Bachelor Thesis

June 27, 2016

Fostering Awareness within Development Teams

Eros Fricker

of Schinznach-Bad, Switzerland (11-920-337)

supervised by

Prof. Dr. Thomas Fritz

Katja Kevic

University of Zurich[UZH]

s.e.a.l.

software evolution & architecture lab

Bachelor Thesis

Fostering Awareness within Development Teams

Eros Fricker

University of Zurich[UZH]

s.e.a.l.
software evolution & architecture lab

Bachelor Thesis

Author: Eros Fricker, eros.fricker@uzh.ch

URL: http://www.erosfricker.ch/bachelor-thesis

Project period: 01.03.2016 - 01.07.2016

Software Evolution & Architecture Lab
Department of Informatics, University of Zurich

Acknowledgements

I want to thank everyone involved in this bachelor thesis. Especially, I would like to thank professor Dr. Thomas Fritz for providing me the opportunity to write this thesis. Also, I want to thank Katja Kevic for the great support and supervision while writing my thesis. Finally, I want to thank everyone who has participated in the evaluation of my thesis.

Abstract

Awareness in development teams has been thoroughly investigated over the past few years. Empirical studies have shown that one of the most sought informations in development team awareness is the information about the ongoing work items of team colleagues [KDV07]. Most of the related research has investigated awareness in situations where developers had access to a computer.

This thesis approaches the field of awareness in a different way by providing a prototype implementation of an application for the Recon JET augmented reality glasses, designed to support spontaneous awareness within development teams in situations without access to a computer. Furthermore, this thesis presents the results of interviews conducted with experienced developers based on the aforementioned application.

In the evaluation of this thesis, we found that our prototype application is able to support certain situations to foster awareness within development teams, especially in stand-up meetings. Also, we found that the functionality which would be used the most by the participants on a daily basis is finding meeting times with other developers.

Zusammenfassung

Das Bewusstsein in Software Entwicklungs-Teams wurde bereits in einer Vielzahl von Studien untersucht. Empirische Studien zeigen auf, dass die meistbenötigte Information im Bereich des Bewusstseins von Entwicklungs-Teams die Information über die aktuelle Arbeit der Teamkollegen ist [KDV07]. Ein Grossteil der Forschungsarbeit hat sich jedoch auf die Unterstützung des Bewusstseins in Situationen, in denen der Zugriff auf einen Computer möglich ist, beschränkt.

Diese Bachelorarbeit untersucht das Thema Bewusstsein in Entwickler-Teams auf eine neue Weise, indem eine Prototyp-Applikation für die Recon JET Brille implementiert wurde, welche ein spontaneres Bewusstsein in Situationen ohne Zugriff auf einen Computer unterstützen soll. Des Weiteren werden die Resultate der Interviews, welche mit erfahrenen Entwicklern durchgeführt wurden und auf der oben genannten Applikation basieren, präsentiert.

In der Evaluation unserer Arbeit fanden wir heraus, dass die Prototyp-Applikation gewisse Situationen unterstützen kann, um das Bewusstsein über die Arbeit der Teammitglieder eines Software-Entwicklers zu verbessern, vor allem in Stand-up Meetings. Ausserdem fanden wir heraus, dass die Teilnehmer das Finden von Meeting-Zeiten mit anderen Teammitgliedern in ihrer täglichen Arbeit am ehesten benutzen würden.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

The awareness in software development teams has been thoroughly investigated. Empirical studies show that in order to improve the success of development efforts, the awareness of ongoing activities and the project status need to be improved [BNPL09, GPS04]. Ko et al. found that one of the most sought informations of developers is the information about the colleagues' ongoing work items [KDV07].

Many contributions on the topic of awareness in development teams focus on collaboration tools. Issue tracking systems, such as `Bugzilla` [bug16], are among the most commonly used tools to foster collaboration. Several researchers investigated the usage and the flaws of these issue trackers in qualitative studies and suggested improvements [IS15, BHG13b]. It was shown that issue tracking systems have many positive effects on the collaboration and awareness in development teams, even if the team members are geographically located in different countries [VCTS15]. Another important medium of developer awareness are dashboards. The usage of dashboards and their effect on the collaboration, i.e. how well they can support development team collaboration and awareness, has been explored in numerous studies [BHG13a].

Several new tools to improve the awareness in development teams have been proposed. They support developers in maintaining awareness by either enhancing IDEs, improving the awareness capabilities of these development environments by showing awareness-specific information [GBMN12, HCRP04, HD08], or by implementing additional stand-alone applications to support developers in maintaining awareness of their colleagues' ongoing work [BD09, GSPP05, BCSR07, Sch01, KBHG14, SNvdH03].

Augmented reality offers completely new ways of displaying contextual information at any time. It was shown that by supporting the user with an augmented reality application, the user's awareness can be improved in situations such as the attendance of a conference [DSAF99] or giving a presentation [HP14].

Contrary to the aforementioned work, which is focused on improving and maintaining awareness in situations with access to a computer, this thesis explores the possibilities to potentially improve awareness within development teams, when the access to a computer is hindered or completely impossible. We present an application prototype for the Recon JET augmented reality glasses, based on the Android operating system, which fosters spontaneous awareness by allowing the user to scan developers in situations like meetings or hallway encounters. The application is able to unobtrusively display information about the scanned developer's current issue tracking items as well as his upcoming calendar events, i.e. the events of the next seven days.
We used our prototypical application to guide a semi-structured interview with experienced de-

velopers of two different companies. Specifically, we aim to investigate the following research questions:

(i) In what situations can developers profit from displaying awareness information through smart glasses?

(ii) What information artifacts would developers seek on smart glasses?

We found that developers might profit from displaying awareness information in stand-up meetings and in situations where they are on-the-go and cannot operate a device without interfering with the awareness of the environment. The evaluation participants mentioned that they would be specifically interested to learn about the status of open issues as well as their interconnection, i.e. which issue may block another important issue. Also, they would like to see build, test and system statuses as well as the availability status of other developers in order to be able to discuss issues without disturbing them at work. The participants generally assessed our prototypical application to be user-friendly. Especially, the scanning of other developers was rated fast enough to be used in their daily work. While the participants could not imagine replacing their information media with the glasses, they would happily dispose of their smart phones in the situation where they want to find a possible meeting time with a team member. This is mainly due to the fact that there is no way to automatically find the possible meeting time with their smart phone and that the glasses can display information more unobtrusively.

1.1 Structure

The related work is discussed in chapter 2 which is categorized into sections by the overall topic of the contributions. In chapter 3 we introduce the reader to our approach by presenting an overview of the application as well as the evaluation method. In chapter 4, the implementation is presented in more detail, discussing the conceptual and technical peculiarities of the application. Chapter 5 is dedicated to the exploratory study conducted as well as to the results of the interviews and the conclusions which can be made, based on them. To conclude, we will propose ideas to further enhance the application in chapter 6 and provide a final conclusion of this thesis in chapter 7.

Chapter 2

Related Work

Over the past years, the awareness in software development has been the subject of intensive research. Work related to this thesis can roughly be categorized into empirical studies which investigated the awareness of developers in collocated and distributed teams, tools implemented to improve awareness, and the use of augmented reality devices to improve awareness.

2.1 Empirical Studies

Developer awareness is a crucial issue in the field of software development. Many researchers found that in order to improve developers' day-to-day work and ultimately the success of the development efforts, the awareness of ongoing activities in their development teams and of the overall project status needs to be improved. Baysal et al. [BHG13a] have investigated on the current usage of developer dashboards to support team awareness. They have shown in a qualitative study that current – mainly quantitative – dashboards poorly support the developers' daily information needs and have further researched on the need of qualitative dashboards to support their work. They found that qualitative, task-specific dashboards may improve the ability to maintain awareness of the evolution of the issues, in which the developer is interested, and that future dashboards need to integrate more qualitative approaches to be of use for the developers' daily work.
Ko et al. [KDV07] investigated the specific information needs, software developers encounter in their daily work. By observing developers, transcribing their activities and analyzing these logs, they categorized the information needs and contributed a catalogue thereof based on these observations. The information most frequently needed was their coworkers' activities and information on source code artifacts.
Begel et al. [BNPL09], however, studied the work coordination within and between different development teams on a larger scale, surveying international software development teams. The survey examined how the teams collaborated and how successful the team members felt in accomplishing their collaborative implementation tasks. They found that the most important information needed for collaboration was not source code, but their colleagues' schedules and features which were to be implemented. Another finding of their research was that in order to successfully collaborate in large-scale development teams, a personal relationship between coworkers should be created and maintained.
In a similar way, Gutwin et al. [GPS04] researched on the collaboration of open-source software development teams. They have interviewed teams from multiple open-source software development projects in order to find out how these loosely coupled teams manage to create and maintain group awareness. They found that these teams were not only able to maintain group awareness in their projects but also that they did so by using simple mechanisms such as mailing lists and

chats. They continued by analyzing some of the problematic effects these communication mechanisms create and provided further suggestions on improving group awareness in open-source software development teams.
Fritz and Murphy [FM10] contributed to the research on group awareness by having interviewed software developers on the specific questions that arise in their work and having created a catalogue of these questions. Their research contributes an information fragment model which can simplify – compared to previous approaches – the answering of these questions.
Storey et al. [SvG05] provided a framework for the description, comparison and understanding of visualization tools which were proposed for creating and maintaining awareness of software developers' activities. They further present and classify these visualization tools and conclude their work by providing a useful table which compares the tools based on their proposed framework. This contribution may be used by the interested reader to gain an overview of the existing tools that have been proposed to improve awareness. The empirical studies mentioned in this section were used as a reference for the implementation of `AwareGlass`. These studies supported us in identifying the most important informations to display in our application.

2.2 Collaboration Tools

The research and development of collaboration tools is strongly related to the research on general awareness. Newly developed collaboration tools try to augment the awareness of development teams by creating new ways to discover the existing data. One of the most prominent tools to increase awareness within development teams are bug tracking applications. Iram and Saeed [IS15] present a case study with the goal to improve the user experience of bug tracking applications towards a more user-centered approach. They have interviewed software developers in Pakistan in order to collect the issues when using bug trackers. Based on the results of their interviews, they implemented a prototype of a more user-centered bug tracking application and evaluated their prototype with the same developers that were interviewed. They managed to improve the user experience of a bug tracking application by changing the standard implementations of such applications, e.g. by using color schemes or implementing a drag and drop control scheme in their prototype.
Going in the same direction, Baysal et al. [BHG13b] conducted a qualitative study on the experiences, developers had with the `Bugzilla` [bug16] bug tracking system. Further, they implemented a prototype for maintaining a personal watch list in order to improve the awareness of the user-specified issues. One of the major findings of their research was that developers had a strong desire for the improvement of situational awareness in bug tracking systems. Also, they found that most of the interviewed developers had problems maintaining a global picture of their application's issues.
By conducting semi-structured interviews and on-site observations, Vyas et al. [VCTS15] researched on the role of bug tracking applications to support developers' activities in a software development team located in Finland and India. They found that bug tracking applications supported the information exchange and communication between developers and that they build mutual understanding and a common knowledge base for all the stakeholders involved, even if they are located in different locations.
`Codebook`, proposed by Begel and De-Line [BD09], tries to incorporate the mechanisms of a social network in order to not only connect software developers with each other but also to include the code artifacts in the social network and connect with them. It helps the software developers to improve awareness of the current statuses of the followed issues as well as the current activities of the followed developers.
Gutwin et al. [GSPP05] proposed `ProjectWatcher`, a system that presents the user a visualiza-

tion of which developers are active in the project, who has changed which code artifacts, as well as what part of the project the corresponding developer has been working on. The system is able to generate these visualizations by mining all user edits of all the artifacts of an application and combining these informations. ProjectWatcher can be used as a standalone tool or as a plug-in for the Eclipse IDE [ecl16].
FASTDash, proposed by Biehl et al. [BCSR07], is another visualization tool focused on improving the awareness of software development team activities. It visualizes the source files which are checked out, the files being viewed and which exact classes / methods are being changed at the moment. The visualization can even be further annotated with more specific status information to enhance the expressiveness of the dashboard. Like the other visualization tools, FASTDash has been implemented after a series of surveys, observations and interviews with the developers.
Schummer [Sch01] has proposed TUKAN, a visualization tool that – like the aforementioned FAST-Dash – is based on a spatial representation of the source code which is under development. It is different to the others in that it dynamically adapts to the user's preferences and tries to improve itself based on them.
Kononenko et al. [KBHG14] propose another dashboard, called DASH, to visualize the information obtained from a Bugzilla [bug16] issue tracker. It incorporates personalized views of the issues selected by the user and informs the user about the current progress and the changes made. It focuses on simplicity, speed and usability and was developed in order to overcome the limitations of Bugzilla issue trackers.
Palantir is a workspace awareness tool proposed by Sarma et al. [SNvdH03]. It informs the user about the current changes made by other developers (i.e. who is changing which artifact), calculates the degree of severity of the change and displays the information in a configurable view. By informing the user about ongoing changes and providing a holistic view of all the workspaces, the tool improves the awareness among developers and enables the coordination of their activities, especially with regard to the future integration of the changes.

While the approaches discussed above focus on the implementation of stand-alone collaboration tools, the following are embedded into IDEs. One of the most prominent approaches is CARES by Guzzi et al. [GBMN12]. It is a communication tool which improves the awareness of a developer in that it is embedded directly into the IDE workspace. It displays photos and information of the software developers which have worked on the currently open file and may be used as a communication tool. With CARES, Guzzi et al. [GBMN12] have proposed a solution to the problem of uniquely identifying contributors of a code artifact, even if the contributor has left the company or changed their e-mail address. CARES is focused on simplifying the communication between developers and tries to create and maintain a sense of community in a development team.
Hegde and Dewan [HD08] proposed another IDE-embedded solution for improving developer awareness with a tool called CollabVS. CollabVS is a plug-in for the Microsoft Visual Studio [vis16] IDE and it is similar to the aforementioned FASTDash in that it provides the user with a detailed view of who is changing which file. It even goes further in that it implements an IDE-embedded text-, voice- and video-chat and it enables its users to receive notifications as soon as an artifact of interest is no longer "locked" by another developer.
Jazz, an awareness-centered plug-in for the Eclipse [ecl16] IDE has been presented by Hupfer et al. [HCRP04]. Jazz is intended to be used in small software development teams which are ideally sitting close to each other. It displays photos of the development team members and also includes status icons. Furthermore, Jazz enhances the IDE's package explorer by adding color-coded icons, displaying the current status of the artifact (e.g. if it is currently being changed or if it is free to be changed, i.e. not opened by any other team member). To further enhance contextual awareness, Jazz also includes a text-based chat, which has several features. The chat enables the inclusion of source code with proper formatting and it includes hyperlinking and

auto-completion to improve the inter-team communication.
As we have seen in this subsection, the implementation of awareness-centered communication and collaboration tools is of high interest in current research and while there are many similarities between the different approaches, every proposal focuses on other aspects of the awareness among software developers. We have used the aforementioned related work in order to find out about the existing implementations of awareness-centered tools. Especially, we have analyzed the existing functionalities of these tools and incorporated implementations of the most important features thereof.

2.3 Augmented Reality

A proposal which is most related to our research, is presented by Dey et al. [DSAF99]. Dey and colleagues proposed an augmented reality application called `Conference Assistant` which is similar to the one presented in this thesis. It is a context-aware mobile application, focused on assisting conference attendees as well as presenters. The `Conference Assistant` helps users to decide which presentations to attend, assists them in taking notes during presentations as well as allows the users to retrieve conference informations after the conference has concluded. The augmented reality application uses an array of contextual information such as location, time, etc. to improve the user's awareness on the go.
Hernandez and Picard [HP14] implemented an application for the Google Glass, `SenseGlass`, which enables its users to measure emotions in natural settings. They combined the sensors of the augmented reality glasses with a smile recognition software such that its user may improve awareness of the overall mood in a natural setting, e.g. during a presentation the user may quickly change his presentation style when the mood drops.
In their research, Lukosch et al. [LBAK15] have compiled different research articles focused on the field of augmented reality. They discuss the progress that has been made with these articles and finally provide a research agenda, identifying the work that still needs to be done.
The research in augmented reality aided us to understand, what augmented reality applications should be capable of and how we can use contextual awareness informations to improve our application.

Chapter 3

An Approach to Foster Awareness in Collocated Development Teams

This chapter provides an overview of the approach to foster awareness within development teams. In section 3.1 we present the `AwareGlass` application, the augmented reality application implemented over the course of this thesis. We will shortly discuss the functionality in order to provide an overview of the application in sections 3.2 and 3.3. Section 3.4 presents the evaluation method chosen in our thesis.

3.1 The AwareGlass Application

`AwareGlass` serves as a prototype application for fostering awareness in development teams. Since it is implemented for augmented reality glasses, it enables the user to display information about his own and his colleagues' ongoing work unobtrusively. The goal of this implementation is to find out about the possibilities of an augmented reality application to improve and maintain developer awareness on the one hand and on the other hand to create a starting point for the walkthrough in the evaluation interviews conducted with developers.

3.2 Features

`AwareGlass` incorporates several functionalities which support the user in maintaining and improving awareness. In order to foster self-awareness, i.e. the awareness of the user's own ongoing work, the following functionalities were implemented:

(i) **Accessing issues:** The application is able to display the user's issues from the `Bugzilla` issue tracking system. The user can therefore see his own ongoing work items in situations where he does not have access to a computer. The details of an issue, e.g. its status, its deadline date, its creation date, etc., can be accessed by selecting one of the issues in the list. Further, the issue list can be sorted, based on selected criteria, such that the user is able to choose what his preferred priorities are. Ko et al. [KDV07] investigated the habits of software development teams and found that one of the most important informations for a developer to foster awareness is the information about colleagues' ongoing work items, therefore we implemented the access to the specific issues of a developer.

(ii) **Accessing calendar events:** The application allows the user to see a list of upcoming calendar events from the next seven days. Especially in situations in which the user does not have any access to a computing device, this functionality supports the user in having a quick access to his future events, thus aiming to improve awareness of upcoming important meetings. Begel et al. [BNPL09] examined the collaboration in large scale development teams and found that the most important informations consist of the team members' schedules as well as the features to be implemented in the current project.

Self-awareness is one crucial part in order to improve overall awareness of the ongoing work in a team, but as shown in [KDV07] one of the most important informations in order to improve awareness in development teams is the knowledge about colleagues' ongoing work. Therefore, the `AwareGlass` application includes different functionalities to improve team awareness:

(i) **Scanning Developers:** In order to display information about a team member, the application allows to recognize the developer by scanning a QR code. At this stage of the prototype application, the scanning of such a code is the only possibility to recognize a developer. At a later stage, the scanning of developers is planned to be implemented through facial recognition such that there is no need to have QR codes for each team member.
Scanning developers allows the application to be more unobtrusive than a smart phone application since the user has his hands free. Also, the application is more mobile than an application for the computer and can support situations in which a developer does not have a computer nearby.

(ii) **Accessing developer issues:** After having scanned a developer, the application allows the user to display the current issues of the scanned developer in the same way as the user's own issues. By allowing the user to display the counterpart's issues, the awareness about the overall ongoing work in a project might be improved. Furthermore, the developer's issues can be sorted based on the same criteria as the user's own issues.

(iii) **Finding a meeting time:** In order to support the organization of face-to-face meetings when encountering a developer in the hallway or similar situations, the `AwareGlass` application allows the user to find meeting times in which the scanned developer as well as the user himself do not have any meetings in their calendars. By displaying a list of possible meeting times, the organization of such an appointment is simplified.

3.3 Further Features

The Recon JET glasses are designed with sports applications in mind and therefore various hardware sensors are built into the glasses. We have taken advantage of this fact and have implemented two functionalities that make use of these sensors:

(i) **Glancing Recognition:** In order to facilitate using the scanning functionality of our application, we have implemented a so-called "Scanning Mode". This functionality allows the user to activate the scanning of developers beforehand, e.g. before an important meeting, and by glancing at the screen, it is activated automatically. This allows the user to focus on the ongoing meeting without having to press any buttons. As soon as the user stops looking at the screen, the application stays in an idle mode, waiting for the user to glance at the screen again and restart the scanning. Since the glancing recognition feature does not require the user to operate the application with his hands, `AwareGlass` does not disturb the interactions with other developers. The user can fully concentrate on the informations given in a meeting, rather than having to use his hands to operate the device.

(ii) **Movement Recognition:** Due to the many sensors built into the Recon JET glasses, it is possible to find out if the user's position is stationary or if he is moving. `AwareGlass` uses this fact in that the application recognizes when the user is walking and displays the next upcoming calendar event. This functionality supports the user in seeing the important details of the next upcoming event without using any device such as a smart phone, with the goal to improve self-awareness and time management. Recognizing when the user walks, enables situational awareness and thus we can display the information at the right time and situation. Since this feature does not require the user to operate the device, he is more aware of his environment and can quickly glance at the screen when he needs the information.

3.4 Evaluation

In the following, we discuss the method used to evaluate this thesis as well as the objectives of the evaluation. In chapter 5 we present the results of this evaluation and the conclusions which can be drawn. The evaluation of this thesis consists of three interviews conducted with experienced developers. One of the developers is working at a rather small company, whereas the other two developers work in a large software development company consisting of multiple teams, organized in task forces and product development teams. The participants are experienced software developers, working between 15 and 30 years in software engineering. All of them have at least ten years of experience with issue tracking systems and make use of them at least once a week at work.

3.4.1 Method

The evaluation is designed in a hybrid fashion, consisting of two separate parts:

(i) **Acceptance Tests:** In order to gain insights of the prototype application, we conducted a short acceptance test with questions concerning the overall usability of the application and the implemented functionalities.

(ii) **Semi-structured Interviews:** By using the prototype application as a starting point, we conducted semi-structured interviews with the same developers, asking them about their behavior in their daily work. The application should present the interviewees what may be possible with such a device and what they could use the device for. Through these semi-structured interviews, we further elaborated on the awareness of oneself and of team members.

3.4.2 Objectives

One of the most important questions that should be answered with this evaluation is what information is needed in situations without access to a computer. If we were able to find out what information is needed in these situations, we could further improve the prototype application in future work. Also, we want to find out, how exactly we can support these situations to foster awareness within development teams and of oneself. Furthermore, we want to analyze in which specific situations developers can profit from displaying awareness information with `AwareGlass`. Finally, by conducting an acceptance test we aim to detect technical flaws and to receive suggestions to improve the user experience of `AwareGlass`.

Chapter 4

Prototype Implementation Details

In this chapter the prototype implementation is discussed in more detail. Especially, the peculiarities concerning the user interface design (Section 4.1), the functionality implementation (Section 4.2) and the data model (Section 4.3) are of interest. The goal of this chapter is to provide the reader with technical implementation details in order to fully understand how the application is designed and implemented.

4.1 User Interface

This section is dedicated to the user interface implementation of the `AwareGlass` prototype application. The user interface (UI) is a crucial part of how the user interacts with the application, especially when considering the small screen of the Recon JET glasses. Figure 4.1 shows the early mock-up of `AwareGlass`. The UI consists of a menu screen which acts as an entry point for the application. Furthermore, we have designed list views for the issue lists, both for the user's issues as well as for the scanned developer's issues. The calendar view has drastically changed from the mock-up to the final implementation, considering that in the early stage, where the mock-up was created, the implementation of a "Find Meetings" functionality was not taken into account.

Controlling the application is different in that the user does not have many possibilities to interact with the application, namely a "back" and a "confirm" button as well as a touch sensitive directional pad. Therefore, the UI design is based on the design of `Recon OS`, the operating system created by Recon Instruments. The following subsections further discuss the building blocks of the user interface design.

4.1.1 Color Scheme

The color scheme we have used in the `AwareGlass` application is based on the `Recon OS`. We have used three colors:

(i) **Dark gray**, used as the main background color

(ii) **Orange**, used to highlight selectable items.

(iii) **White**, used as the main text color. In particular for textual representations of non-selectable content.

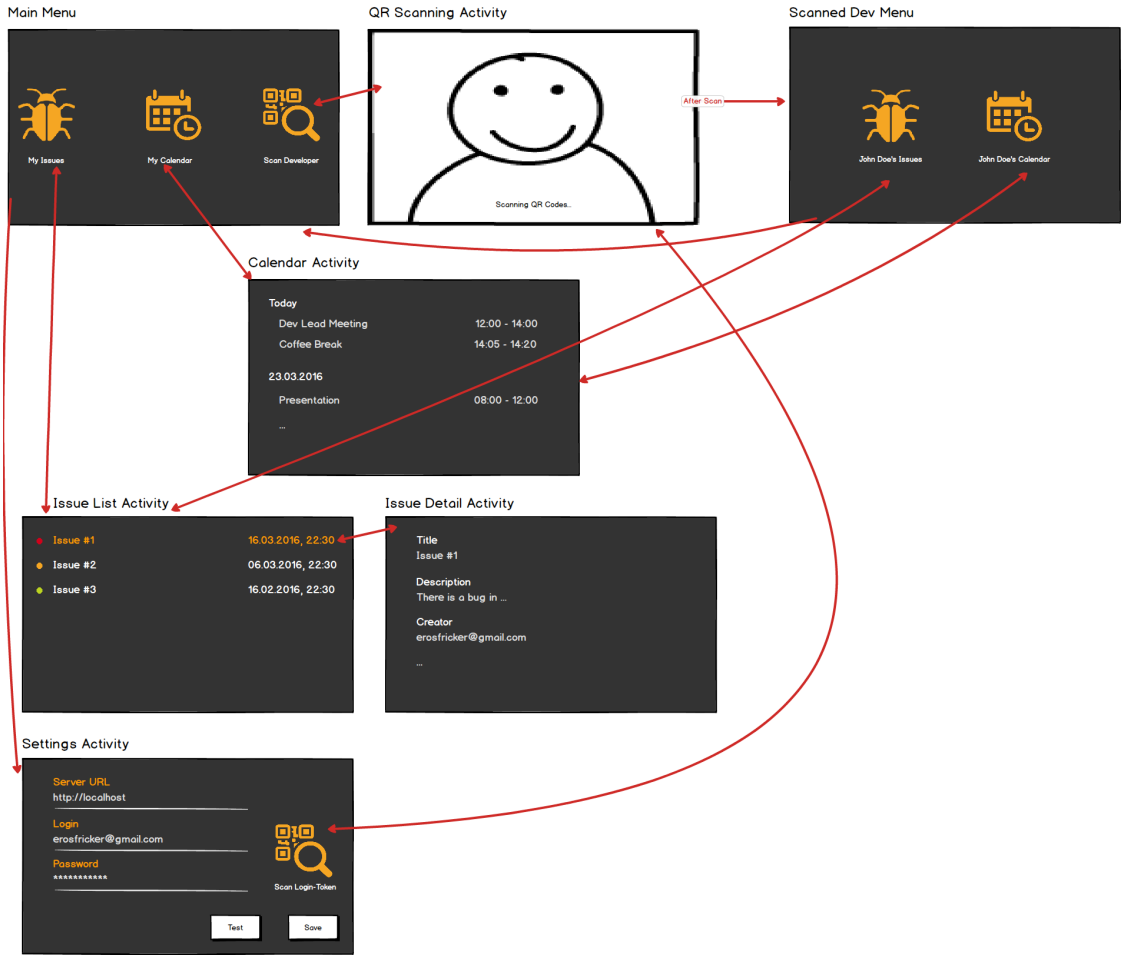

Figure 4.1: An early mock-up of the `AwareGlass` application

The chosen colors allow us to help the user distinguish between read-only content and interactable content.

4.1.2 Navigation Structure

Since the Recon JET glasses do not have many possibilities for the user to interact with our application, it is important to provide a simple navigation structure in `AwareGlass`. Figure 4.2 shows the application's navigation patterns in more detail. The navigation hierarchy consists of a main menu including the most important functionalities of the application and when the user has scanned one of his colleagues, another menu is presented which contains the developer-related functionalities of the application.

4.1.3 Controls

To navigate in the application, the Recon JET glasses provide a touch sensitive directional pad to swipe between menus and two buttons, a "back" button to navigate back in the application and a "confirm" button to emulate selecting a view such as a list item or an on-screen button. Since the hardware controls are limited to the aforementioned actions, the navigation within the application had to be designed to be as simple as possible. Swiping on the directional pad allows the user to switch between menu items and list rows.

One of the most difficult decisions in terms of control scheme was how to trigger the sorting view shown in figure 4.3. Since the screen size is minimal, it was not possible to use a typical Android navigation drawer view to display the sorting possibilities. To solve this problem, we used a swiping gesture to the left or to the right to trigger the sorting overlay, which allowed us to use the complete screen to display the sorting criteria. As can be seen in figure 4.4, this overlay is based on the `Recon OS` design language.

4.2 Functionality

This section describes the functionality of `AwareGlass` in more detail. It is organized in sections, each describing a particular functionality of the application.

4.2.1 Accessing Issue Tracking System Information

One of the most important functionalities of `AwareGlass` is the access to the issue tracking system in order to display useful information based on the data gathered from the issue tracking server.

Accessing User's Issues

In order to display the issue tracking information, the user can click on the menu item "My Issues" (see Figure 4.5). The application then executes an asynchronous call to the issue tracking system's REST API, in our case `Bugzilla` [bug16]. As soon as the asynchronous call returns, its delegate is notified that the information has been successfully gathered. The `Bugzilla` server returns a JSON object as a String, which is then parsed into a Java object with the help of `Gson` [Goo16]. While this procedure is running in the background, the user sees a **`ProgressBar`** which informs him that the information is being loaded.

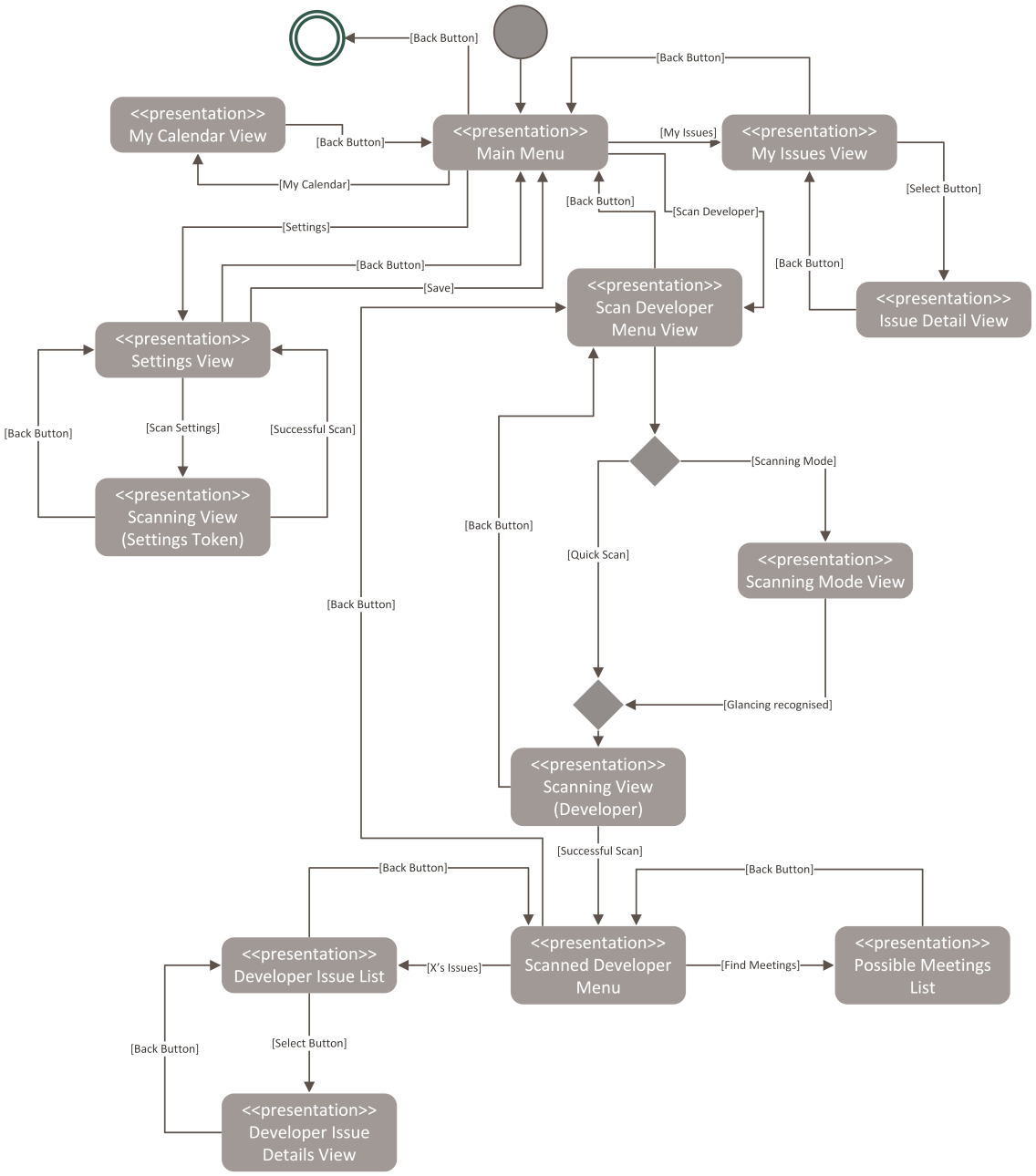

Figure 4.2: The user navigation model of the application

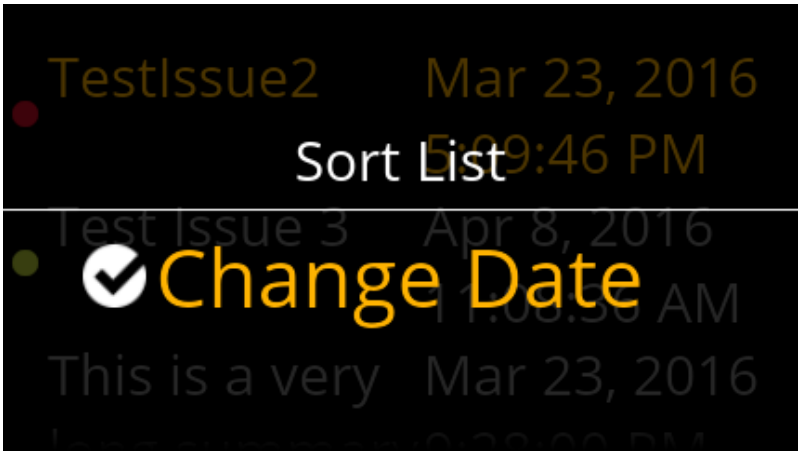

Figure 4.3: The view when the user triggers the sorting of the list

Figure 4.4: The `Recon OS` overlay in the settings menu

Figure 4.5: The main menu with the selected "My Issues" menu entry.

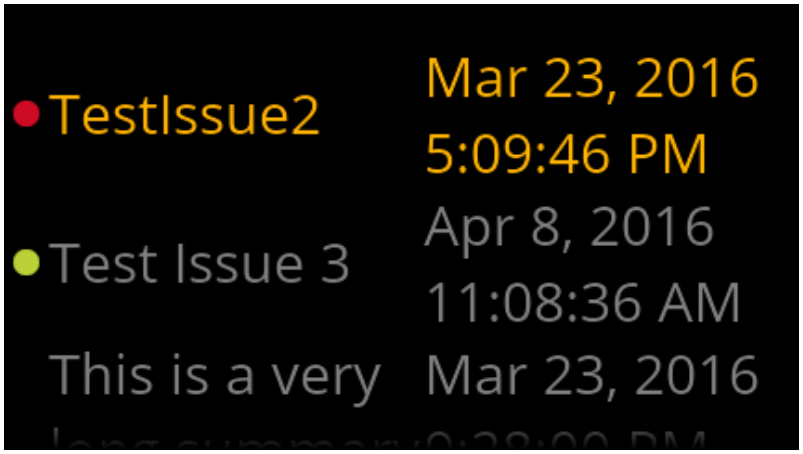

Figure 4.6: The user's issues list

After the JSON strings have successfully been parsed into Java objects, the **`ListView`** of the issues is presented to the user (see Figure 4.6). The **`ListView`** rows display the most important information such as the issue's status on the left hand side of the row, simplified with a colored dot. The dot is green if the issue has been marked as resolved, orange if the issue is in progress and red if the issue has only been confirmed as of now. By representing the issue's status with a traffic light analogy, we were able to reduce the space needed for this information to a minimum. Next to the issue's status information, the title of the issue is displayed. This information helps identifying the issue at a glance. On the right hand side of a list row, the application displays the last change date of an issue. This information may be helpful in situations where the user is asked when he has changed the issue for the last time. Also, this information may help the user viewing the last updated issues and being more aware of the current changes of an issue. The sequence diagram in Figure 4.7 demonstrates the procedure calls and delegate mechanisms.

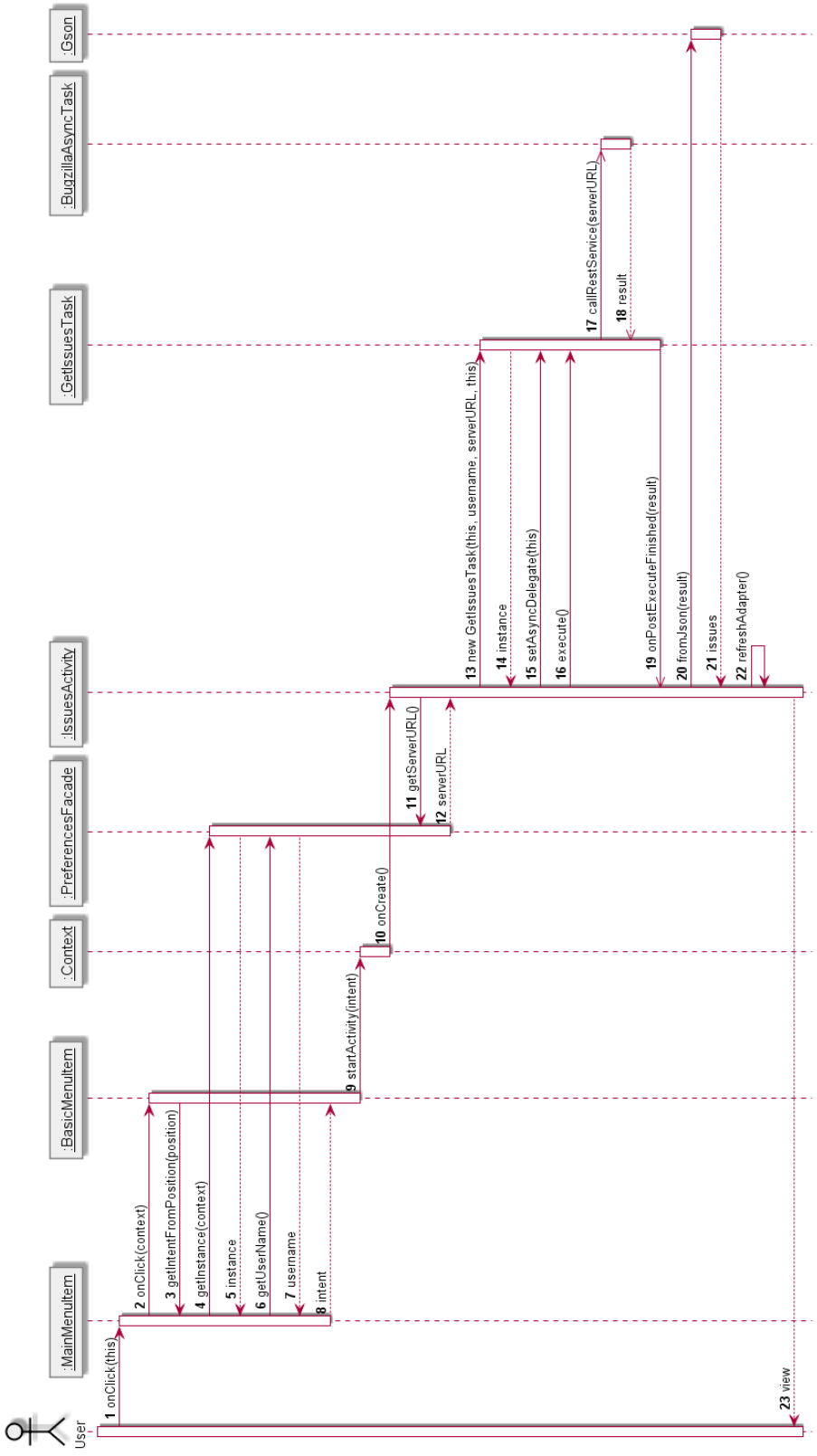

Figure 4.7: The sequence diagram of the process of retrieving the user's issues

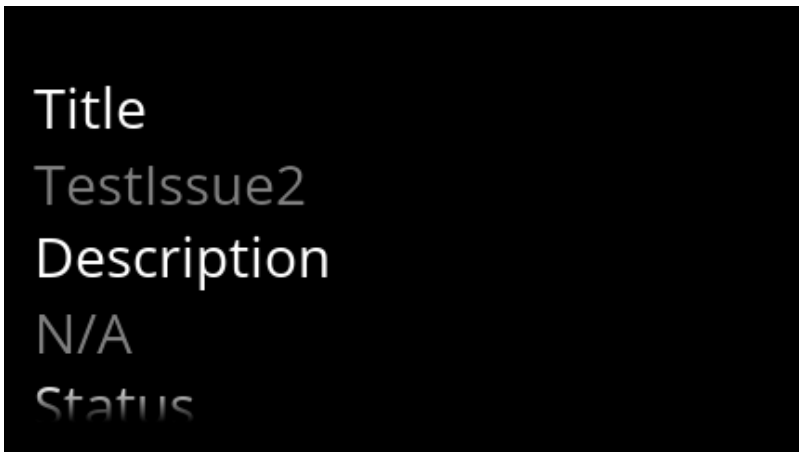

Figure 4.8: The details of an issue showing the title and the description of the issue.

The main issue list can be sorted, based on three criteria:

(i) **Name**: The issue list can be sorted by the issues' names in alphabetical order.

(ii) **Status:** The issues are sorted based on the issues' statuses, from "Confirmed" to "In Progress" up to "Resolved". This is the standard sorting criterion.

(iii) **Last Change Date**: The issues are sorted by last change date. From the most recent change date to the oldest change date.

In order to trigger the sorting view (see Figure 4.3), the user can swipe to the left or to the right on the directional hardware pad. Since the Recon JET glasses do not have many hardware controls, we have overwritten the standard functionality of the left / right swipe, which was used as a normal swipe gesture. The sorting overlay was the simplest solution to enable the user to sort the list view, preserving the readability of the texts. The currently selected sorting criterion is shown with a check mark. When the user selects a sorting criterion, the sorting strategy (see Section 4.4.2) is applied and the issue list is reloaded without making another call to the `Bugzilla` REST API.

If the user wants to know more about an issue, he may use the "Select" hardware button to select a specific issue and to display a static list of detailed informations about the issue (see Figures 4.8 and 4.9). This scrollable **`ListView`** is a static representation of all the existing informations of an issue and may support the user in finding detailed informations, which are not displayed in the main issue list.

Accessing Scanned Developer's Issues

Besides accessing the own issue tracking information, `AwareGlass` allows the user to access informations about a team member by scanning a QR code representing the developer. In order to scan a developer, the user needs to select "Scan Developers" in the main menu screen (Figure 4.10). The user then has two possibilities: Either he selects "Quick Scan" with which it is possible to scan a developer once or he selects "Scanning Mode" which starts the scanning mode of `AwareGlass` (see Section 4.2.2 for more information about the scanning functionality).

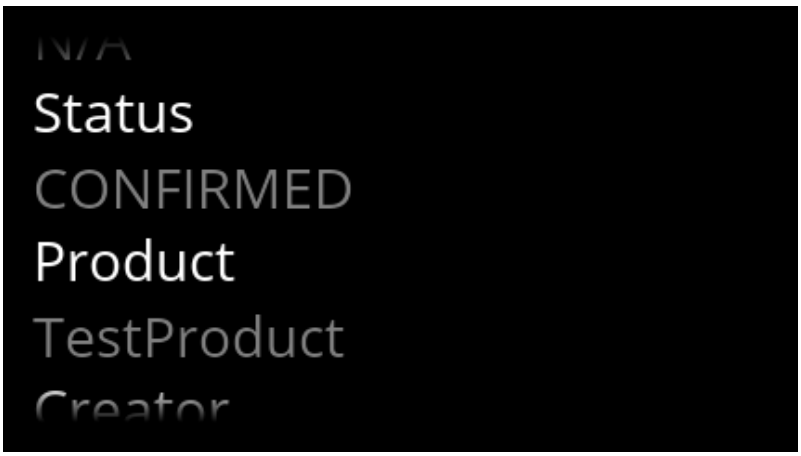

Figure 4.9: The details of an issue showing the status and the corresponding product of the issue.

Figure 4.10: The main menu with the selected "Scan Developers" menu entry

As soon as a developer has been successfully scanned, the application executes an asynchronous call to the issue tracking system's REST API. With this call, the scanned developer is searched in the `Bugzilla` user database and the stored real name is returned. Then, the user is presented with the "scanned developer's menu" (see Figures 4.11 and 4.12) which has the same user interface design as the main menu. In this menu, the user can select "X's issues" (with "X" being the real name of the developer) and he is presented the same **`ListView`** as in his own issue list. The user has the same possibilities to sort the issues and view the details of an issue. The process of retrieving scanned developer's issues is similar to the process of retrieving the user's issues (see Figure 4.7).

4.2.2 Scanning Developers

The scanning functionality of `AwareGlass` is one of the most important features of the application, both in terms of application usability as well as having a unique characteristic when compared to other information media such as smart phones and computers.

Figure 4.11: The scanned developer's menu with the menu entry "X's issues" highlighted

Figure 4.12: The scanned developer's menu with the menu entry "Find Meeting Time" highlighted

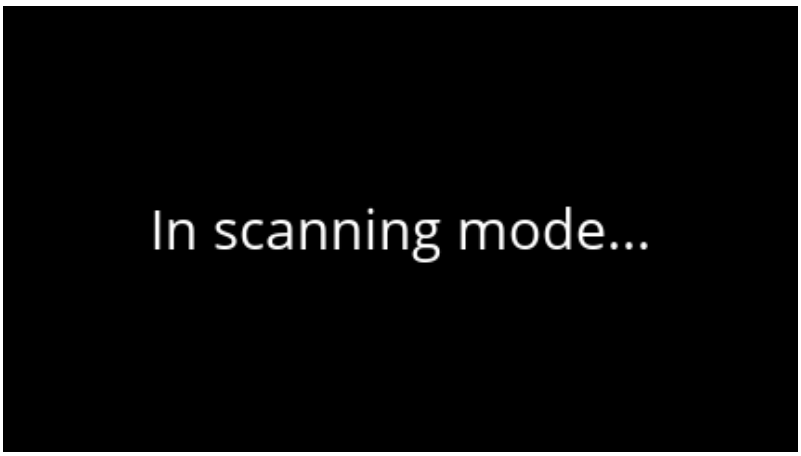

Figure 4.13: The application in the "Scanning Mode" state

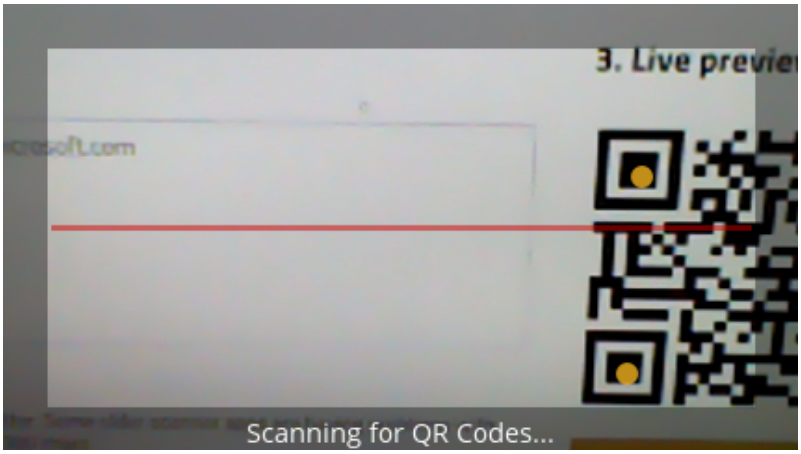

Figure 4.14: The user's view when searching for a developer (QR code) to scan

The application is designed to mimic the user's view while scanning developers.

By selecting "Quick Scan", the user may manually start the scanning process. When selecting "Scanning Mode", the application is put into an unobtrusive scanning mode and whenever the user glances at the screen, the scanning process is started (see Figure 4.13). The scanning mode can be used in situations in which the user does not want to operate the application with his hands and wants quick access to the scanning functionality.

The scanning view displays what the Recon JET's camera is recording. As soon as a QR code is recognized, the user is notified with visual feedback in the form of orange points, demonstrating the scanning process (see Figure 4.14). As soon as the scanning has successfully finished, the user is notified with a beep and the scanning view is replaced with the scanned developer's menu.

The scanning functionality is implemented with a stand-alone port of the `ZXing` library [Jou16]. It was not possible to use the standard `ZXing` library since the Recon JET glasses do not have the

Figure 4.15: The main menu with the selected "My Calendar" menu entry

`Google Play Services` installed and the standard library needs access to the `Google Play Store` in order to enable the scanning process. The glancing recognition's core implementation is part of the `Recon JET API`.

4.2.3 Accessing Calendar Information

In order to further improve awareness about ongoing work in development teams, we implemented the possibility to access calendar events. The user is able to access his own calendar events such that he can easily see his daily schedule and the application provides the possibility to find meeting times with a scanned developer, without manually having to find a meeting time that suits both team members.

Accessing User's Calendar Events

With `AwareGlass`, the user can access his own calendar events by selecting "My Calendar" in the main menu (see Figure 4.15). The application calls the `Office 365 REST API` in order to access the calendar event informations of the user. This call is executed with the `EWS Android API` [Lip16] which implements the ability to access `Microsoft Exchange` Servers and authenticate the user with his credentials. The `Exchange` server returns a list of calendar events which are then displayed to the user in a custom **`ListView`** (see Figure 4.16). This mechanism is presented in the sequence diagram in Figure 4.17.

The list view contains two types of rows:

(i) **Title rows**: These rows are used to display the date to which the calendar events belong. They serve as a separator for the calendar events.

(ii) **Event rows**: Event rows are used to represent a single calendar event. They consist of a title text view on the left and the time range in which the event takes place on the right.

By swiping up and down, the user can scroll through the calendar events and have an overview of his upcoming events of the next seven days.

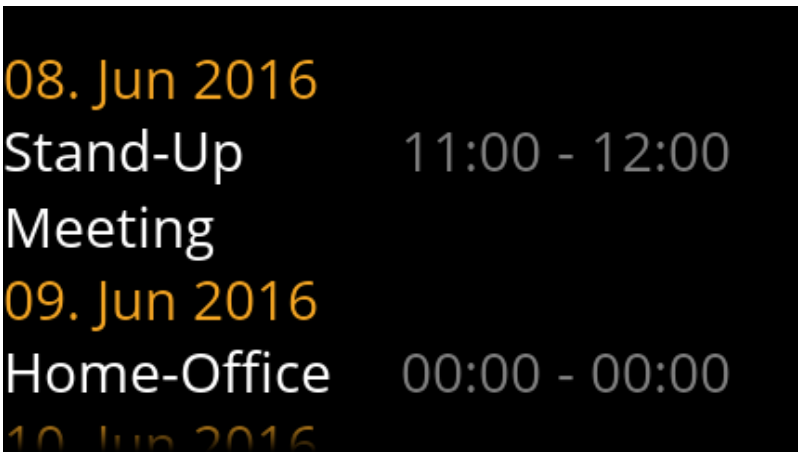

Figure 4.16: The list of the user's upcoming events

Finding Meeting Times

When the user has scanned a developer with his corresponding QR code (see Section 4.2.2), he can select "Find Meeting Time" in the scanned developer's menu in order to find a meeting time that fits both team members. As soon as the user has selected this menu item, the application calls the `Office 365 REST API` twice, once for the user's calendar events and once for the scanned developer's events. In order for this functionality to work, the scanned developer needs to give read-access to the user via the Outlook Online service (`http://outlook.office365.com`).

As soon as the calendar events of both developers are collected, the asynchronous task's delegate is called – in this case the "Find Meeting Time" activity – which then starts the algorithm to calculate the meeting times of the two parties involved. Since the Recon JET glasses have rather limited hardware capabilities, this process may take some time, depending on the number of events and how they overlap.

When the possible meeting time ranges have been calculated, `AwareGlass` displays a **`ListView`** with rows presenting these ranges (see Figure 4.18). The sequence diagram in Figure 4.19 further elaborates on the procedure calls of this process. This list consists of the same types of rows as in the user's own calendar event list. The "title rows" display the date of the possible meeting time, while the "event rows" display the time ranges that are possible on this date.

The functionality of finding possible meeting times without having to manually compare calendars does not only support awareness within development teams but also means more efficiency in day-to-day situations. All of the participants of our exploratory study would appreciate having such a functionality in their daily work and would even replace a smart phone with the Recon JET glasses because of this functionality (see Chapter 5 for further information).

4.2.4 Movement Detection

Since the Recon JET glasses are primarily constructed for sports applications, we wanted to make use of the built-in sensors of the glasses. With the Recon JET API it is possible to detect movements of the user by using the hardware sensors. We have implemented a movement detection

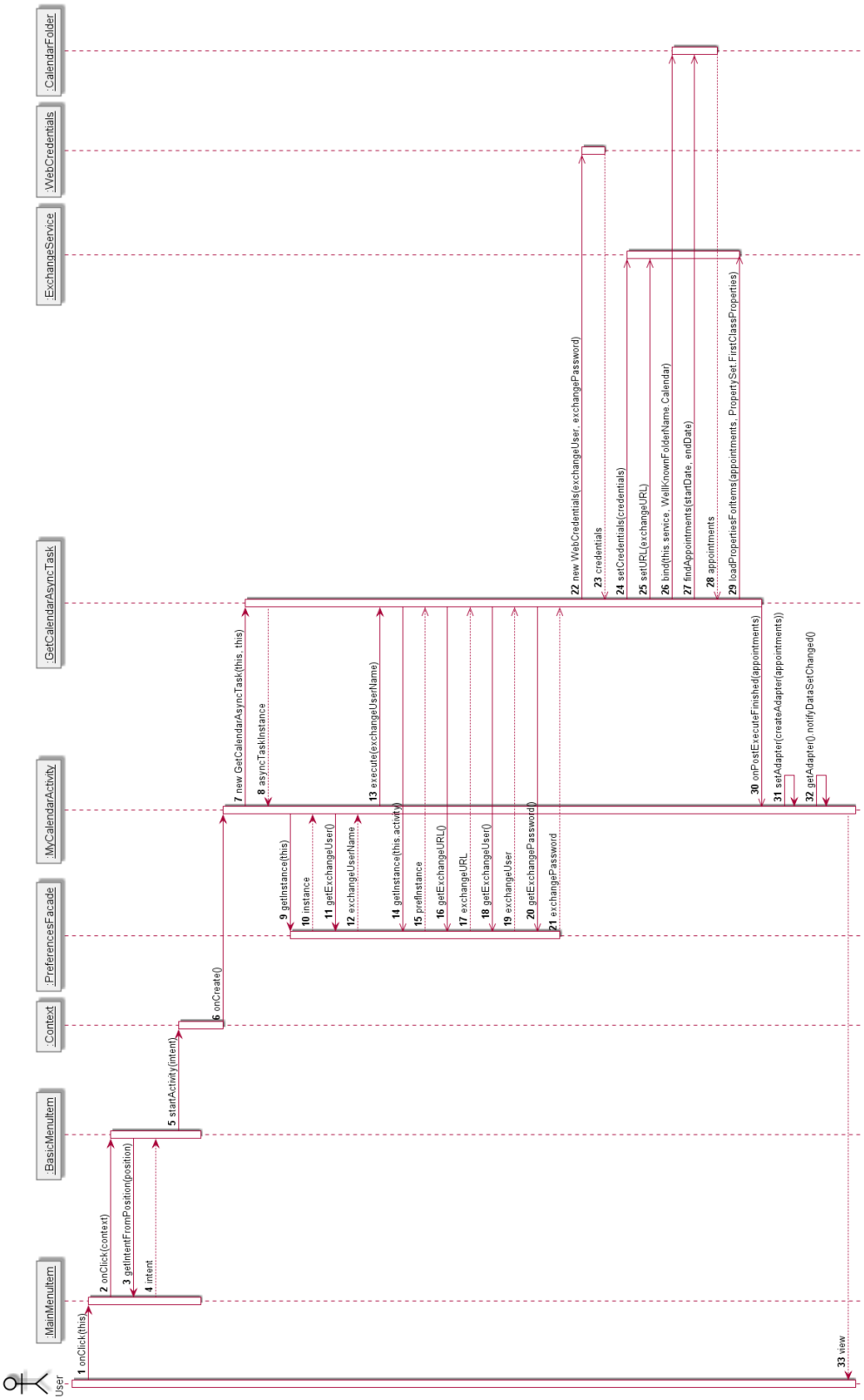

Figure 4.17: The sequence diagram of the process of retrieving the user's calendar events

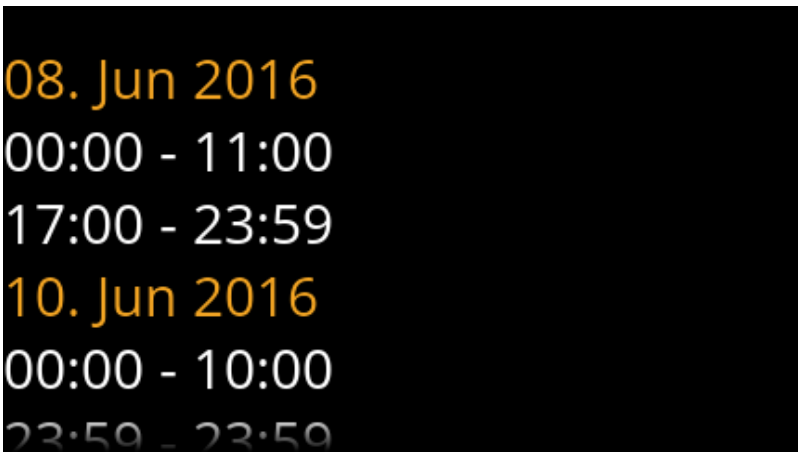

Figure 4.18: The list of the free meeting times of both developers calculated by the algorithm

which works as follows: When the application is in a neutral state in the main menu, i.e. it most probably is not in active use, a movement event listener waits for user movements. As soon as a horizontal speed of more than 5.0 km/h is detected, the application automatically displays the "My Calendar" view, which displays the user's upcoming calendar events. When the horizontal speed decreases below 5.0 km/h, the application returns to its normal state, i.e. to the main menu. The implementation of the movement event listener is built into the `Recon OS` and uses the built-in GPS sensors to calculate the movement speed. We have pinpointed the exact speed to be used for detecting a walking movement through trial and error.

The main idea behind this implementation is that the user is supported in his self-awareness when he is walking. Especially, when the user is on the way to his next meeting, he can see what the meeting title is and when it starts.

4.2.5 Application Settings

Since `AwareGlass` connects to two independent server instances (a `Bugzilla` and a `Microsoft Exchange` server instance), the authentication information needs to be persistent as well as accessible to the user. When the user selects "Settings" in the main menu (Figure 4.20), a list of editable settings is displayed (see Figure 4.21). The preferences view consists of the following entries:

(i) **Server URL**: The URL for accessing the issue tracking server instance.

(ii) **User Name**: The issue tracking server user name to authenticate.

(iii) **Password**: The issue tracking server password to authenticate.

(iv) **Exchange Server URL**: The URL for accessing the `Microsoft Exchange` server instance.

(v) **Exchange User Name**: The `Exchange` server user name to authenticate.

(vi) **Exchange Password**: The `Exchange` server password to authenticate.

(vii) **Movement Detection**: Turning the movement detection functionality on/off.

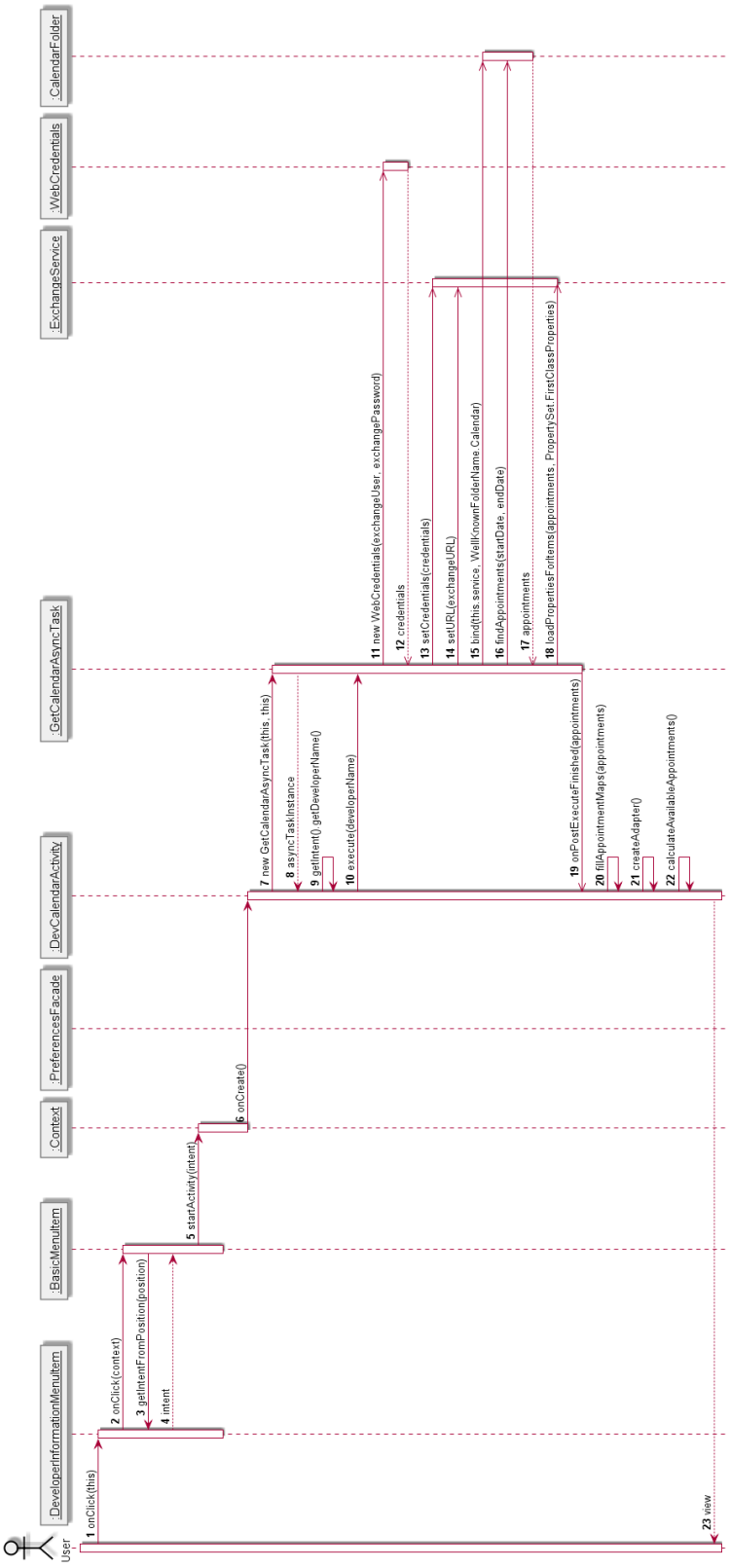

Figure 4.19: The sequence diagram for the process of finding meeting times with the scanned developer

Figure 4.20: The main menu with the selected "Settings" menu entry

Figure 4.21: The list of editable settings in the "Settings" view

The user can manually edit these preference entries by selecting the **`EditText`** views and by swiping up and down, he can scroll through the list.

Entering the preferences manually with the help of the built-in software keyboard is a cumbersome process. Therefore we have implemented a "Scan Settings" functionality (see Figure 4.22): When the user clicks on the button "Scan Settings", a similar view to the "Scan Developers" view is displayed. The user can then scan a QR code, i.e. a "settings token", instead of entering the preferences manually. The settings token consists of all the entries, separated by a semicolon.

As soon as the settings token was successfully scanned, the QR code is converted into strings and finally saved in Android's **`SharedPreferences`**. The use of **`SharedPreferences`** allowed us to persist the preferences in a secure way, even after the application has stopped running (either because the user has stopped the application or because the OS freed resources, stopping the application). Figure 4.23 elaborates on the process of saving the preference entries.

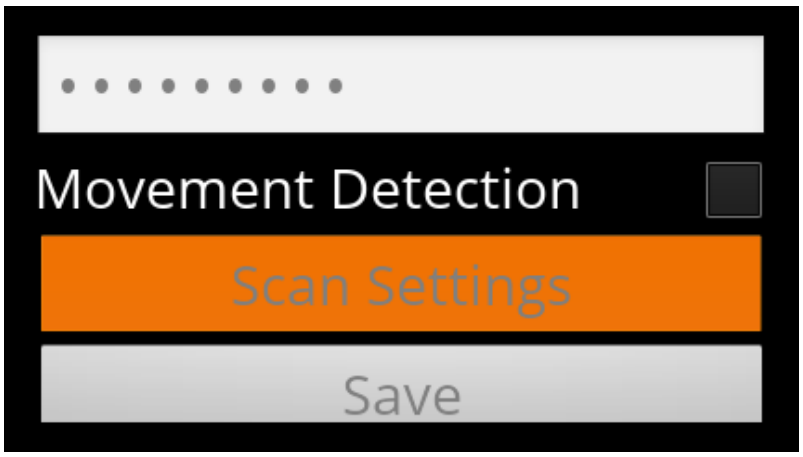

Figure 4.22: The button to start the scanning of the settings token

4.3 Data Model

This section describes the application's underlying data model. The data model has to be consistent with the data models accessed from the two sources described in the following subsections.

4.3.1 Bugzilla Issues

One of the main functionalities of `AwareGlass` is accessing issues from an issue tracking system. The data needed is retrieved by using calls to the `Bugzilla` server instance's REST API which defines the structure of the data. Since our prototype application is based on Android, i.e. on Java code, the data needs to be parsed from JSON to Java objects. To simplify this task, we use Google's `Gson` [Goo16] library. Figure 4.24 shows our data model used to represent the `Bugzilla` issues. It consists of the following classes:

(i) **Issue**: The **`Issue`** class represents a single instance of an issue obtained from `Bugzilla`. It contains several fields to access the detailed informations of a `Bugzilla` issue.

(ii) **IssueRestResult**: The **`IssueRestResult`** class contains a list of all the issues that have been gathered from the call to the `Bugzilla REST API`. This class is mainly used in order to be able to parse the JSON data with `Gson` in a simple way.

(iii) **IssueInformationTuple**: The **`IssueInformationTuple`** class consists of a title and a description field and is used for representing a list view row in the issue's detail view.

(iv) **IssueDetailInformation**: The **`IssueDetailInformation`** is a wrapper class for the **`IssueInformationTuple`** instances. It was created in order to fill the static list view when displaying the detail information of an issue.

(v) **IssueStatus**: The **`IssueStatus`** enumeration represents the status of an issue, which may be one of the following:

- CONFIRMED: The issue was confirmed.
- IN_PROGRESS: The issue's resolution is in progress.

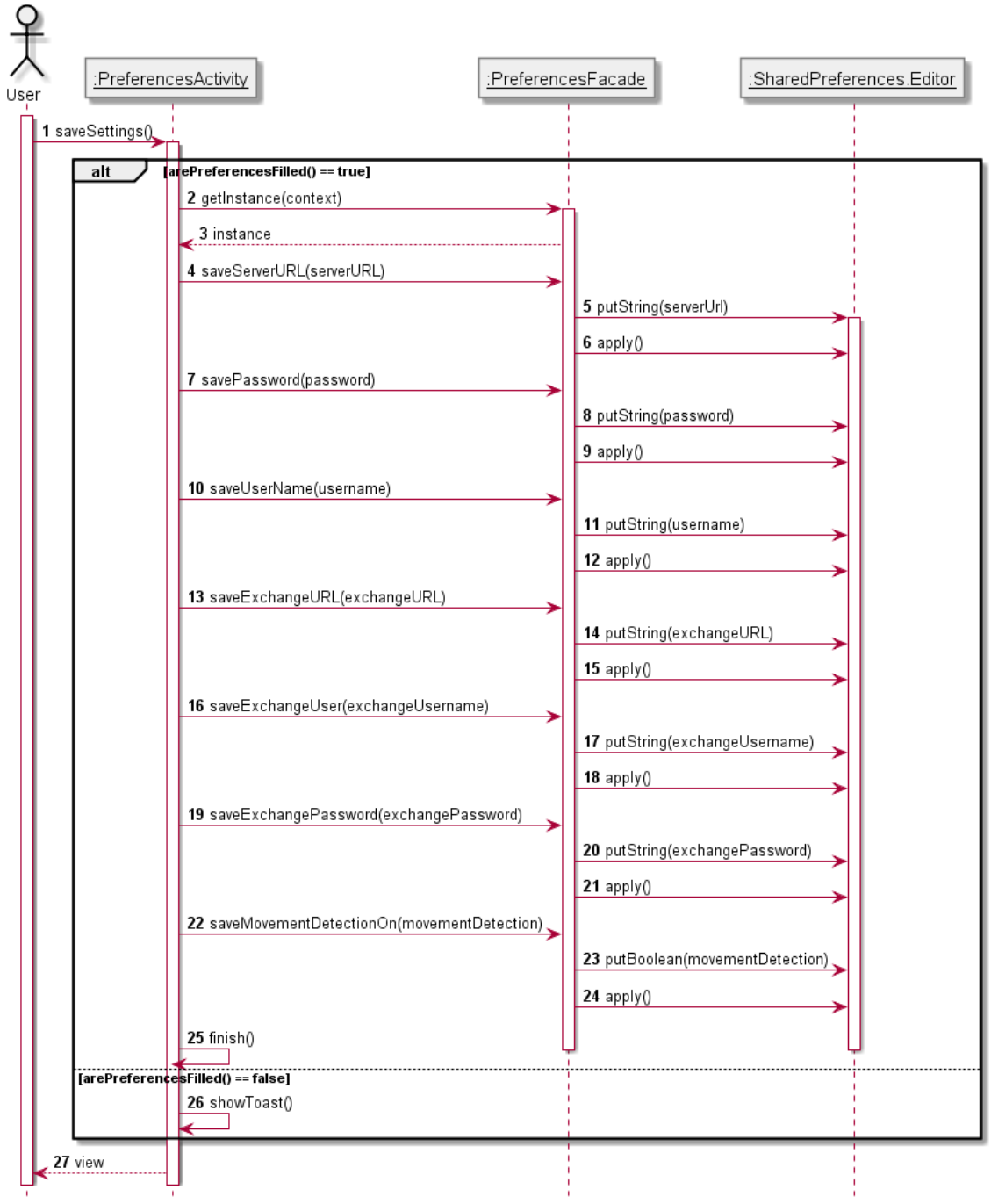

Figure 4.23: The process of saving the user preferences to Android's `SharedPreferences`

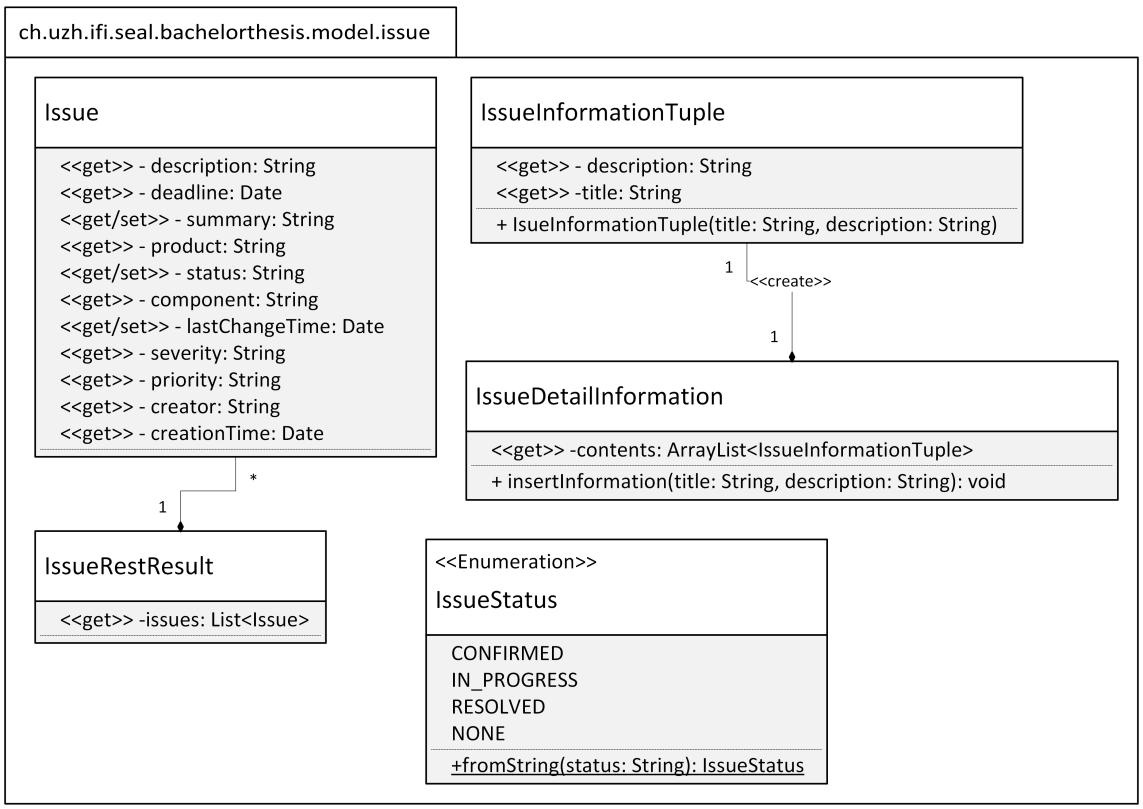

Figure 4.24: The data model for the issues

- RESOLVED: The issue has been successfully resolved.
- NONE: The issue does not have any status information.

Furthermore, the class contains a helper method in order to create an instance of the **`IssueStatus`** enumeration from a **`String`** given. This is used when parsing the JSON data.

4.3.2 Calendar Events

Another main functionality of the `AwareGlass` application is the access to calendar event data of the user and of the scanned developer. This information is obtained by calling the REST API of the `Microsoft Office365 Exchange` servers [Mic16]. In order to facilitate the implementation of the calendar access, we used the `Exchange Webservice Android API (ews-android-api)` [Lip16] which is a repackaged version of the `EWS Java API` [Dev16] for the Android operating system.

The `ews-android-api` [Lip16] contains methods for accessing the `Exchange REST API` as well as the data model needed for the representation of calendar events in Java. See Listing 4.1 for an example implementation of the calendar access.

```
/**
* Gets the events from the App user
* (address stored in {@link android.content.SharedPreferences})
* @param startDate The start date to get the events
* @param endDate The end date to get the events
* @return The events from the App user between startDate
* and (including) endDate
*/
private ArrayList<Item> getUserCalendar(Date startDate, Date endDate) {

   //Set up the exchange service instance and credentials
   ArrayList<Item> appointments = new ArrayList<>();
   CalendarFolder cf;
   ExchangeService service =
      new ExchangeService(ExchangeVersion.Exchange2010_SP2);
   final PreferencesFacade preferencesFacade =
      PreferencesFacade.getInstance(this.activity);
   String exchangeURL = preferencesFacade.getExchangeURL();
   String username = preferencesFacade.getExchangeUser();
   String password = preferencesFacade.getExchangePassword();
   ExchangeCredentials credentials =
      new WebCredentials(username, password);
   service.setCredentials(credentials);
   try {
      service.setUrl(new URI(exchangeURL));
   } catch (URISyntaxException e) {
      ... //Handle exception
   }

   //Get the calendar appointments
   try {
      cf = CalendarFolder.bind(service, WellKnownFolderName.Calendar);
      FindItemsResults<Appointment> findResults =
         cf.findAppointments(new CalendarView(startDate, endDate));
      for (Appointment appt : findResults.getItems()) {
         appointments.add(appt);
      }
      if(appointments.size() > 0) {
         service.loadPropertiesForItems(appointments,
                                 PropertySet.FirstClassProperties);
      }
   } catch (Exception e) {
      ... //Handle exception
   }
   return appointments;
}
```

Listing 4.1: A sample implementation of the calendar access on the `Exchange` server (adapted from the **`GetCalendarAsyncTask`** class implementation)

Furthermore, we have implemented a wrapper class in order to represent date ranges (**`DateRange`**) used for the implementation of the "Find Meetings" functionality (see Figure 4.25). This class consists of a getter method for a comparator and of a start date and end date field.

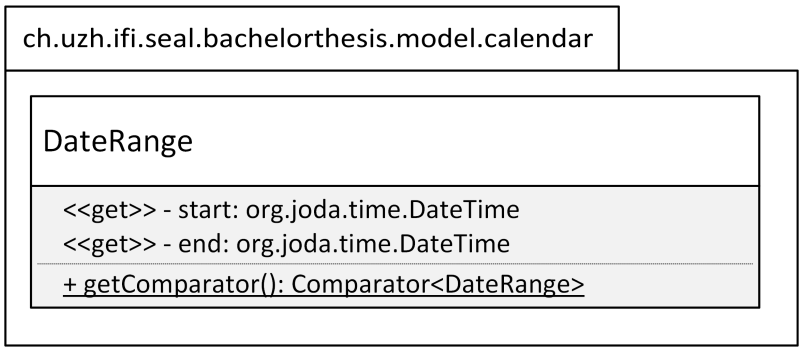

Figure 4.25: The data model for the date ranges in the calendar package

4.3.3 Bugzilla User

In order to be able to implement the functionality of retrieving the real name of a scanned developer in the scanned developer menus, we have used the same pattern as with the **`Issue`** and **`IssueRestResult`** classes. We implemented the following classes to represent a `Bugzilla` user:

(i) **UserRestResult**: This class is used to represent the result of the REST API call to gather user information. It consists of a list of **`User`** instances and is used to facilitate parsing the JSON data.

(ii) **User**: The **`User`** class is used to represent a `Bugzilla` issue tracking system user. It consists of a field for parsing the real name of the user as well as of the field "name" which represents the e-mail address of the user as of the JSON data from `Bugzilla`.

The class diagram for the "user" package is shown in Figure 4.26.

4.4 Design Patterns

This section describes the design patterns we have used in our prototype application. We also discuss why we have decided to use these patterns and why they are appropriate.

4.4.1 Facade

The first design pattern we have used is the Facade pattern. As described in [GHJV95], the facade provides a simplified interface to a larger body of code. We have used this pattern to model the access to Android's **`SharedPreferences`**. The **`SharedPreferences`** are used to persist key-value data, in our case the `Bugzilla` and `Exchange` server settings. The **`PreferencesFacade`** class implemented in our prototype application simplifies the access and the management of these shared preferences.

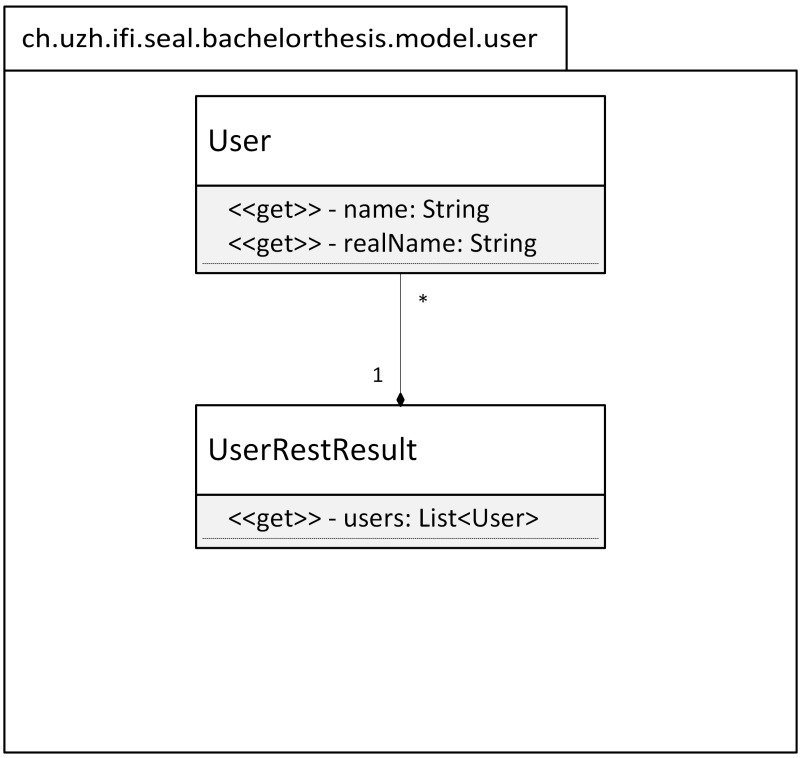

Figure 4.26: The data model for the user package

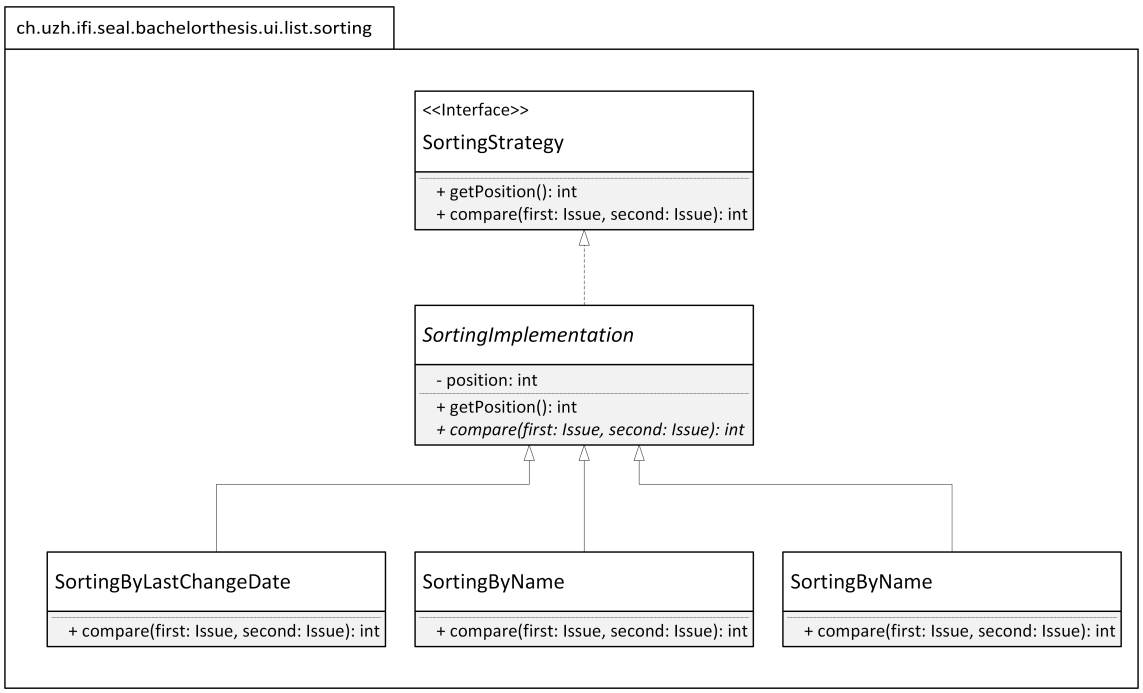

Figure 4.27: The class diagram for the sorting strategy design pattern in `AwareGlass`

4.4.2 Strategy

Another "Gang of Four" design pattern we have used in our application is the Strategy pattern. The Strategy pattern makes algorithms interchangeable with each other and allows us to select a specific algorithm at runtime [GHJV95]. The sorting of the `Bugzilla` issue list is implemented by providing sorting strategies, i.e. sorting algorithms which can be interchanged at runtime and therefore allow our implementation to be more loosely coupled and thus more maintainable (see Figure 4.27).

4.4.3 Delegation

Our prototype application heavily relies on delegation. As described in [GHJV95], "delegation is a way to make composition as powerful for reuse as inheritance". We have used delegation with all of our **`AsyncTask`** classes, which serve the purpose of executing asynchronous calls to the REST APIs. By using delegation, we were able to notify the delegate classes – mostly the classes that have started the asynchronous task – as soon as the asynchronous call has been successfully completed. This is useful in order to update the user interface as soon as the data has been gathered from the server, without blocking the UI thread.

4.4.4 Factory Method

Depending on the menu item the user selects, the application needs to create a different **`Intent`** and thus we used a Factory Method design pattern. The Factory Method pattern "defines an interface to instantiate an object, but lets subclasses decide which class to instantiate"[GHJV95],

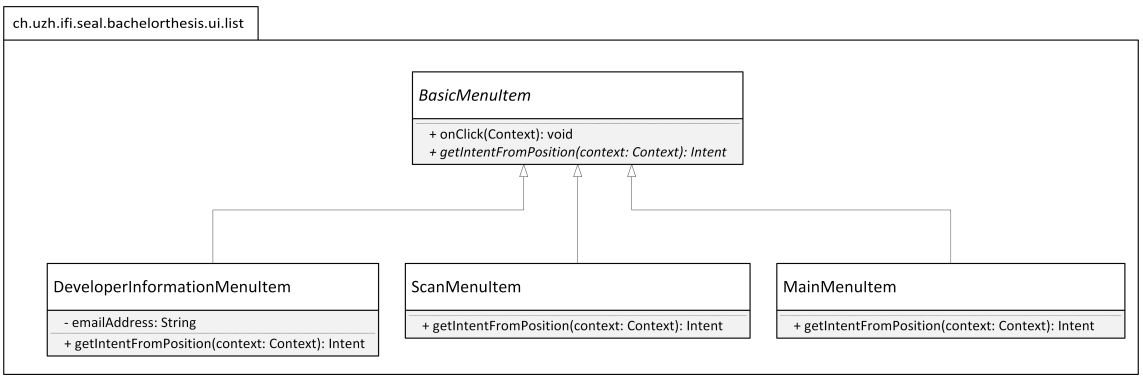

Figure 4.28: The Factory Method design pattern applied to the implementation of the menu classes

which helped us to keep the instantiation of new Android **`Activity`** instances maintainable. Figure 4.28 shows the class diagram of this pattern.

4.4.5 Singleton

The **`PreferencesFacade`** class is implemented based on the Singleton design pattern which is presented in [GHJV95]. This pattern allows us to control the instantiation of an object of this class and through the static **`getInstance()`** method, we can make sure that there exists only one instance of the class at runtime. We implemented the Singleton pattern because the access to Android's **`SharedPreferences`** should only ever be done from within one instance of our facade class.

4.5 Package Structure

The implementation of `AwareGlass` is structured into packages based on their layer. The first package is the **`model`** package which is structured into **`calendar`**, **`issue`**, **`preferences`** and **`user`**. The **`model`** package represents our data model.

The next package is **`rest`** which contains all classes which are used for the REST calls.

The last package is the **`ui`** package which is again split into **`activities`** for the Android **`Activity`** classes and the **`list`** package, which contains the menu item classes and other classes related to the list views. It further contains a **`sorting`** package which contains the sorting strategy pattern classes and the **`SortType`** enumeration.

4.6 Testing

Our application consists of standard `Android SDK` calls, UI implementations as well as REST API calls, thus testing could not only be done by writing `JUnit` unit tests. Therefore we have also made use of the `Android Instrumentation Test` subclasses for testing calls between **`Activity`** classes, since these methods need a working application **`Context`** in order to function. Furthermore, we have implemented UI tests where appropriate, in order to make sure that

our application displays the correct content in the correct form.

Since code coverage tools do not work with Android 4.4 (KitKat) in combination with the `Android Instrumentation Test` classes, we cannot provide code coverage information as of now.

Chapter 5

Exploratory Study

We conducted an exploratory study with three participants of two different companies to investigate the possible usage scenarios of our `AwareGlass` application (see Section 3.1). We first introduced `AwareGlass` and the Recon JET glasses to the participant. Then we conducted a semi-structured interview for about 45 minutes. The goal of these semi-structured interviews was to investigate how software developers would possibly use the application, i.e. in what situations and under what circumstances. Furthermore, we conducted an acceptance test in order to assess and quantify the quality of `AwareGlass`.

5.1 Procedure

The study was conducted in cooperation with two software development companies at their respective locations. The first company is the SBB Informatik AG in Worblaufen, Bern where we interviewed two professional software developers. The SWIT Solutions AG in Wollerau SZ is a rather small company from which we have interviewed the software development lead. For both locations, we conducted interviews with the following procedure:
When the participant arrived, we first introduced our study by explaining the motivation and research questions. The second step consisted of having the participant sign a consent form, allowing us to use the recordings we made for this thesis. Then, we conducted a semi-structured interview with the participant in order to investigate the current usage of issue tracking systems and calendars as well as to discuss possible use-case scenarios for the `AwareGlass` application. Each question block consisted of introducing a feature of the application, i.e. we used the application's feature as a starting point for the discussion, such that the participant is able to understand the possibilities and limitations of the Recon JET glasses. After this semi-structured interview, we conducted a short acceptance test using Likert scales. Specifically, we provided a statement which the participant could rate between 1 (does not agree) to 5 (completely agree) in order to quantify the usability and quality of the application. The transcripts of the interviews as well as the questionnaires can be found in appendix A.

5.2 Participants

In our study, we recruited two participants from SBB Informatik and one participant from SWIT Solutions AG through personal contacts. One of the participants of SBB Informatik is a team leader in a software engineering team and a lead software architect, the other participant is a senior architect ICT. The participant from SWIT Solutions AG is the lead software engineer and

architect. All of the participants have worked in a developer position for 15 up to 30 years. The participants from the SBB work with `JIRA` [jir16] on a daily basis and have about ten years of experience with issue tracking systems. The developer from SWIT has worked with the `Serena PVCS` version control system [ser16] for about 15 years and has recently implemented his own issue tracking system tailor-made for the company's needs. The developers at SBB reported to have the most experience with the `IntelliJ` and `Eclipse` IDEs, while the developer from SWIT has the most experience with the `Borland C++` and `Microsoft Visual Studio` IDEs. All of the interviewees rated their programming experience as at least above average.

5.3 Results

In this section, we present the evaluation results of both the semi-structured interviews as well as the quantitative acceptance test. While all of the participants do see the additional value of the application, not all of them would use the application on a daily basis in its current state. All participants remarked that the application still needs improvements in order to be useful for their daily work.

5.3.1 Interview Results

This section is dedicated to the results of the semi-structured interviews. We have split the discussion of the results based on the feature reviewed in the interviews.

Access to Own Issues

All of the participants stated that there are situations in which they would like to access their issues but in which they do not have access to a computer. P1 said the following: "A typical situation is when I meet another developer at the cafeteria and we should discuss an issue in more detail. Often I cannot remember the details of the issue because I left my work station for some time. If I want to discuss the issue, I would have to regain insights on the context of the issue.".

The participants of SBB Informatik reported that they could imagine using the application in stand-up meeting situations in which they either want to check, if the assigned work has been completed, or they could easily retrieve the informations about a specific issue. P1 was irritated by the fact that the application only shows static information. The information he needs the most are how the issues are related, e.g. which issue is blocking other issues he is working on at the moment. The static information is useful for him to recreate the context of an issue but in a discussion situation he would need some kind of a network of how the issues are related.
P2 said that he does not think that this functionality of the application would be used heavily in his team. The combination of needing access to his own issues and not having access to a computer appears infrequently. Even though they have weekly SCRUM meetings, they bring their computers with them and are able to access all the information needed without having the Recon JET glasses.
P3 also thinks that accessing the static information of issues is useful for recreating the context of an issue and gaining insights on the own current work, but he would be in need of more information than this static information. In particular, most of the issues are reported in their issue tracking system using screen shots and displaying them on such a small display is not feasible.

When asked what information is the most important when accessing the own issues, the answers ranged from "the interconnection between issues" and "the issue's status is the most important

information" to "the step by step instructions in order to reproduce the issue". The answers given reflect that the participants have a lot of specialized information which should be displayed in AwareGlass.

Access to Own Calendar

The participants were asked how they access their calendar events when they do not have access to a computer and all of them stated that the smart phone is their primary device used. P1 and P2 stated that the feature of accessing the own calendar events on the glasses is a really practical feature. P2 reported that he has even destroyed a smart phone, when he was walking and accessing his calendar events and then stumbled on the stairs. They reported that they could imagine replacing the smart phone when on the go and needing to access the own calendar events. This is mainly due to the fact that they have their hands free and they are more aware of their current environment. P3 on the other hand stated that he thinks the display is too small to access his calendar events in a simple way. He would rather make use of a smart watch than wear the Recon JET glasses since the display is hardly readable and the hardware has limited controls.

All of the participants stated that in situations where they would use the glasses, they need the information of when and where the meeting takes place and who participates in the meeting. P1 could also imagine integrating informations from their employee database in order to know what employees participate and what roles they have in the company.

Other Self-Awareness Situations

P1 reported that he would wish for an emotion recognition feature built into the application. Since he is a team leader and has to concentrate on the discussions taking place in stand-up meetings, he may oversee certain reactions to statements and as a result conflicts could appear.
P2 as a lead architect is interested in viewing the build, test and overall system status information in order to see live updates of issues that appear with his products.
P3 would like to have a functionality built into the application that displays status information about his developer colleagues. He stated that many times he has to shortly discuss a feature or a change he wants to make to parts of the code that another employee has written. He then has to walk to the other developer's office and ask him if he has the time to discuss this. If the other developer is completely concentrated on his work, he has to repeat this process many times. Also, the other developer is interrupted in his work flow and needs to regain concentration.

Access to Developer's Issues

When the participants were asked if they could imagine using the feature of scanning a developer and viewing his issues in their daily work, all reported that they do not think this is an added value. P1 stated that he would only make use of this feature in situations where he has to supervise one of his developer's work. This could help him to execute managerial reporting of his development team. Otherwise, he does not think that this static information adds any value.
P2 said that he never checks other developers' issues in JIRA, he rather displays the team's (Kanban) board. About five years ago, he was in charge of splitting the work of a project into tasks and assigning these tasks to his developers. In this situation he could have made use of this feature but nowadays they do not work following this pattern.
P3 stated that he could imagine using this feature when having a meeting with the CEO and needing to report on the progress made in the project. But in these situations he either has access to a computer or needs a smart phone in order to use a remote desktop client application to access the

informations.

The participants stated that the due date of an issue and the current status or priority are of interest in situations, where they would access another developer's issues. P1 stated that he could also use the time when he does not have anything to work in order to check the status of his developers' issues and he could forecast problems that could occur with the work on these issues.

Finding Meeting Times

The participants were asked what they do exactly when they need to find a meeting time with a team member. P1 and P3 reported that they use their smart phone to manually find a meeting time which fits all of the meeting participants. P2 said that finding a meeting time with the smart phone is too cumbersome and he would rather do this when he arrives back at his work station.

All of the participants stated that they would happily use this feature since it is easy to use and does not need any manual work. P1 and P2 said that the meeting room informations could also be integrated in this functionality in order to find one which is free at the respective meeting time. They also stated that it should be possible to directly enter the meeting in the calendar, even though they are aware of the limited controls. P1 made the suggestion to implement voice commands in order to facilitate entering a meeting event with the glasses.
P3 said that besides the situations where he meets a developer spontaneously, he could also imagine using this functionality in meetings to find a follow-up appointment with the meeting participants. P3 cannot imagine entering appointments directly since the hardware controls are rather limited.

The most important information in these situations, as stated by the participants, are the free meeting time slots as well as the meeting room information, i.e. which room is available at what time. Also, the time needed to travel to the meeting location could be of interest for the interviewees.

Further Situations

When asked about further situations in which they could imagine using the `AwareGlass` application, P1 stated that there are situations, in which a meeting participant does not show up at the time arranged and also sometimes he cannot find a team member in the open-plan office. In these situations, he could imagine viewing the location of his colleagues in order to find them and also to know whether to start with the meeting or wait for the missing participant.

5.3.2 Quantitative Results

We also conducted a short acceptance test with the interview participants in order to assess the quality and usability of the application. The questions of the acceptance test can be found in Appendix A and Figure 5.1 summarizes the results of our findings.

The participants rated the application as being user friendly and easy to use, however they stated that the hardware controls are rather difficult to use. The QR code scanning functionality was rated as being usable on a daily basis. The participants were not completely satisfied with the information, the `AwareGlass` application displays and they wish for more thorough information, especially concerning the issue lists. The participants agree that the application could be used on a daily basis and support their daily work, but here also the hardware deteriorated the results. P1

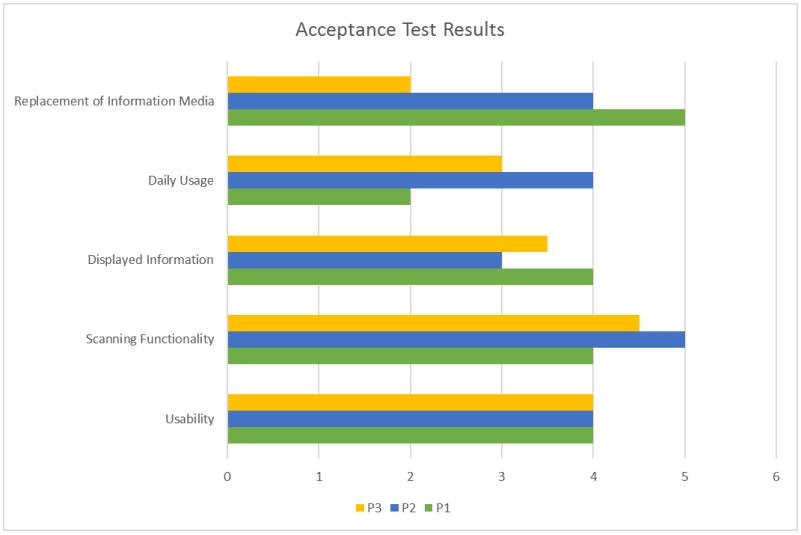

Figure 5.1: Graph of the acceptance test results

also said that he would have to use contact lenses in order to be able to use the Recon JET glasses. The participants cannot imagine that these glasses in combination with `AwareGlass` are able to fully replace the used information media but they rather see the application as an additional support medium, especially when considering the limited hardware. However in the case of finding meeting times, all of the participants could imagine replacing their smart phones or computers with the `AwareGlass` application.

5.4 Conclusion

The preliminary results of our evaluation indicate that the `AwareGlass` application is able to support certain situations in order to foster awareness in development teams. Our evaluation implies that stand-up meetings can be supported by displaying informations about the current status of issues and their recent changes. However, the use cases in which the application is useful are limited by the hardware and the controls.
The most useful functionality of the application is quickly finding meetings with team members and the participants could even imagine replacing their current information media in these situations.
The acceptance test results show that the application is perceived to be user friendly and to be able to support daily work activities, if further changes are made to the application and more features are implemented. In chapter 6 we discuss possible ideas to improve the application and to make it more usable in supportable situations.

5.5 Threats to Validity

5.5.1 Hardware Limitations

The Recon JET glasses may heavily influence the results of our application's evaluation. Since the glasses are primarily made for sports applications, the hardware needs to fulfill the needs of this area. They need to be unobtrusive and robust under different circumstances. During the evaluation, all of the participants said that the glasses' display is hard to read and remarked that the controls of the glasses are not as responsive as they would expect. With this in mind, the evaluation of our application may be impacted negatively because of the hardware limitations of the Recon JET glasses.

5.5.2 Networking Limitations

Another threat to validity of our evaluation are the networking capabilities of the Recon JET glasses. The `AwareGlass` application needs a wireless network connection in order to function in a proper way. We have used a smart phone as a wireless hot spot while evaluating the application. Since the evaluation took place in meeting rooms, the network may have been slower and have negatively affected the responsiveness of the application as a whole.

5.5.3 Light Conditions

The camera functionality of the Recon JET glasses is rather suboptimal, thus the QR code scanning functionality of our application may be strongly influenced by the light conditions of the room in which the evaluations took place. Furthermore, the QR code was scanned from a notebook monitor, which may reflect light and therefore impede the scanning functionality.

Chapter 6

Future Work

In this chapter we discuss possible future implementations in order to improve `AwareGlass`. We consider the ideas which appeared in the evaluation of our thesis (see Chapter 5) as well as our own experiences.

6.1 Issue Network

In the evaluation of this thesis, P1 stated that he would like to have access to a network view, showing the interdependencies of the `Bugzilla` issues. This issue network could be implemented with an additional field in the issue detail view, displaying the issues which are dependent from this issue and those on which the current issue depends. The user could then select one of the dependent issues and switch to this issue's details with a single button click. Furthermore, an algorithm to sort the issues by their most important issues, i.e. those that block the most issues, could be implemented. Since the Recon JET display is small, we don't think it is feasible to implement a network diagram view, showing the issues and their dependencies.

6.2 Appointment Creation

P1 and P2 reported that the creation of appointments in the "Find Meeting Times" view would improve the application. Since the Recon JET glasses have limited hardware controls, we propose to implement a voice recognition functionality to create an appointment directly with the glasses. `Recon OS` does not include `Google Play` services nor the `Google Play` store, thus implementing a voice recognition may be cumbersome and needs to rely on third-party frameworks.
Another possibility to implement the creation of appointments with the glasses without having to implement voice recognition is to include appointment suggestion algorithms in `AwareGlass`. The user could select the meeting time in the "Find Meeting Time" view and the application could then lead the user through the creation process by suggesting the title, the meeting time (based on previous entries, especially in connection with the scanned developer) as well as the location of the meeting based on a history of previous entries. This approach would require the implementation of machine learning algorithms to improve the suggestions made by the application.

6.3 Calendar Event Informations

The calendar event view rows include informations about the meeting time and date as well as the title of the meeting. To further improve `AwareGlass`, it should include information about the meeting location, the meeting room and, as proposed by P3, information whether a preparation is necessary and how this preparation needs to be done. This can be done by implementing an additional "Event Detail" view which displays this information similar to the "Issue Detail" view.

6.4 Emotion Recognition

In the interview, P1 proposed an interesting functionality: the implementation of an emotion recognition to support team leaders in stand-up meetings. He stated that most of the time he is concentrated on the information expressed by his team and it may occur that he does not recognize certain developers' reactions to the statements made. By implementing an emotion recognition functionality, team leaders can be supported in the awareness of participants' emotions during stand-up meetings. P1 stated that this would prevent dissatisfactions in the decisions he makes and improve the satisfaction of team members.

6.5 Build, Test and System Status

As a software architect, P2 stated that he could imagine using the Recon JET glasses to monitor build, test and system statuses. The implementation of real-time information about system statuses can improve the awareness of software engineers and architects about their current work and if there occurred problems with their code. Also, seeing real-time information about the system health can improve the reaction time to solve problems with their system. We propose to implement a view containing the same analogy we used for the issue view, i.e. traffic light colors, to display build, test and system status information. Furthermore, when there occurs a problem, the application could display a pop-up and notify the user with an audio notification to check the statuses. Also, the implementation of error details when selecting a failed build or test needs to be implemented to support the developer in solving the error.

6.6 Developer Status Information

P3 proposed to implement status information of his team members. He often finds himself in the situation that he needs to shortly discuss the code written by another developer and when the team member is fully concentrated on his work, his flow is interrupted and P3 needs to ask at a later time. Thus, the implementation of status information about a developer could improve efficiency and awareness of other team members' current work status. One could implement status informations which the team members can change manually, i.e. they would have to post their current work status themselves. We propose to take this idea one step further and use the work station's current informations to automatically display status information. We could use the informations about the open applications as well as the software engineer's mouse movements and keyboard inputs to analyze the behavior and process this information to gain insights about the developer's work flow.

6.7 Meeting Participants' Locations

P1 stated that many meetings are delayed because one or more participants do not appear to the meeting at the agreed time. This has many reasons, one of them being that the meeting participant does not find the specified meeting room, another one that the participant cannot be there on time. P1 proposed to implement a view which displays all the participants' locations such that the meeting organizer is able to evaluate, whether they should wait for the remaining participants to appear or whether they may start with the meeting. Furthermore, when the meeting organizer sees that a participant cannot find the meeting room, one of the participants may find the missing participant and show him the meeting room location.
One of the possibilities to implement such a functionality is to display a map of the office building and the participants' locations on it. Since the Recon JET glasses have a small display, it is unclear if this is a feasible solution. Another possibility is to display the user's distance to the other participants in a list, which may at least support the user to decide whether to begin with the meeting or to wait. Furthermore, a navigation arrow pointing towards the selected missing participant may help the user to find him, when the participant cannot find the room.

Chapter 7

Conclusion

In this thesis we presented `AwareGlass`, a prototype application with the goal to foster awareness within development teams. The application is implemented for the Recon JET augmented reality glasses and consists of features designed to support more spontaneous awareness within development teams. With `AwareGlass`, the user is able to view his own issues as well as his own calendar events. When the user scans another developer's QR code, he can access this developer's issues and find a meeting time based on the developer's and his own calendar events. The application served as a starting point for our thesis evaluation.

To evaluate our thesis, we conducted semi-structured interviews with three experienced software engineers. One of them works at SWIT Solutions AG, a rather small company with about 20 employees, the other two work at SBB Informatik AG, a larger company with about 1000 employees. Furthermore, we conducted an acceptance test to assess the quality and usability of the application. While the hardware deteriorated the results of this test, the potential to use this application on a daily basis – if further improvements are made – has been rated high by the participants.

In what situations can developers profit from displaying awareness information through smart glasses? The evaluation of our thesis indicates that indeed, we can support awareness in situations without a computer. Especially stand-up meetings can be supported in an unobtrusive way by displaying the issues of each developer taking part in it. Even though `AwareGlass` may not fully replace a mobile phone or a computer, the participants would prefer using the glasses in certain situations. One of them is the situation in which a developer is walking in the hallway, looking at the upcoming meetings. With a smart phone, the developer is not able to be aware of his environment and may even stumble or walk into other people. In this situation, the glasses allow the developer to see his upcoming calendar events without interfering with the user's awareness of the environment.
Furthermore, in situations in which a developer wants to find a possible meeting time with one of his colleagues, the participants could even imagine replacing their smart phone functionality with the Recon JET glasses, since the feature of automatically finding possible meeting times does not exist for smart phones as of now.

What information artifacts would developers seek on smart glasses? In stand-up meeting situations, the most important information – as stated by the participants – is the interconnection between issues as well as the status of the issue. With this information, the user can see at a glance which issues are the most critical, i.e. which issues are blocking others from being solved. Also, the status of an issue allows the user to discuss it with the assigned developer in these stand-ups. When accessing calendar event information, the participants stated that along with the title and

the meeting time, the location is an important information. The participants reported that meeting room informations such as the room availability and the equipment in the room should be integrated in the application.

In our exploratory study, we learned more about the needs of the interviewed developers and how we can improve our application. Besides the aforementioned issue network view, there are several improvements which can be made in a future work. One of the features the participants wish for is the creation of appointments directly within `AwareGlass`, possibly through voice commands. Also, viewing the build, test and overall system status is desired in order to improve the awareness of the current project on which the user is working. By implementing a developer status information view, the user can see if the developer with whom he wants to talk about an issue is currently busy. This improves work efficiency, since the user does not have to repeatedly ask if the developer is concentrated and if he has the time to discuss the issue. Also, this feature improves the team awareness, i.e. which team members are currently in the office and also if they are not to be disturbed at the moment.

Our evaluation results indicate that the information needs in situations without a computer differ from those in situations where the access to a computer is possible. The participants stated that they do not necessarily need access to a team member's issues, if they are not attending a stand-up meeting, but rather they want the Recon JET glasses to provide self-awareness information such as the user's calendar. Also, the participants stated that the most useful functionality of `AwareGlass` is finding possible meeting times, which indicates that the application is perceived to be more of a support medium to improve efficiency and awareness in these situations.

From the results of our evaluation, we may propose to further improve the application based on the strengths of augmented reality glasses: displaying information in an unobtrusive way in situations in which the user is moving and cannot access a computer or other information medium, such as a smart phone, without interfering with his awareness of the environment. Furthermore, the usage of the built-in hardware sensors in future implementations of `AwareGlass` may improve the application's ability to provide useful informations, based on the user's current situation and environment. Since the Recon JET glasses' hardware was rated rather low, eventually the use of more capable hardware could improve the user experience with the application.

Appendices

Appendix A

Questionnaires, Transcripts

A.1 Information About Participants

	Participant 1	Participant 2	Participant 3
Years of Experience as a Developer	15 years	16 years	25 years
Job Title	Team Leader Software Engineering / Lead Software Architect	Senior Architect ICT	Lead Software Developer / Lead Software Architect
IDEs with Most Experience	IntelliJ, Eclipse	IntelliJ, Eclipse	Borland C++ Builder, Microsoft Visual Studio
Years of Experience with Issue Tracking Systems	8 years	11 years	15 - 20 years
Usage of Issue Tracking Systems at Work	Daily	Daily	Less than twice a week
Self-rated Programming Abilities	Above-average	Above-average	Above-average
Contribution to Open Source Projects	Yes, SBB Javascript Web-Stack	No	No

Table A.1: The Information about the participants obtained from a Pre-Questionnaire

A.2 Questionnaires

A.2.1 Semi-structured Questionnaire

1. Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, aber Zugriff auf Ihre aktuell zu bearbeitenden Issues benötigen?

   a) Welche?

   b) Wie gehen Sie in solchen Situationen vor?

2. AwareGlass erlaubt es Ihnen, jederzeit auf Ihre aktuellen Bugzilla / Issue Tracking Informationen zuzugreifen.
   a) Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?
   b) Welche Informationen würden Sie in diesen Situationen am meisten benötigen?
3. Wie greifen Sie auf Ihren eigenen Kalender zu, wenn Sie keinen Zugriff auf einen Computer haben?
4. AwareGlass erlaubt es Ihnen, auf Ihren eigenen Kalender bzw. dessen Events zuzugreifen.
   a) In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?
   b) Welche Informationen würden Sie in diesen Situationen am meisten benötigen?
   c) Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?
5. Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über Ihre Arbeit zu erlangen?
   a) Welche?
   b) Welche Informationen wären in diesen Situationen für Sie am wichtigsten?
6. Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, und Zugriff auf Informationen der Entwickler Ihres Teams benötigen würden?
   a) Wenn ja, welche?
   b) Wie gehen Sie in solchen Situationen vor?
7. Mit der AwareGlass-Applikation ist es möglich, ihre Teamkollegen einzuscannen (beispielsweise an einem Meeting) und direkt auf die aktuellen Bugzilla Issues Ihres Teamkollegen zuzugreifen.
   a) Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?
   b) Welche Informationen würden Ihnen im Zusammenhang mit dieser Funktionalität am meisten weiterhelfen?
8. Wie gehen Sie in Situationen, in denen Sie keinen Zugriff auf einen Computer haben, vor, wenn Sie einen Termin mit einem Teamkollegen vereinbaren möchten?
9. Mit AwareGlass können Sie verfügbare (freie) Termine auf der Basis Ihres eigenen Kalenders sowie dem des gescannten Entwicklers anzeigen.
   a) In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?
   b) Welche Informationen würden Sie in diesen Situationen am meisten benötigen?
   c) Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?
10. Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über die Arbeit Ihrer Teamkollegen zu erlangen?
    a) Welche?
    b) Welche Informationen wären in diesen Situationen für Sie am wichtigsten?

A.2.2 Acceptance Test Questions

1. Die Applikation ist benutzerfreundlich / einfach zu benutzen
    1. Trifft gar nicht zu
    2. Trifft eher nicht zu
    3. Neutral
    4. Trifft eher zu
    5. Trifft voll und ganz zu
2. Die Scanning-Funktionalität ist schnell genug für einen täglichen Einsatz
    1. Trifft gar nicht zu
    2. Trifft eher nicht zu
    3. Neutral
    4. Trifft eher zu
    5. Trifft voll und ganz zu
3. Die Informationen, die die Applikation bereitstellt, könnten mich in meiner täglichen Arbeit unterstützen
    1. Trifft gar nicht zu
    2. Trifft eher nicht zu
    3. Neutral
    4. Trifft eher zu
    5. Trifft voll und ganz zu
4. Ich könnte mir vorstellen, die Applikation in meiner täglichen Arbeit einzusetzen
    1. Trifft gar nicht zu
    2. Trifft eher nicht zu
    3. Neutral
    4. Trifft eher zu
    5. Trifft voll und ganz zu
5. Ich könnte mir vorstellen, dass die Applikation andere Informationsmedien, die ich täglich einsetze, ersetzen könnte
    1. Trifft gar nicht zu
    2. Trifft eher nicht zu
    3. Neutral
    4. Trifft eher zu
    5. Trifft voll und ganz zu

A.3 Transcripts

A.3.1 Participant 1

1. **Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, aber Zugriff auf Ihre aktuell zu bearbeitenden Issues benötigen?**

   a) **Welche?**

   b) **Wie gehen Sie in solchen Situationen vor?**

   Ja die gibt es definitiv. Eine klassische Situation ist, wenn man sich in der Cafeteria trifft und ein Problem diskutieren sollte, aber man kennt die Details nicht mehr genau, da man bereits eine Viertelstunde bis Stunde vom Arbeitsgerät weg ist. Man möchte dann dies in 5 Minuten kurz klären, hat aber den Kontext nicht mehr.

   Ich versuche einen Termin zu vereinbaren oder kurz an einen Arbeitsplatz zu sitzen und mich einloggen um die Details nochmals anzuschauen.

2. **AwareGlass erlaubt es Ihnen, jederzeit auf Ihre aktuellen Bugzilla / Issue Tracking Informationen zuzugreifen.**

   a) **Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   Es ist spannend, ja es würde sicher helfen mir die Informationen zu beschaffen, die ich momentan gerade nicht hätte.

   Was mich irritiert ist, dass ich rein die statische Information des Issues sehe. Ich sehe nicht, wie dieser Issue mit anderen vernetzt ist oder wo der Schwerpunkt in Bezug auf einen Mitarbeiter, den ich gerade getroffen habe, ist. Es wäre besser, wenn die Informationen verknüpft wären und ich zusätzliche Informationen zu meinem Gegenüber abrufen könnte, zum Beispiel ob ich bereits mit dieser Person zusammengearbeitet habe und wo es Berührungspunkte gibt. Und auch die Vernetzung zwischen den eigenen Issues würde mir mehr bringen, als die statische Information zu einem Issue. Die statische Information ist nützlich, um meinen Kontext wieder aufzubauen, eventuell die letzten Kommentare noch einmal anzuschauen oder zu sehen ob sich in der Zwischenzeit jemand anderes an dem Issue beteiligt hat. Aber die Verknüpfung zu anderen Issues wäre sehr gut.

   Die Sortierung ist meines Erachtens eher redundant, wenn ich die Vernetzung der Issues hätte. Eine Filterung z.Bsp. auf Projektbasis würde mir mehr Nutzen bringen.

3. **Wie greifen Sie auf Ihren eigenen Kalender zu, wenn Sie keinen Zugriff auf einen Computer haben?**

   Ich greife hauptsächlich über mein Smartphone auf den Kalender zu.

4. **AwareGlass erlaubt es Ihnen, auf Ihren eigenen Kalender bzw. dessen Events zuzugreifen.**

   a) **In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

c) **Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?**

Mit dem Smartphone habe ich eine integrierte Lösung, ich kann Skype for Business Anrufe annehmen und ich habe ein Audio-Device, mit welchem ich direkt mit anderen kommunizieren könnte. Wenn ich das Smartphone nicht dabei hätte, wäre es natürlich sehr spannend die Brille zu verwenden. Aber es ist tendenziell sehr unwahrscheinlich, dass ich das Smartphone nicht dabei habe. Wenn die Brille die Smartphone-Funktionalität mit Voice etc. übernehmen könnte und es einfacher zu verwenden wäre als das Smartphone, dann wäre ich auch bereit umzusteigen. Das Smartphone wäre aber meine Präferenz.

Wenn ich unterwegs wäre, würde ich definitiv die Brille vorziehen. Ich wäre dadurch viel weniger gestört und könnte mein Umfeld besser wahrnehmen. Und es würde mir helfen, mich zu koordinieren und den Weg zu finden. Der Ort, der Weg dorthin, welche Teilnehmer, ev. die Mitarbeiterinformationen bzw. Rolle der Teilnehmer eines Meetings wären wichtig für mich.

5. **Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über Ihre Arbeit zu erlangen?**

   a) **Welche?**

   b) **Welche Informationen wären in diesen Situationen für Sie am wichtigsten?**

   In Stand-Up Meetings wäre eine Emotionserkennung und deren Anzeige sehr interessant. Beispielsweise, wenn ein Konflikt aufgetreten ist, den ich nicht bemerkt habe. Dies würde mir helfen, das Zwischenmenschliche besser handzuhaben.

6. **Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, und Zugriff auf Informationen der Entwickler Ihres Teams benötigen würden?**

   a) **Wenn ja, welche?**

   b) **Wie gehen Sie in solchen Situationen vor?**

   Wir haben eine starke Tendenz zu Papierboards bei der SBB seit kurzer Zeit, vor allem bei Stand-up Meetings. Auf statische Informationen des Issue Tracking Systems zuzugreifen bringt mir aber nicht wirklich viel. Eher z.Bsp. Burn-Down Charts wären spannend oder weitere Informationen, z.Bsp. wenn mir ein Entwickler sagt, dass er einen Issue fertig bearbeitet hat, müsste ich die Code Qualität überprüfen können, d.h. hat er die Definition of Done eingehalten, hat er die Richtlinien eingehalten. Solche Informationen wären für mich spannend, also kurz gesagt das "Manager-Reporting". Dies würde mir erlauben, einen Entwickler besser zu kontrollieren.

7. **Mit der AwareGlass-Applikation ist es möglich, ihre Teamkollegen einzuscannen (beispielsweise an einem Meeting) und direkt auf die aktuellen Bugzilla Issues Ihres Teamkollegen zuzugreifen.**

   a) **Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?**

   b) **Welche Informationen würden Ihnen im Zusammenhang mit dieser Funktionalität am meisten weiterhelfen?**

   Controlling ist grundsätzlich immer schlecht, denn ich habe ein Team, dem ich vertrauen schenke und möchte auch den Teammitgliedern vertrauen. Wenn ich z.Bsp. einen Entwickler auf dem Weg im Gang antreffen würde und seinen Status haben möchte, also wo er

steht und welche Probleme es gibt. Auch wenn ich Idle-Time habe, dann könnte ich kurz schauen, wo ein Teammitglied gerade drinsteckt und welche Probleme es geben könnte. Diese Idle-Time könnte ich effizienter nutzen und spontaner auf Informationen zugreifen um meine Effizienz zu steigern.

8. **Wie gehen Sie in Situationen, in denen Sie keinen Zugriff auf einen Computer haben, vor, wenn Sie einen Termin mit einem Teamkollegen vereinbaren möchten?**

   Wenn ich mein Smartphone dabei habe, gleiche ich mit dem Gegenüber manuell die Kalender ab und das ist etwas, was mich extrem nervt. Wenn ich kein Smartphone dabei hätte, würde ich das Teammitglied auf später vertrösten und ihm sagen, dass ich ihm eine Einladung zusenden werde. Oft gerät dies aber auch wieder in Vergessenheit.

9. **Mit AwareGlass können Sie verfügbare (freie) Termine auf der Basis Ihres eigenen Kalenders sowie dem des gescannten Entwicklers anzeigen.**

   a) **In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   c) **Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?**

   Ich würde diese Funktionalität gerne verwenden, aber unter der Bedingung, dass das System korrekte Informationen anzeigt. Da wir mehrere Standorte für SBB-Büros haben, möchte ich diese Information vom System berücksichtigt haben. Auch sollten die freien Meetingräume miteinbezogen werden. Dies ist ein Feature, welches ich stark vermisse und ich würde dies gerne in meiner täglichen Arbeit einsetzen. Das Miteinbeziehen der Wegzeit zwischen den Standorten wäre aber sehr wichtig.

   Auch das direkte Erfassen von Meetings wäre eine gute Funktionalität. Da das Gerät von den Controls her eher limitiert ist, müsste eine Voice Recognition oder ähnliches implementiert werden, um die Handhabung zu vereinfachen.

10. **Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über die Arbeit Ihrer Teamkollegen zu erlangen?**

    a) **Welche?**

    b) **Welche Informationen wären in diesen Situationen für Sie am wichtigsten?**

    Manchmal gibt es Situationen, in denen ich unterwegs bin und finde das gesuchte Teammitglied nicht, denn wir haben Grossraumbüros. Oder auch wenn jemand nicht zu einem Meeting erscheint und ich nicht weiss, wo er ist. Dies wäre eine Situation, die die Brille sehr gut unterstützen könnte. Beispielsweise könnte mir die Brille den Standort des Teilnehmers anzeigen, damit ich ihn abholen könnte. Dies würde auch meine anderen Kommunikationskanäle wie Anrufe, WhatsApp-Nachrichten etc. ersetzen können.

A.3.2 Participant 2

1. **Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, aber Zugriff auf Ihre aktuell zu bearbeitenden Issues benötigen?**

   a) **Welche?**

   b) **Wie gehen Sie in solchen Situationen vor?**

   Ja und nein. Sehr oft arbeiten wir in Teams und es gibt wenige Issues, die sich auf eine bestimmte Person beziehen, sondern eher auf das Team. Das Team hat sehr oft ein Board, sei es auf einem Bildschirm oder effektiv in Papierform. Wenn man über dieses Board spricht, hat man sehr oft keinen Zugriff auf den eigenen Computer. Dort finden typischerweise täglich Stand-Up Meetings statt. Das wäre der wichtigste Zeitpunkt, bei dem man Zugriff auf seine Issues haben sollte. Bei der SBB ist es üblich, dass diese Boards zusätzlich in Jira erfasst sind. In der Entwicklung läuft eigentlich alles über Jira, seien es Bugs oder Boards etc.

   Dass man sich im Gang trifft und dann effektiv den Zugriff auf seine Issues braucht, ist eher weniger der Fall. Da man meistens nur ein bis zwei Issues bearbeitet, sollte man das grundsätzlich im Kopf haben. Ich denke diese Funktionalität würde weniger die Entwickler interessieren, sondern eher die Personen, die keinen unmittelbaren Einblick in die Entwicklung haben. Die Kombination, dass man Zugriff auf seine Issues haben muss und zusätzlich nicht an einem Computer sitzt, tritt sehr wenig auf. Diese Situation trifft meist in den vorhin genannten Stand-up Meetings auf. Ich könnte mir auch vorstellen, dass jede Person eine Brille trägt und die Brille dadurch das Board virtualisieren und ersetzen könnte. In einer Situation, in der ein Entwickler alleine die Wartung eines Produkts übernimmt, wäre diese Funktionalität sicher viel hilfreicher, da man sich mehr mit einzelnen Personen von ausserhalb des Projekts besprechen muss.

2. **AwareGlass erlaubt es Ihnen, jederzeit auf Ihre aktuellen Bugzilla / Issue Tracking Informationen zuzugreifen.**

   a) **Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   Ich denke, es gibt einzelne Situationen in denen man diese Applikation verwenden könnte, aber ich zweifle daran, dass sich diese Applikation wirklich durchsetzen würde.

   Das Wichtigste ist der aktuelle Status, sei es nur um den Abgleich mit dem Papier-Board korrekt zu machen. Manchmal muss man auch auf die Details eines Issues zugreifen, aber dafür, denke ich, ist die Brille das falsche Instrument. Wenn man wirklich wissen möchte, was die genauen Akzeptanzkriterien sind oder ob jemand noch einen zusätzlichen Kommentar hinzugefügt hat, würde ich eher auf den PC ausweichen. Das Display ist sehr klein und daher würde ich eher auf den PC zugreifen, da es hardwaremässig sehr unkomfortabel ist, längere Texte durchzuscrollen. Wenn ich diese Brille so sehe, würde ich weniger auf die Issues zugreifen wollen, als eher auf den Buildstatus, Codequalität oder ähnliche Dinge, die ich permanent im Auge behalten möchte. Beispielsweise auch den allgemeinen Überblick des Boards.

3. **Wie greifen Sie auf Ihren eigenen Kalender zu, wenn Sie keinen Zugriff auf einen Computer haben?**

   Mit dem Smartphone.

4. **AwareGlass erlaubt es Ihnen, auf Ihren eigenen Kalender bzw. dessen Events zuzugreifen.**

   a) **In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   c) **Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?**

   Ich finde dieses Feature sehr praktisch. Es ist vielleicht für die Entwickler fast am unwichtigsten von allen, da sie täglich an ihrem Platz im Team arbeiten und eher spontane Meetings haben. Aber auch bei Entwicklern wäre es nicht uninteressant, da diese wahrscheinlich am ehesten Mails und Meetings übersehen, wenn sie konzentriert an etwas arbeiten. Für mich fehlt jedoch die wichtigste Information, nämlich der Ort. Es ist sehr wichtig zu sehen, wo muss ich beim nächsten Meeting genau hingehen. Ich habe schon einmal ein Handy zerstört, da ich unterwegs auf meinem Handy den Ort des Meetings nachschauen wollte und gestolpert bin und es ist ein gängiges Szenario, dass man durch die Gänge des Grossraumbüros geht und auf dem Smartphone nachschaut, wo man genau hin muss. In solchen Situationen ist so eine Funktionalität mit der Brille sehr angenehm, da man das nähere Umfeld im Blick haben kann.

   Vor allem benötige ich den Ort, die Zeit und den Titel des Meetings. Meiner Meinung nach, würden die Kalender-Events des heutigen Tags reichen, um mir einen Überblick zu verschaffen.

   Ich denke nicht, dass die Brille das Smartphone komplett ersetzen kann, da das Smartphone Skype for Business integriert hat. Wenn die Brille dies auch könnte und ansonsten komfortabler zu bedienen wäre, könnte es das Smartphone eventuell ersetzen.

5. **Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über Ihre Arbeit zu erlangen?**

   a) **Welche?**

   b) **Welche Informationen wären in diesen Situationen für Sie am wichtigsten?**

   Build- und Test-Status zu sehen wäre eine wichtige Funktion. Wenn man auch SMS und verpasste Anrufe sehen würde, dann wäre das auch sehr gut. Eine Integration von Skype for Business wäre für Entwickler sehr interessant. Die Kombination von erreichbar sein und sich auf den Code zu konzentrieren ist sehr selten und man übersieht oft, ob man eine Nachricht oder einen Anruf in Skype for Business erhalten hat. Man müsste dieses Feature aber auch ausschalten können, wenn man sich stark konzentrieren möchte.

6. **Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, und Zugriff auf Informationen der Entwickler Ihres Teams benötigen würden?**

   a) **Wenn ja, welche?**

   b) **Wie gehen Sie in solchen Situationen vor?**

   Eigentlich habe ich diese Situation bis jetzt fast noch nie angetroffen. Auch im Bugtracker suche ich nicht nach anderen Entwicklern. Ich greife eher auf meine eigenen Issues zu oder auf das Board des Teams. Als wir noch nach anderen Prozessen gearbeitet haben, musste ich oft die einzelnen Tasks definieren und diese meinen Entwicklern zuweisen. In diesen Situationen hätte ich es eher gebraucht, aber diese Prozesse führen wir nicht mehr durch.

Die Synchronisation mit dem Entwicklungsteam findet in den täglichen Stand-Up Meetings statt und dies reicht mir aus. Zudem sind diese Informationen sehr volatil und ändern sich täglich, von daher ist es eher unwichtiger.

7. **Mit der AwareGlass-Applikation ist es möglich, ihre Teamkollegen einzuscannen (beispielsweise an einem Meeting) und direkt auf die aktuellen Bugzilla Issues Ihres Teamkollegen zuzugreifen.**

   a) **Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?**

   b) **Welche Informationen würden Ihnen im Zusammenhang mit dieser Funktionalität am meisten weiterhelfen?**

   In einem nicht-agilen Prozess wäre mir wichtig, das Due Date zu wissen. In agilen Prozessen wie beispielsweise in einem SCRUM-Prozess wäre diese Information irrelevant, da alle aktuellen Issues zum aktuellen Sprint dazu gehören. Der Titel ist natürlich sehr wichtig. Je nach Prozess (Kanban, Bugfixing) wäre noch relevant, ob es ein Blocker ist, d.h. die Priorität sowie auch der Ersteller des Issues.

8. **Wie gehen Sie in Situationen, in denen Sie keinen Zugriff auf einen Computer haben, vor, wenn Sie einen Termin mit einem Teamkollegen vereinbaren möchten?**

   Mit dem Smartphone ist es zu unpraktisch und zu unübersichtlich. Meist mache ich das erst am Arbeitsplatz und sende dem Teammitglieder eine Einladung. In Outlook lasse ich die unterschiedlichen Kalender einblenden und versuche manuell einen Termin zu finden, der frei ist.

9. **Mit AwareGlass können Sie verfügbare (freie) Termine auf der Basis Ihres eigenen Kalenders sowie dem des gescannten Entwicklers anzeigen.**

   a) **In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   c) **Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?**

   Diese Funktionalität ist sehr praktisch und könnte sehr oft benutzt werden. Ich würde dieses Feature fast täglich verwenden. Beispielsweise in der Situation, in der man einen Entwickler im Gang antrifft, gibt es oft die Situation, dass man einen Meetingtermin vereinbaren möchte. Dieses Feature würde sicher oft benutzt werden.

   Für mich wäre vor allem interessant, wann ein Terminslot frei wäre. Was auch sehr wichtig ist, sind die Rauminformationen, d.h. wann welcher Raum frei ist und wo sich dieser befindet. Ich denke diese Situationen können sehr gut unterstützt werden, aber ich müsste auch direkt auf der Brille einen Termin eintragen können, allenfalls wird aber durch die beschränkte Hardware die Bedienung beeinträchtigt.

10. **Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über die Arbeit Ihrer Teamkollegen zu erlangen?**

    a) **Welche?**

    b) **Welche Informationen wären in diesen Situationen für Sie am wichtigsten?**

    Keine.

A.3.3 Participant 3

1. **Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, aber Zugriff auf Ihre aktuell zu bearbeitenden Issues benötigen?**

   a) **Welche?**

   b) **Wie gehen Sie in solchen Situationen vor?**

   Wenn beispielsweise die Netzwerkverbindung des PCs nicht funktioniert. Gerade im Bereich Home-Office gibt es Situationen, in denen ich nicht auf das Issue Tracking System zugreifen kann, da die Netzwerkverbindung nicht aufgebaut werden kann.

   Eventuell über eine Remote Desktop Client App könnte ich auf den Bugtracking-Server zugreifen. Es wäre jedoch zwingend eine Netzwerkverbindung auf das Firmennetz notwendig.

2. **AwareGlass erlaubt es Ihnen, jederzeit auf Ihre aktuellen Bugzilla / Issue Tracking Informationen zuzugreifen.**

   a) **Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   Ja, ich denke die textuellen Informationen, die ich benötigen würde, d.h. statische Informationen des Issues, könnte ich in solchen Situationen gut verwenden. Der Fehlerkontext ist sehr wichtig, um den Fehler ausfindig zu machen und diesen auch aufzulösen. Darauf müsste ich in der Applikation Zugriff haben. Was ich mir jedoch nicht vorstellen könnte, wäre das Anzeigen von Screenshots auf dem kleinen Bildschirm. In Fällen, in denen ich Zugriff auf die Datenbank des Kunden haben müsste oder Screenshots anzeigen müsste, würde mir die Applikation nicht weiterhelfen können.

   Für mich die wichtigsten Informationen wären die einzelnen Schritte, d.h. das schrittweise Vorgehen, um den Fehler zu reproduzieren. Die Bedingungen / der Kontext, der den Fehler provoziert müsste abrufbar sein.

3. **Wie greifen Sie auf Ihren eigenen Kalender zu, wenn Sie keinen Zugriff auf einen Computer haben?**

   Mit dem Smartphone.

4. **AwareGlass erlaubt es Ihnen, auf Ihren eigenen Kalender bzw. dessen Events zuzugreifen.**

   a) **In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   c) **Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?**

   Ich sehe den Vorteil der Brille in diesen Situationen nicht wirklich. Der Bildschirm ist sehr klein und man muss die Brille tragen. Ich würde eher eine Smartwatch tragen als diese Brille zu verwenden.

Für mich sind vor allem die Zeit und die Örtlichkeit wichtig. Auch die Wegzeit zum Meeting wäre interessant. Eine weitere wichtige Information ist, ob der Termin Vorbereitungsmassnahmen benötigt und allenfalls welche Termine als "Vorbereitungstermine" für den aktuellen Termin einzutragen sind.

Ich denke nicht, dass die Brille die Smartphone- oder PC-Funktionalität ersetzen könnte. Die Darstellungsmöglichkeiten sind sehr limitiert und auch die Bedienung ist eher mühsam.

5. **Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über Ihre Arbeit zu erlangen?**

   a) **Welche?**

   b) **Welche Informationen wären in diesen Situationen für Sie am wichtigsten?**

   Eine Status-Information der anderen Entwickler wäre noch interessant zu sehen, beispielsweise, ob der andere Entwickler im Büro ist und woran er gerade arbeitet. Vor allem in Fällen, in denen der Entwickler an Modulen arbeitet, die mich auch betreffen.

6. **Gibt es Situationen, in denen Sie keinen Zugriff auf einen Computer haben, und Zugriff auf Informationen der Entwickler Ihres Teams benötigen würden?**

   a) **Wenn ja, welche?**

   b) **Wie gehen Sie in solchen Situationen vor?**

   Da wir ein eher kleines Entwicklerteam sind, gibt es in dem Sinne keine Spezialisten, denen die Issues zugewiesen werden. Die Issues werden eher im Team bearbeitet. In Meetings mit Nicht-Entwicklern, müsste ich Zugriff auf die Issues eines anderen Entwicklers haben, um den aktuellen Status des Issues abzufragen und dies mit dem CEO zu besprechen.

   Ich würde über das Smartphone über einen Remote Desktop Client auf den Issue Tracker zugreifen. Anderenfalls würde ich es auf später verschieben und dem Fragenden per E-Mail ein Feedback zusenden.

7. **Mit der AwareGlass-Applikation ist es möglich, ihre Teamkollegen einzuscannen (beispielsweise an einem Meeting) und direkt auf die aktuellen Bugzilla Issues Ihres Teamkollegen zuzugreifen.**

   a) **Denken Sie, diese Funktionalität könnte Sie in den vorher genannten Situationen unterstützen?**

   b) **Welche Informationen würden Ihnen im Zusammenhang mit dieser Funktionalität am meisten weiterhelfen?**

   Ja, die Applikation könnte mich in solchen Situationen unterstützen. Vor allem um die Deadlines und den aktuellen Status eines Issues abzufragen. Das spontane Besprechen von Issues könnte durch die Applikation unterstützt werden.

   Die erfasste Deadline bzw. der Aufwand um einen Issue zu bearbeiten wäre für mich sehr interessant. Auch der aktuelle Status des Issues eines anderen Entwicklers ist für mich notwendig um abzuschätzen, ob es möglich ist, die Deadline einzuhalten.

8. **Wie gehen Sie in Situationen, in denen Sie keinen Zugriff auf einen Computer haben, vor, wenn Sie einen Termin mit einem Teamkollegen vereinbaren möchten?**

   Ich verwende das Smartphone und suche manuell einen Termin mit dem Mitarbeiter.

9. **Mit AwareGlass können Sie verfügbare (freie) Termine auf der Basis Ihres eigenen Kalenders sowie dem des gescannten Entwicklers anzeigen.**

   a) **In welchen Situationen würden Sie Gebrauch von dieser Funktionalität machen?**

   b) **Welche Informationen würden Sie in diesen Situationen am meisten benötigen?**

   c) **Kann AwareGlass die vorher genannte Situation unterstützen bzw. das vorher genannte Medium ersetzen?**

   In Meetings, um spontan einen Termin zu finden, würde ich diese Funktionalität verwenden. Ich könnte mir auch vorstellen, wenn ich den Entwickler im Büro antreffe und mit ihm ein Problem genauer besprechen möchte, die Brille zu verwenden.

   Für mich ist vor allem wichtig, die freien Slots zu sehen. Die Vorbereitungen, die getroffen werden müssen, sind auch wichtig. Die Örtlichkeit und der Anfahrtsweg zum Meeting sind auch notwendig.

   Ich denke diese Funktionalität ist sehr nützlich, vor allem wenn es ein Meeting mit mehreren Entwicklern sein sollte. Diese Funktionalität habe ich bis jetzt weder auf dem PC, noch auf dem Smartphone und daher könnte diese Funktionalität einen gewissen Teil ersetzen. Wenn ich die Meetings aber erfassen möchte, dann denke ich nicht, dass die Brille die Funktionalität der anderen Geräte ersetzen könnte.

10. **Können Sie sich weitere Situationen vorstellen, in denen AwareGlass in Ihrer täglichen Arbeit zum Einsatz kommen könnte um mehr Awareness über die Arbeit Ihrer Teamkollegen zu erlangen?**

    a) **Welche?**

    b) **Welche Informationen wären in diesen Situationen für Sie am wichtigsten?**

    Keine.

Appendix B

Contents On The CD-ROM

- **Zusfsg.txt**: The unformatted Abstract text of this thesis in German.
- **Abstract.txt**: The unformatted Abstract text of this thesis in English.
- **Bachelorarbeit.pdf**: The bachelor thesis in PDF format.
- **application.zip**: The source code for `AwareGlass` in ZIP format.

Appendix C

Used Libraries, Tools and Plug-ins

- IDEs:
  - `Android Studio 1.5.1`: The IDE used for implementing AwareGlass
  - `IntelliJ IDEA Ultimate 15.0.3`: The IDE used for analyzing the source code
- SDKs, Libraries and Frameworks:
  - `Recon SDK`: The SDK used for implementing the application on the Recon JET glasses
  - `Android SDK`: The Android SDK used for implementing the application and creating instrumentation tests
  - `JUnit`: The unit testing framework used
  - `GSon`: The library used to parse JSON strings
  - `Joda Time`: The library used to implement date- and time-specific functionality
  - `JourneyApps ZXing Android`: The Android embedded implementation of the `ZXing` library, used to implement the scanning functionality
  - `Google ZXing`: The Google ZXing library needed for the embedded `ZXing` library
  - `EWS-android-api`: The Exchange Web Service library used to access `Microsoft Exchange` servers
  - `Mockito`: The framework used for mocking Java objects in the tests
- Other Tools:
  - `TeXstudio`: The LaTeX editor used to write this thesis
  - `OmniPlan`: The project management application used to plan this thesis and create the thesis website
  - `Androidscreencast.jnlp`: The application used to showcase `AwareGlass` in presentations
  - `PlantUML`: The plug-in for `IntelliJ IDEA` used to create the sequence diagrams.
  - `Balsamiq Mockups`: The application used to design the initial mock-up of `AwareGlass` (see Figure 4.1)

Bibliography

[BCSR07] Jacob T. Biehl, Mary Czerwinski, Greg Smith, and George G. Robertson. Fastdash: A visual dashboard for fostering awareness in software teams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1313–1322, New York, NY, USA, 2007. ACM.

[BD09] Andrew Begel and Robert DeLine. Codebook: Social networking over code. In *Proceedings of ICSE 09 (New Ideas and Emerging Results)*. Association for Computing Machinery, Inc., June 2009.

[BHG13a] Olga Baysal, Reid Holmes, and Michael W. Godfrey. Developer dashboards: The need for qualitative analytics. *IEEE Software*, 30(4):46–52, July 2013.

[BHG13b] Olga Baysal, Reid Holmes, and Michael W. Godfrey. Situational awareness: Personalizing issue tracking systems. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1185–1188, Piscataway, NJ, USA, 2013. IEEE Press.

[BNPL09] Andrew Begel, Nachiappan Nagappan, Christopher Poile, and Lucas Layman. Coordination in large-scale software teams. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering (CHASE)*, page 1–7, Washington, DC, USA, May 2009. IEEE Computer Society.

[bug16] Bugzilla homepage. `https://www.bugzilla.org`, 2016. [Online; accessed 16.03.2016].

[Dev16] Microsoft Office Developer. EWS Java API homepage. `https://github.com/OfficeDev/ews-java-api`, 2016. [Online; accessed 07.06.2016].

[DSAF99] Anind K. Dey, Daniel Salber, Gregory D. Abowd, and Masayasu Futakawa. The conference assistant: combining context-awareness with wearable computing. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*, pages 21–28, Oct 1999.

[ecl16] Eclipse homepage. `https://www.eclipse.org`, 2016. [Online; accessed 16.03.2016].

[FM10] Thomas Fritz and Gail C. Murphy. Using information fragments to answer the questions developers ask. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 175–184, New York, NY, USA, 2010. ACM.

[GBMN12] Anja Guzzi, Andrew Begel, Jessica K. Miller, and Krishna Nareddy. Facilitating enterprise software developer communication with cares. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 527–536, Sept 2012.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[Goo16] Google. GSON homepage. `https://github.com/google/gson`, 2016. [Online; accessed 07.06.2016].

[GPS04] Carl Gutwin, Reagan Penner, and Kevin Schneider. Group awareness in distributed software development. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 72–81, New York, NY, USA, 2004. ACM.

[GSPP05] Carl Gutwin, Kevin Schneider, David Paquette, and Reagan Penner. Supporting group awareness in distributed software development. *Engineering Human Computer Interaction and Interactive Systems*, pages 383–397, 2005.

[HCRP04] Susanne Hupfer, Li-Te Cheng, Steven Ross, and John Patterson. Introducing collaboration into an application development environment. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 21–24, New York, NY, USA, 2004. ACM.

[HD08] Rajesh Hegde and Prasun Dewan. Connecting programming environments to support ad-hoc collaboration. In *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on*, pages 178–187, Sept 2008.

[HP14] Javier Hernandez and Rosalind W. Picard. Senseglass: Using google glass to sense daily emotions. In *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST'14 Adjunct, pages 77–78, New York, NY, USA, 2014. ACM.

[IS15] Ashi Iram and Saqib Saeed. A case study of improving user experience of bug tracking applications. *Science International*, 27(2), 2015.

[jir16] JIRA homepage. `https://de.atlassian.com/software/jira`, 2016. [Online; accessed 27.05.2016].

[Jou16] JourneyApps. Zxing android embedded homepage. `https://github.com/journeyapps/zxing-android-embedded`, 2016. [Online; accessed 07.06.2016].

[KBHG14] Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W. Godfrey. Dashboards: Enhancing developer situational awareness. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 552–555, New York, NY, USA, 2014. ACM.

[KDV07] Andrew J. Ko, Robert DeLine, and Gina Venolia. Information needs in collocated software development teams. In *ICSE '07: Proceedings of the 29th international conference on Software Engineering*, page 344–353, Washington, DC, USA, May 2007. IEEE Computer Society.

[LBAK15] Stephan Lukosch, Mark Billinghurst, Leila Alem, and Kiyoshi Kiyokawa. Collaboration in augmented reality. *Computer Supported Cooperative Work (CSCW)*, 24(6):515–525, 2015.

[Lip16] Alex Lipov. EWS Android API homepage. https://github.com/alipov/ews-android-api, 2016. [Online; accessed 07.06.2016].

[Mic16] Microsoft. Microsoft Exchange homepage. https://products.office.com/de-ch/exchange/email, 2016. [Online; accessed 07.06.2016].

[Sch01] Till Schummer. Lost and found in software space. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, pages 10 pp.–, Jan 2001.

[ser16] Serena PVCS Pro homepage. http://www.serena.com/index.php/en/products/application-development/pvcs-pro/, 2016. [Online; accessed 27.05.2016].

[SNvdH03] Anita Sarma, Zahra Noroozi, and André van der Hoek. Palantir: raising awareness among configuration management workspaces. In *Software Engineering, 2003. Proceedings. 25th International Conference on*, pages 444–454, May 2003.

[SvG05] Margaret-Anne D. Storey, Davor Čubranić, and Daniel M. German. On the use of visualization to support awareness of human activities in software development: A survey and a framework. In *Proceedings of the 2005 ACM Symposium on Software Visualization*, SoftVis '05, pages 193–202, New York, NY, USA, 2005. ACM.

[VCTS15] Dhaval Vyas, Tara Capel, Deven Tank, and David Shepherd. Understanding the use of a bug tracking system in a global software development setup. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*, OzCHI '15, pages 222–226, New York, NY, USA, 2015. ACM.

[vis16] Visual Studio homepage. https://www.visualstudio.com, 2016. [Online; accessed 16.03.2016].