5).
- [MNH15] S. Mahmood, M. Niazi, A. Hussain. "Identifying the challenges for managing component-based development in global software development: Preliminary results". In: *2015 Science and Information Conference (SAI)*. IEEE, 2015, pp. 933–938 (cit. on p. 1).
- [Moz] *background - Mozilla: MDN*. URL: <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/background> (cit. on pp. 5, 39).
- [Por13] R. Porst. *Fragebogen: Ein Arbeitsbuch*. Springer, 2013 (cit. on pp. 17, 44).
- [RH09] P. Runeson, M. Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empir. Softw. Eng.* 14.2 (2009), pp. 131–164 (cit. on p. 50).
- [RSH09] C. Rupp, M. Simon, F. Hocker. "Requirements engineering und management". In: *HMD Praxis der Wirtschaftsinformatik* 46.3 (2009), pp. 94–103 (cit. on pp. 11, 13, 15, 24).
- [SBB20] S. Speth, U. Breitenbücher, S. Becker. "Gropius—A Tool for Managing Cross-component Issues". In: *European Conference on Software Architecture*. Springer, 2020, pp. 82–94 (cit. on pp. 1, 4, 6).
- [SBB21] S. Speth, S. Becker, U. Breitenbücher. "Cross-Component Issue Metamodel and Modelling Language". In: *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER 2021)*. SciTePress, May 2021, pp. 304–311. DOI: [10.5220/0010497703040311](https://doi.org/10.5220/0010497703040311) (cit. on pp. 4, 6).
- [SKBB21] S. Speth, N. Krieger, U. Breitenbücher, S. Becker. "Gropius-VSC: IDE Support for Cross-Component Issue Management". In: *Companion Proceedings of the 15th European Conference on Software Architecture (ECSA-C 2021)*. CEUR, Oct. 2021. URL: <http://ceur-ws.org/Vol-2978/tool-paper103.pdf> (cit. on pp. 4, 6).
- [SP10] B. Shneiderman, C. Plaisant. *Designing the user interface: Strategies for effective human-computer interaction*. Pearson Education India, 2010 (cit. on pp. 5, 30, 34).
- [Zdr00] J. F. Zdralk. *Clarifying the fidelity dimensions of prototypes*. 2000 (cit. on p. 26).
- [Zena] *Differences between GitHub and ZenHub*. Accessed: 2021-10-27. ZenHub, 2021. URL: <https://zenhub.medium.com/zenhub-vs-github-projects-what-are-the-key-differences-eeafbf477183> (cit. on p. 7).
- [Zenb] *ZenHub - Extension for GitHub*. Accessed: 2021-10-27. ZenHub, 2021. URL: <https://www.zenhub.com/extension> (cit. on p. 7).

All links were last followed on November 03, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Reichenbach, 03.11.21,  _____
place, date, signature