# Collaborative Program Comprehension based on Augmented Reality

Master's Thesis

Malte Hansen

July 6, 2021

Kiel University
Department of Computer Science
Software Engineering Group

Advised by: Prof. Dr. Wilhelm Hasselbring
Alexander Krause, M.Sc.

**Eidesstattliche Erklärung**

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel, 6. Juli 2021

# Abstract

The complexity of software systems increases and thereby maintainability and extensibility become more and more important. Thus, it is crucial to have tools at hand which facilitate program comprehension for various use cases. As complex commercial software systems are developed by large teams with changing team members, it is desirable for such tools to enable and promote collaboration among the involved software professionals.

In this thesis, we present a collaborative visualization approach for program comprehension which employs augmented reality (AR). We integrate our approach by means of an extension to ExplorViz, a web-based open source research and software visualization tool. To achieve AR, we combine printed markers and commercial off-the-shelf mobile devices, i.e. tablet computers and smartphones. For the visualization, we supplement the live camera feed of the employed devices with 3D software models which are aligned with the markers. We provide users with various options to adapt the visualization and retrieve information about the runtime behavior of a software system. The state of the 3D software models can be synchronized between multiple users, thus enabling for collaborative program comprehension.

We gathered preliminary feedback about our approach in a pilot study. Subsequently, a case study with 20 study participants, who solved program comprehension tasks in teams of two, was conducted. The study was conducted remotely due to the COVID-19 pandemic. Hence, our approach is evaluated in diverse environments with a wide range of mobile devices and browsers. The results indicate that collaborative program comprehension by means of an AR software visualization on mobile devices is a promising addition to existing visualization approaches. The comprehensive feedback of the case study also reveals possible adjustments to the implemented approach. We use the gathered feedback to present ideas for the further extension of our approach and conclude by presenting relevant topics for future work.

# Acknowledgments

First of all, I would like to thank Prof. Dr. Wilhelm Hasselbring for his support and for making my master thesis possible. The collaboration of the Software Engineering Group with the adesso SE also opened up further opportunities for me. In this context, I would like to thank Uwe Lutter for the thematic collaboration and the valuable experiences which I gained during my six months as a student employee at the adesso SE in Hamburg.

I would also like to specifically thank Alexander Krause and Stefan Carstensen for the supervision of my thesis. I look back on many interesting discussions which significantly supported the development of my approach and the elaboration of my thesis. In addition, I am grateful for the technical discussions with the developers of BIMSWARM and ExplorViz which provided me with new insights and ideas.

Furthermore, I thank all probands of my case study for their voluntary participation and the use of their time for the evaluation of my approach. I appreciate the extensive and helpful feedback that reached me.

Moreover, I would like to thank my family and friends for their interest in my work and constant support throughout the course of my thesis.

# Contents

Chapter 1

# Introduction

*ExplorViz*[1] is a software visualization tool which uses a 3D city metaphor to visualize data from dynamic program analysis. Recently, a virtual reality (VR) extension, which allows the collaborative exploration of software landscapes, has been integrated into ExplorViz [Hasselbring et al. 2020].

Based on this, we explore the potentials of a novel collaborative augmented reality (AR) approach for program comprehension. *BIMSWARM*[2] is a software in the field of building information modeling (BIM) which is currently under development and deploys a Java-based microservice architecture. As such, it is a well-suited piece of software to evaluate the envisioned approach.

## 1.1 Motivation

The complexity of software systems increases and thereby maintainability and extensibility become more and more important. Thus, it is crucial to have tools at hand which facilitate program comprehension. As complex commercial software systems are developed by large teams with changing team members, it is desirable for such tools to enable and promote collaboration among the involved software professionals.

The collaborative VR approach of ExplorViz is such a tool. However, this approach requires expensive hardware. We propose a collaborative AR approach for ExplorViz which uses commercial off-the-shelf tablet computers and smartphones. Our approach envisions to supplement the live camera feed of the used hardware with 3D models of the live trace software visualization which ExplorViz provides. An interactive visualization which is synchronized among the devices of software professionals could enhance the process of collaborative program comprehension.

We evaluate how software professionals can use our collaborative AR approach of ExplorViz to gather valuable information about software and in turn, increase their program comprehension. This is done by means of a case study in which BIMSWARM is analyzed. The design of the study needs to incorporate the diverse and unpredictable challenges which arise from the COVID-19 pandemic and the resulting governmental restrictions. It is

[1] http://www.explorviz.net

[2] https://www.bimswarm.de/

notable that the design of a study, which asks software professionals to use collaborative AR to explore and evaluate software, is still a novel field of research.

## 1.2 Document Structure

In Chapter 2 we give an overview of our goals. Chapter 3 is an introduction to the foundations and technologies which are necessary to achieve our goals. In Chapter 4 we take a look at related work to give an overview of existing and related visualization approaches. The gathered knowledge about foundations and related work is a building block for Chapter 5 which gives insights into the envisioned approach. Chapter 6 is concerned with the implementation of the developed concept. The resulting implementation is then evaluated in Chapter 7. At last, Chapter 8 summarizes the gathered results and takes a look at potential future work.

Chapter 2

# Goals

We have the overall goal to explore the potentials and limitations of a collaborative AR approach for program comprehension. Following this, the overall goal is split up into three goals which represent the required steps to achieve our overall goal.

## 2.1 G1: Concept for a Collaborative Augmented Reality Approach

The concept for the collaborative AR approach describes the envisioned visualization which aims at improving program comprehension. The realization of a collaborative AR approach requires several design decisions. The resulting set of design decisions builds the foundation for the later implementation and evaluation of the approach.

The concept should at least contain specifications about the visualization approach, the user interface (UI), desired target group, the envisioned use cases, and the minimal technical requirements. Proceeding from these specifications, existing technologies and software frameworks are compared to specify the technical environment which is used in the subsequent implementation.

## 2.2 G2: Development & Integration of the Collaborative Augmented Reality Approach

The second goal focuses on the architectural design and implementation of the collaborative AR approach. Both architecture and implementation should be in accordance with existing design principles of ExplorViz and make use of existing code where applicable. It is desirable to incorporate the AR features into the existing extension for VR, such that users can seamlessly switch between different visualizations.

An optional part of this goal is to enable interoperability between 2D, VR, and AR visualizations. This would enable cross-platform collaboration. However, as described in the next goal, we do not envision to evaluate cross-platform features within this context.

## 2.3 G3: Evaluation of the Collaborative Augmented Reality Approach

To gain insights into the potentials and limitations of the developed collaborative AR approach, our goal is to conduct a case study using a real-world application. BIMSWARM is a Java-based enterprise application in the realm of BIM and is currently in development. As such, it is a suitable test subject. The study aims to use off-the-shelf hardware, namely tablet computers, to evaluate the approach in a practically applicable and relevant scenario.

The active developers of BIMSWARM are valuable probands for the case study as they are familiar with BIMSWARM and can give informed feedback about potential applications and limitations of our approach within the design and development process of a complex software application. In preparation for the case study, a smaller pilot study with students should be conducted. The pilot study could identify improvements of the case study design and give hints about desirable changes to the implementation.

Chapter 3

# Foundations and Technologies

In the following, we give an overview of the necessary conceptional foundations and employed technologies. Following that, we give an overview of AR and VR in Section 3.3, including the software solutions for AR and VR, which are relevant for the envisioned approach. We start in Section 3.1 with the introduction of BIM before we present the BIM software BIMSWARM in Section 3.2. At last, we introduce the software visualization tool ExplorViz and its collaborative VR approach in Section 3.5.

## 3.1 Building Information Modeling

We plan to evaluate our approach by means of a case study that employs BIMSWARM, a piece of software in the realm of BIM. BIM is founded on the use of digital models for buildings, called building information models. In addition to their 3D information, building information models contain metadata about the represented physical entities. Furthermore, standardized processes are essential to effectively use BIM throughout all lifecycle phases of a building [Meins-Becker et al. 2019].

The number and definition of the individual lifecycle phases vary throughout the literature [Meins-Becker et al. 2019; Borrmann et al. 2018; Jiang et al. 2016]. For our purpose, we define the lifecycle of a building in a simplified manner but in accordance with Borrmann et al. [2018]. The lifecycle includes the phases of conceptual design, detailed design, construction, operation, and modification (see Figure 3.1). Modification includes the process of demolition. The overall process is represented as a cycle because both building material and gathered knowledge can be recycled for new building projects.

Beyond replacing paper drawings and utilizing the lifecycle of a building, BIM introduces new opportunities for automatic analysis, including checks for safety hazards [Aires et al. 2018]. Also, recent research explores how BIM can support meeting current building requirements, e.g. to build sustainably and energy-efficient [wu et al. 2017; Jiang et al. 2016].

## 3.2 BIMSWARM

BIMSWARM is a piece of software under development in the realm of BIM. As such, it is a candidate for our case study to test the collaborative AR approach. BIMSWARM is an online

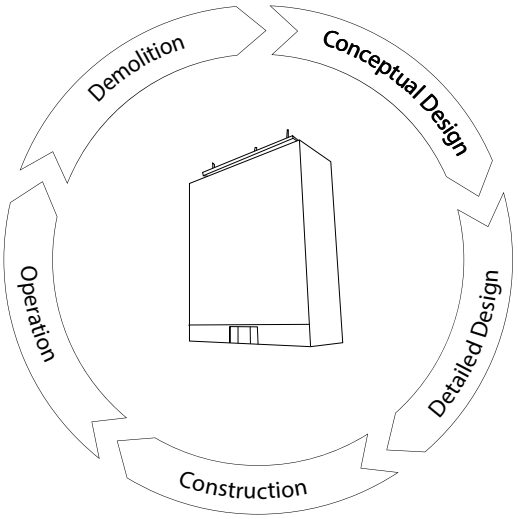



**Figure 3.1.** Lifecycle phases of a building (adapted from Borrmann et al. [2018])

marketplace which aims at enabling users to easily find, evaluate, and combine BIM-related software products. It is developed as part of a research project whereby *planen-bauen 4.0*[1] acts as coordinator and *adesso*[2] is responsible for the software implementation.

The backend of BIMSWARM is composed of microservices. An overview of the functionally relevant microservices and some of their relations is presented in Figure 3.2. *Eureka*[3] is a service registry and used by BIMSWARM to introduce the remaining microservices to each other. Just as Eureka itself, all microservices use Representational State Transfer (REST) for inter-service communication.

Zuul[4] is employed by BIMSWARM as a gateway to dynamically route requests to other microservices and increase resilience. The remainder of the depicted microservices in Figure 3.2 are used to implement the core features of BIM. The development of BIMSWARM focuses on four main features, namely BIMSWARM-Marketplace, BIMSWARM-Certification, BIMSWARM-API, and BIMSWARM-Composer.

The *BIMSWARM-Marketplace* provides a user with an overview of software in the realm of BIM. In addition to a simple property-based search, users should be provided with a search for BIM software which takes their employed use cases into account.

The *BIMSWARM-Certification* aims at supporting users in their choice of software. BIMSWARM can provide information about certifications of the offered software products. A software product can be certified by independent authorities and for example, could

[1] https://planen-bauen40.de/

[2] https://www.adesso.de/de/

[3] https://github.com/Netflix/eureka

[4] https://github.com/Netflix/zuul

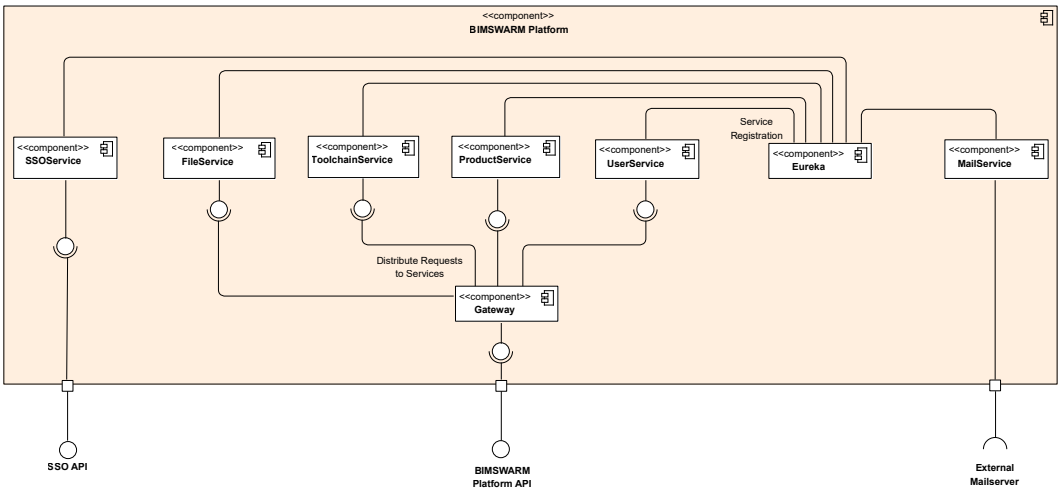



**Figure 3.2.** Overview of the microservices and interfaces of the BIMSWARM backend.

provide information about the compatibility with other products or the use of open technology standards. In addition to the certification of single products, product types and toolchains might acquire a certification.

The *BIMSWARM-API* is a coherent API which can be used by BIM software to enable them to work in combination with other BIM software products. For example, a single sign-on (SSO) feature allows users to enter login credentials only once in order to use multiple BIM software products which support the BIMSWARM-API. Additionally, BIMSWARM provides the software products with a common data environment (CDE) to facilitate sharing of data throughout BIM processes.

The *BIMSWARM-Composer* takes use cases or processes of a user and can determine which software products are capable to achieve the desired tasks in combination. The result is a possible toolchain. The compatibility of the software products, with regard to the processes and data flows, is ensured with help of the BIMSWARM-API.

In sum, BIMSWARM aims at supporting and combining BIM software for all BIM lifecycle phases and the transitions between them. As such, it tackles the problem of incompatible BIM software, which is one of the major technical challenges for the current use of BIM in the industry [Azhar 2011].

## 3.3 Extended Reality

For program comprehension and software visualization, a variety of hardware devices and approaches which augment or replace the real environment are explored [Chotisarn et al. 2020]. The degree to which a visualization makes use of real or virtual objects respectively,

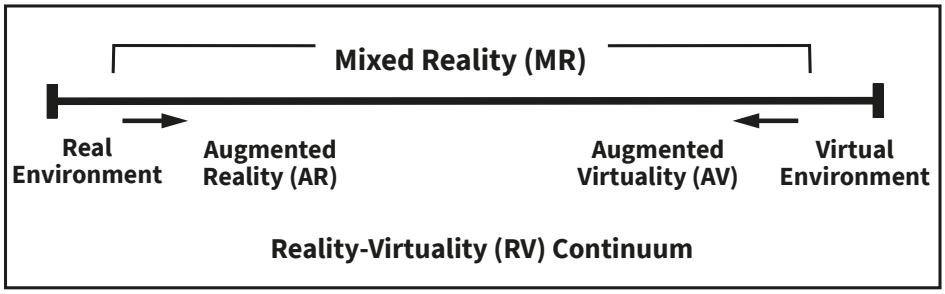



**Figure 3.3.** Simplified reality-virtuality continuum by Milgram et al. [1994]

can be categorized with respect to the reality-virtuality (RV) continuum (see Figure 3.3). As mixed reality (MR) describes the combination of both real and virtual environments, we use the term extended reality (XR) to subsume AR, VR, and related technologies. In the following, we introduce the concepts for AR and VR.

### 3.3.1 Augmented Reality

In the reality-virtuality continuum, AR is categorized as a mixed reality (MR), which adds virtual elements to a real environment. However, as augmented virtuality (AV) is rarely used, there exist broader definitions which focus on the user experience. For example, Azuma [1997] defines AR as a combination of real and virtual (1), interactive in real-time (2), and registered in 3D (3), whereas registration of 3D means that 3D virtual objects are added to a 3D real environment. For the context of this work, we adopt this definition as it represents the properties of AR which we aim for in our envisioned approach. In addition, this definition does make assumptions about the employed hardware devices.

### 3.3.2 Virtual Reality

In the reality-virtuality continuum VR is categorized as a virtual environment, i.e. VR does incorporate no elements of the real environment into the visualization for the user. However, since VR-capable devices often allow the movement of users, in practice a VR visualization might include indicators for barriers in the real world to prevent collisions.

VR tries to let a computer-generated world feel as real and immersive as possible for a user [Billinghurst et al. 2001; Robertson et al. 1997]. To enable the perception of depth, i.e. stereopsis [Ohzawa 1998], the left and right eye are usually given two slightly different images. Furthermore, headphones or even omnidirectional treadmills might be employed in addition to the visual input to further decouple the user from the real environment.

In 1968, a head-mounted display (HMD), known as 'Sword of Damocles', was developed [Sutherland 1968]. Afterwards, especially recently, there have been numerous technical

developments in the realm of virtual reality [Anthes et al. 2016], e.g. the Oculus Rift[5] or HTC Vive Pro VR system.[6]

## 3.4 Web Technologies

In this section, we take a closer look at web technologies which we plan to use or take their use into consideration. Namely, we present AR.js, WebXR, Hammer.JS, and WebRTC.

### 3.4.1 AR.js

*AR.js*[7] is a JavaScript library which provides functionalities for the use of AR in the browser. Given a video feed, e.g. of a webcam or an integrated camera of mobile devices, AR.js provides three ways to introduce virtual objects to a given video in order to achieve AR, namely image tracking, location-based AR, and marker tracking.

For *image tracking*, AR.js aims at detecting a provided image within the video. If such an image is detected, its position can be used to place virtual objects within the visualization, e.g. on top of the image. This approach is quite versatile as arbitrary images might be selected to get reference points for the visualization. However, depending on the image, the detection accuracy might vary and the approach is compute-intensive. Undesirably, this leads to increased power consumption for mobile devices.

*Location-based AR* makes use of positional information which can be provided by modern smartphones, e.g. via their GPS, WLAN, or gyro sensors. Use cases include the use of virtual objects to provide a user with information for navigation in foreign environments.

For *marker tracking*, AR.js uses images which are easy to recognize, called markers. Markers employ two high contrast colors, e.g. black and white. If additionally a low resolution is used, markers can be recognized within the video of a camera with low computational effort. Markers can be printed on paper or displayed on screens. The recognized markers are points of reference to add and position virtual 3D models to the video, e.g. on top of a marker.

As AR.js uses widely adopted web technologies such as *WebGL*[8] and *WebRTC*[9], it is compatible with commonly used modern browsers.

### 3.4.2 WebXR

*WebXR*[10] is an application programming interface (API), which brings support for AR, VR, and related devices to the web browser. The term XR subsumes AR, VR, related

[5] https://www.oculus.com/rift/

[6] https://www.vive.com/de/product/vive-pro/

[7] https://github.com/AR-js-org/AR.js/

[8] https://get.webgl.org/

[9] https://webrtc.org/

[10] https://www.w3.org/TR/webxr/

technologies in this context. WebXR is the successor of the experimental API for VR in web browsers, *WebVR*.[11]

In order to present XR content to a user, WebXR exposes a list of all connected and supported XR hardware devices. After a session for a device is initiated, content can be rendered, e.g. via WebGL, to the respective display(s). WebXR also provides access to positional information and contains events for user inputs.

Since WebXR uses *JavaScript*[12] and both AR and VR become more widely available, support for many browsers across a variety of devices and operating systems can be expected for the future. However, WebXR was recently released in 2018 and as of now, only Chrome[13], *Egde*[14], Chrome for Android, and *Samsung Internet*[15] offer native support [*MDN Web Docs WebXR Device API*]. However, web browsers which offer support for WebVR, like *Firefox*[16], there exists a *WebXR Polyfill*.[17]

### 3.4.3 Hammer.JS

*Hammer.JS*[18] is a Javascript library for the recognition of touch gestures in web browsers. Besides simple single-tap and multi-tap recognizers, Hammer.JS also provides recognizers for multi-touch gestures. These include recognizers for pan, pinch, and rotate gestures. With regard to these recognizers, Hammer.JS is highly configurable. Each recognizer for a gesture can be set to trigger for a specific number of fingers on a touchscreen. In addition, it can be specified which gestures can be combined or take precedence over each other.

### 3.4.4 WebRTC

*WebRTC*[19] is an API for the web, which enables real-time communications. Among others, it supports the direct transfer of audio and video data, including the streaming from cameras and microphones. Almost all modern browsers, including mobile browsers, support WebRTC by default.

## 3.5 ExplorViz

ExplorViz [Fittkau et al. 2017; Hasselbring et al. 2020] is a piece of software for the visualization and monitoring of software landscapes. It employs dynamic program analysis

[11] https://webvr.info/
[12] https://www.javascript.com/de
[13] https://www.google.com/intl/en_gb/chrome/
[14] https://www.microsoft.com/en-us/edge
[15] https://www.samsung.com/uk/apps/samsung-internet/
[16] https://www.mozilla.org
[17] https://github.com/immersive-web/webxr-polyfill
[18] https://hammerjs.github.io/
[19] https://webrtc.org/

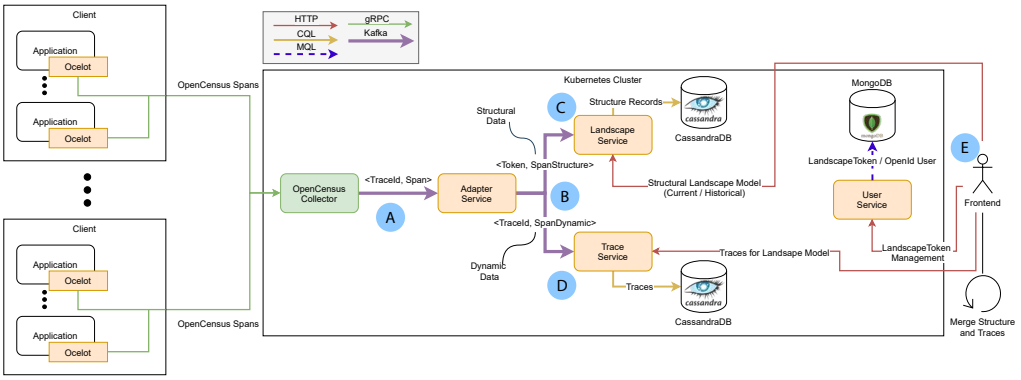



**Figure 3.4.** An overview of the architecture of ExplorViz.

to provide a live trace visualization of the monitored data. As such, ExplorViz is a tool to improve program comprehension, i.e. to support users in software maintenance activities.

In the following, we take a closer look at the architecture of ExplorViz and its visualization approach.

### 3.5.1 Architecture

ExplorViz has been under continuous development since 2012. In 2017 a structural change from a monolithic architecture to an architecture organized in microservices was realized in ExplorViz [Zirkelbach et al. 2018]. The most important components of ExplorViz are the backend and frontend. The backend, which employs a microservice architecture, is mainly written in Java. The data flow and implemented microservices are displayed in Figure 3.4.

For the envisioned case study it is necessary to collect live trace data, i.e. employ a dynamic program analysis framework. For Java-based applications, ExplorViz employs inspectIT Ocelot.[20] inspectIT Ocelot is a Java agent which collects performance, tracing, and business data. For our context, the collected data about trace executions is the most relevant feature.

The trace data is exported in accordance with the *OpenCensus*[21] standard, a library for application metrics and distributed traces. The data of the Java agent is received by a collector via *gRPC*[22] and then forwarded via *Kafka*[23] (Figure 3.4 - A) to the Adapter Service. The Adapter Service processes the incoming data such that it is split into structural and dynamic data (Figure 3.4 - B). The Landscape Service (Figure 3.4 - C) then manages and persists the structural data, e.g. data about the structure of applications, packages, and

[20] https://www.inspectit.rocks/
[21] https://opencensus.io/
[22] https://grpc.io/
[23] https://kafka.apache.org/

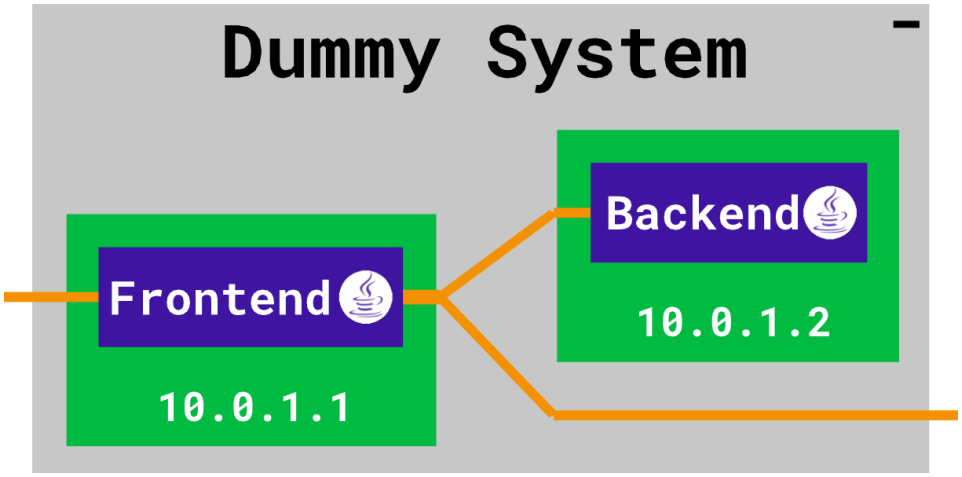



**Figure 3.5.** An example system and its content as displayed by the ExplorViz landscape view.

classes. On the other hand, the Trace Service (Figure 3.4 - D) manages and persists dynamic data, i.e. data about observed method calls.

The collected data can then be retrieved by the frontend (Figure 3.4 - E). The frontend uses the Ember.js[24] web development framework and is executed by the user's web browser. Therefore, it is mostly written in *TypeScript*[25] and JavaScript. Through this architecture, the development of extensions is very flexible [Zirkelbach et al. 2019] and through the use of JavaScript on the client-side it is possible that ExplorViz can be used system-independently with modern browsers. The employed visualization approach of the frontend is explained in further detail in the following section.

### 3.5.2 Landscape and Application View

The visualization of ExplorViz is rendered on a `canvas` using WebGL. To create and arrange the virtual objects for the `canvas`, *three.js*[26], a popular Javascript 3D library, is used. ExplorViz uses two different views [Fittkau et al. 2015]. In Figure 3.5 a snippet of the landscape view for an exemplary landscape is depicted.

The landscape view is a two-dimensional representation of a software landscape and is particularly suitable to get an overview of the monitored software. There are applications (blue), nodes (green) and systems (grey). Nodes can represent a physical server which can be identified by an IP address. Systems on the other hand are a semantic construct

[24] `https://emberjs.com/`
[25] `https://www.typescriptlang.org/`
[26] `https://threejs.org/`

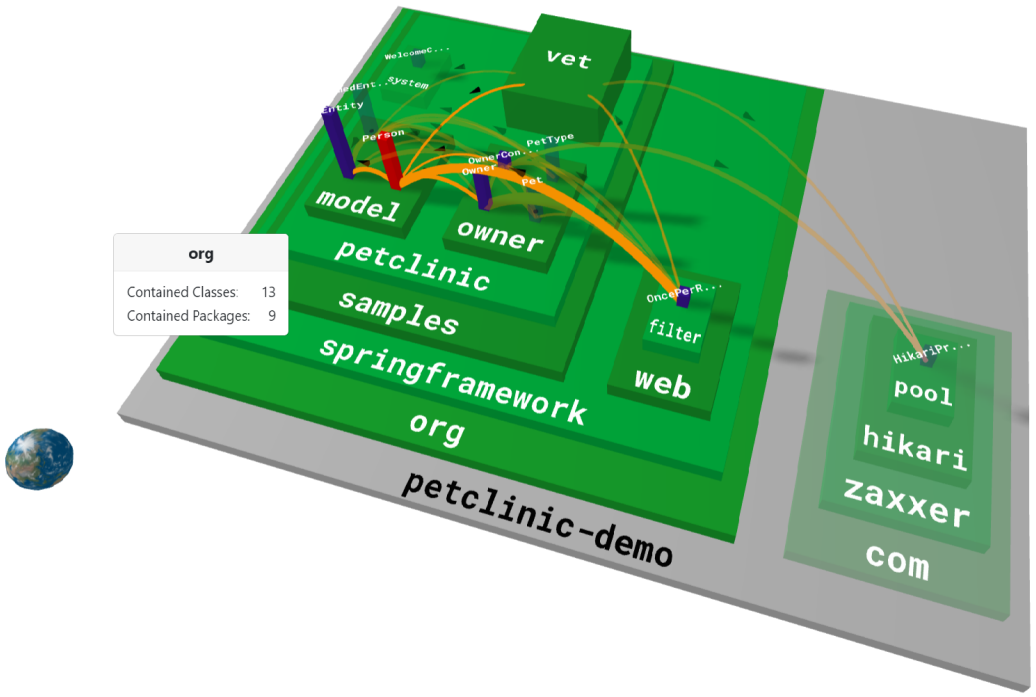



**Figure 3.6.** A snapshot of the Spring PetClinic as displayed by ExplorViz.

and therefore cannot be detected automatically. The communication between software is represented by orange lines. The thickness of the lines correlates with the number of calls it represents. Interaction with an application (e.g. via a double click) leads to the application view (see Figure 3.6).

The application view represents a three-dimensional model of the software and offers many interaction possibilities. Software packages are shown in green, which in turn can contain packages or individual classes (blue). The height of a class model indicates the number of objects belonging to the class. Here, too, the communication between objects is represented with orange lines. One can highlight individual classes and components or call up additional information for a component or class in form of a popup. This representation of applications is supposed to be a metaphor for a three-dimensional city [Dieberger and Frank 1998] [Wettel and Lanza 2007]. Packages can be interpreted as districts, classes correspond to houses, and communication between classes is comparable to streets as it is connecting classes with each other.

Chapter 4

# Related Work

In this chapter, we take a closer look at related approaches and existing implementations in the realm of software visualizations. To the best of our knowledge, there are no approaches which employ AR for collaborative program comprehension with tablet computers. Therefore, we present related work which focuses on comparable software visualization approaches.

## 4.1 ExplorViz

ExplorViz [Fittkau et al. 2017; Hasselbring et al. 2020] is a piece of software for the visualization and monitoring of software landscapes (see Section 3.5). Building on top of its core features, ExplorViz includes an extension for collaborative VR which has undergone several design iterations [Zirkelbach 2021].

First, Krause [2015] introduced VR to ExplorViz using an early version of the Oculus Rift as a HMD and Microsoft Kinect[1] to realize motion controls. With the growing maturity of standalone VR solutions, Häsemeyer [2017] adapted the VR implementation such that VR systems with dedicated controllers, e.g. the HTC Vive[2], HTC Vive Pro, and Oculus Rift are supported. Building upon that, König [2018] and Hansen [2018] implemented the collaborative VR approach in its current state.

The collaborative VR approach, just as the rest of the frontend of ExplorViz, uses three.js. WebXR is used to access VR devices. As three.js offers support and documentation for the use of WebXR, the combination of these technologies is straightforward. However, as Chrome does not yet support standalone VR devices such as the HTC Vive [*W3C Chrome Hardware Support*], Firefox in combination with the WebXR Polyfill is used.

For the VR visualization approach, a user enters a virtual world, which contains a square floor. An example of the VR visualization can be seen in Figure 4.1. On top of the floor, a 3D adaption of the landscape model is placed. The VR controllers are represented by 3D models. Colored rays originate from the front of the controllers. The rays aid the interaction with software models from a distance such that a user can use a controller to manipulate objects which intersect with its corresponding ray. One controller, referred to as the interaction controller, is augmented with a red ray and mainly used to manipulate

[1] https://developer.microsoft.com/de-de/windows/kinect/

[2] https://www.vive.com/de/

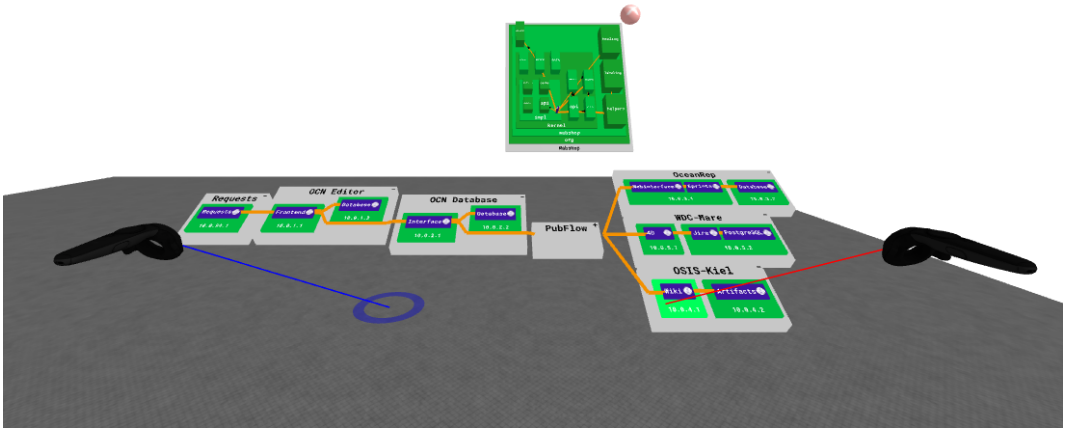



**Figure 4.1.** An example landscape and one opened application as displayed by the ExplorViz VR approach. On the left is a model of the utility controller and on the right a model of the interaction controller.

the landscape and applications. Most actions, e.g. opening and closing entities of the landscape and applications, are achieved by pointing at the respective entity with the ray and actuating the controller's trigger button. This action, applied to an application within the landscape model, allows a user to open an application model which can be moved and interacted with by pressing and holding a button on the side of the controller.

The second controller, referred to as the utility controller, is mainly used for additional features like highlighting classes and teleporting within the 3D space. Teleporting is achieved by actuating the trigger of the utility controller when pointing at the desired position on the virtual floor. This allows a user to move in the virtual environment independent of the space restrictions which the physical room might have.

The interactions with the landscape and applications are supplemented by textual overlays and menus. The canvas-based menus are attached to the utility controller and allow additional functionality like the rotation of the landscape model or to connect with other users. By connecting with other users, the collaborative VR experience is started. Other users are represented by 3D models of their HMD and their controllers. An extension for the backend ensures that actions, e.g. opening and closing landscape entities, are synchronized among all users.

The VR extension for ExplorViz shares several similarities with our approach. Mainly, the visualization of the software models is comparable as we want to use the same 3D models. In contrast, as our approach employs AR, the real environment of users is incorporated into the visualization whereas the VR approach does only visualize a virtual

floor. The main difference originates from the different hardware requirements. The VR approach requires expensive hardware, our approach relies on the use of widespread tablet computers.

## 4.2 SkyscrapAR

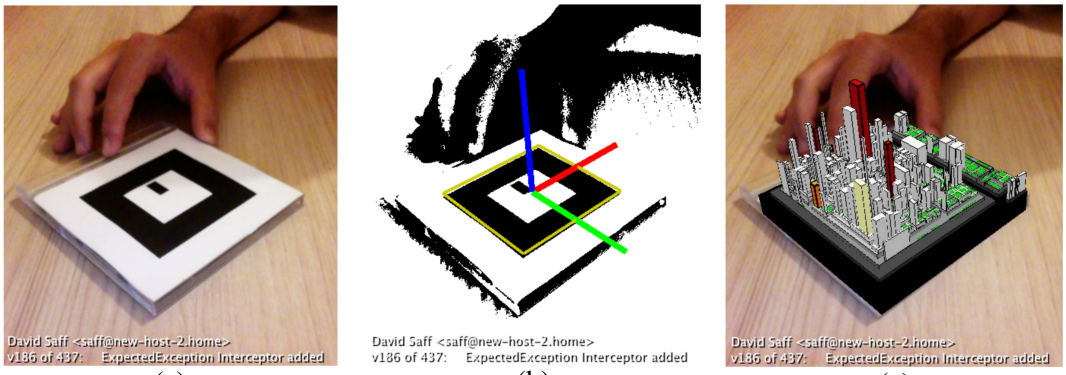
SkyscrapAR [Souza et al. 2012] uses an AR software visualization approach to depict the process of software evolution. The representation of packages and classes is a variation of the widespread 3D city metaphor. The classes can be interpreted as buildings and are placed on top of rectangular quarters which represent packages. As SkyscrapAR is concerned with software evolution, the visualization displays a selected revision of a piece of software and can change over time.

Even though classes can be added or removed throughout the development of software, a static layout is used. For this, data about all packages and classes which have existed at any time within the software is collected and using that information a layout for the metaphorical city is calculated which could accommodate all those packages and classes. The packages for classes which are not present at a given software revision are represented as green lots. Classes which are changed in the current revision are depicted as red buildings. In addition, the user can highlight individual classes in yellow or orange. In addition to the colors, the footprint of a building gives a viewer insights into the lines of code (LOC) for the associated class and the height of the building correlates with the number of recent modifications to that class.

The software model of an application is displayed in AR by SkyscrapAR. To have a point of reference for the placement in the real environment, printed markers (Figure 4.2 (a)) are placed such that a webcam can capture it as part of an image. The image is then transformed to black and white such that the rectangular marker and its content can be used to calculate a matching coordinate system (Figure 4.2 (b)). At last, the rendered 3D model is centered on the marker with respect to the calculated coordinate system (Figure 4.2 (c)).

SkyscrapAR is executed on computers and thus a keyboard and mouse are used as input devices. A mouse can be used to highlight classes or hover on a class to display information like the class name or the LOC. The keyboard is used to switch to another software revision, filter out buildings, or scale the application. Changing the orientation of the application is achieved by rotating the underlying marker.

The approach of SkyscrapAR shares similarities with our approach as it uses common hardware and easy to produce markers to achieve an AR software visualization. However, the use of a desktop computer and webcam prohibits the movement within the real environment such that the viewed image remains mostly stationary. In addition, keyboard and mouse do not represent the ideal input devices for this visualization, which is also recognized by Souza et al. [2012]. Aside from the use of the 3D city metaphor, the visualizations share few similarities because SkyscrapAR is concerned with software




(a) (b) (c)

**Figure 4.2.** For the visualization approach of SkyscrapAR [Souza et al. 2012], pictures containing a marker are captured using a camera (a), processed to calculate a corresponding coordinate system (b). The coordinate system is used to place and align the 3D model (c).

evolution and ExploViz is concerned with software behavior.

## 4.3 IslandViz

IslandViz [Schreiber et al. 2019; Seipel et al. 2019] is a software visualization tool which provides visualizations for both AR and VR. In this section, we focus on the visualization approach for AR.

IslandViz, as the name suggests, employs a custom metaphor to visualize component-based software architectures called island metaphor. The overall software system is represented as an ocean containing several islands representing bundles (applications). The islands are partitioned into regions of different colors and irregular shapes through which the island's appearance shares similarities with political maps. The regions represent software packages. Only regions which contain classes, represented by tall buildings, are visualized. Thus, the hierarchical structure of packages is neglected to simplify the visual metaphor. To visualize dependencies between bundles, green and red ports are placed aside an island. Here, green ports with corresponding arrows indicate import dependencies to another bundle and red ports are used to depict export dependencies.

To realize the visualization in AR, IslandViz focuses on the Microsoft Holo Lens as a hardware solution. The Holo Lens makes use of translucent glasses on which images are projected to add 3D virtual objects to the real environment. The Holo Lens possesses several sensors to scan the environment and calculate a 3D representation of it through which AR content can be aligned with real objects. Figure 4.3 illustrates that the 3D models for IslandViz can be placed freely on real objects, e.g. on a table. The Holo Lens does not come with dedicated input devices. Therefore, the interaction with IslandViz is realized

**Figure 4.3.** AR Visualization of IslandViz [Schreiber et al. 2019]. An "Air-Tap" gesture can be performed to select a bundle.

through gestures, voice commands, and gaze actions. An "Air-Tap" can be performed to select bundles and bring up additional information. A two-handed gesture, similar to a pinch gesture on mobile devices, allows a user to change the zoom of the visualization. A cursor which is navigated by gazing can be combined with a "Tap-and-Hold" gesture to navigate through the virtual ocean. At last, natural language processing is employed to allow more complex interactions. For example, a user can filter and select bundles through voice commands.

From a technical perspective, the Holo Lens uses the Universal Windows Platform (UWP) as a runtime environment. IslandViz is developed with *Unity3D* [3], which provides support for both UWP and the Holo Lens specific Mixed Reality Toolkit [4] (MRTK). The MRTK has built-in support for sharing application states over multiple devices. IslandViz utilizes this feature to allow for collaborative software exploration. For example, two users, who are both wearing a Holo Lens, can stand in the same room and interact with the AR visualization of IslandViz which is placed on a table. This collaborative scenario is similar to the single-user scenario depicted in Figure 4.3.

Overall, IslandViz is a software visualization tool which allows to view and explore software systems in AR. Just like our approach, it strives to enable users to work collaboratively. In contrast, the visualization approach and required hardware, as well as the employed software technologies, differ. IslandViz employs the island metaphor which aims

[3] https://unity.com/

[4] https://github.com/microsoft/MixedRealityToolkit-Unity

to enhance immersion through a detailed depiction of islands which are placed within a virtual ocean. Even though ExplorViz is also based on a metaphor, the 3D city metaphor, we do not pursue the goal to resemble real-world cities as closely as possible. We provide a user with a landscape model, which gives an overview of the software system, and application models which are suitable for a more detailed software exploration. These models can be explored independently from each other as they can be placed on different markers. At last, IslandViz uses specialized and expensive hardware for AR whereas our approach uses widely available and affordable hardware.

Chapter 5

# Concept

In this chapter, we introduce the concepts for our collaborative AR approach. We start by introducing the general ideas, including envisioned use cases, for our approach. The Chapter then goes into more detail by specifying the required hardware before we present a draft for the user interface. We continue by presenting how our approach can be incorporated with the existing implementation of ExplorViz. The chapter is concluded by a discussion about the available software technologies which are suitable to implement our approach.

## 5.1 Classification

In this section, we classify the envisioned software visualization approach which we employ for collaborative program comprehension. Maletic et al. [2002] propose a taxonomy for software visualization. Therefore, a software visualization is characterized by the five dimensions *tasks*, *audience*, *target*, *medium*, and *representation*.

The *tasks* define why a software visualization is needed. For our approach, we envision the main task to be familiarization with software systems such that the maintenance of software systems is facilitated. In addition, the analysis of the visualized program behavior could also be used to identify architectural shortcomings or identify unwanted dependencies between classes.

The *audience* defines who will use a software visualization approach. We primarily envision software developers and software architects with varying levels of expertise for our approach. As we also envision the collaborative use, we also assume that the immersive character of AR can be used in some use case scenarios which include persons with a background in software development. We present the envisioned use cases in more detail in Section 5.2.

The *target* defines the employed data source and what aspects of a piece of software are visualized. We want to incorporate our approach seamlessly into ExplorViz and therefore adopt the same data as used for the existing visualization (see Section 3.5).

The *medium* specifies which devices are used for visualization. We envision to support the use of mobile devices, i.e. tablet computers and smartphones. We describe the properties of the required hardware in more detail throughout Section 5.3.

The *representation* defines how the available data is presented. In general, we want to adopt the existing 3D software models of ExplorViz that are already employed for its VR

extension (see Section 4.1). The visualization and user interface is specified in more detail in Section 5.4.

## 5.2 Use Cases

In general, AR visualizations tend to be more immersive as they combine the real environment with virtual objects. There are several use cases for which our visualization approach could be suitable due to the use of mobile devices and the immersive character of AR. Firstly, we present examples for use cases in which our approach is used alone as opposed to collaboratively. These example use cases include the *familiarization* or *mobile solution*.

For *familiarization*, developers or software architects who are unfamiliar with a given software system can use our approach to become more familiar with the overall architecture and behavior of the analyzed software system. Mobile devices are very common and our approach does only require a browser and markers in addition. Notably, it is sufficient to display the markers on a monitor as opposed to printing them to try out our approach.

Our approach could also be suitable as a *mobile solution* for software visualization. Only a mobile device and a marker, both of which are portable, are required. This opens up the potential for brief presentations about software architecture and behavior. For example, the immersive and three-dimensional properties of our visualization approach could be appealing to customers and complement ordinary presentations.

We do not only focus on an AR visualization for tablet computers but want to enable users to work collaboratively. We also envision use cases where our approach could be used collaboratively, including *meetings* and a *pair programming* use case.

*Meetings* are a common practice in the realm of software engineering. It could be beneficial to incorporate findings which originate from software visualization tools like ExplorViz into the meeting. However, the use of desktop computers or laptops can divert the attention of meeting attendees. Our approach employs mobile devices in conjunction with printed markers. We provide collaborative features, employ small devices, and use markers which can be moved and pointed at, thereby encouraging and facilitating communication.

*Pair programming* is the practice of developing software in pairs of two at a single workspace. Thereby, one developer is writing code while the other developer is reviewing the written code and is giving advice. This practice is employed to increase software quality, especially for the development of complex software systems. Since recently, pair programming might also be conducted remotely as opposed to the developers occupying a single workspace. We suspect that our approach can be used in the fashion of pair programming, i.e. multiple software developers could collaborate to achieve a common understanding of a piece of software. The analysis of the architecture or runtime behavior of software can be complex. By working in a team and exchanging views on different aspects of the depicted visualization, the program comprehension of the collaborating users could be improved. This use case could be particularly beneficial for users who have different backgrounds or different levels of expertise. At last, the practice of pair

programming could be augmented by our visualization approach since ExplorViz uses live trace analysis. Thus, changes to the behavior of a software system could be inspected during the development process.

## 5.3 Required Equipment

In this section, we present the required equipment to use the envisioned collaborative AR approach. This includes tablet computers, markers, and the required deployment infrastructure.

### 5.3.1 Tablet Computers

Our approach aims to provide an augmented reality solution through the web browser. Our only hard requirement for devices is that they possess a modern web browser and a camera which can be accessed within the web browser. Up until now, ExplorViz does not support visualizations, input methods, or features which are designed for mobile devices which possess small touchscreens.

We aim to provide a visualization approach which is usable for the vast majority of mobile devices. However, we assume it to be unlikely that smartphones are used extensively for software exploration due to their limited screen size. Therefore, we consider that smartphones might be used temporarily but we focus our development efforts on an implementation which is suitable for tablet computers. Tablet computers can be characterized as devices which have a smaller screen than computer monitors but do have a significantly larger screen than smartphones.

### 5.3.2 Markers

In addition to the technical requirements, we need to specify which other requirements in terms of physical equipment are present. For our approach, these are markers which act as points of reference in the real world. Thus, they are used to position landscape and application models as expected in the virtual coordinate system on top of the markers. The markers themselves require well-thought-out design decisions, i.e. regarding their size, employed symbols, and the choice of material.

Markers for AR.js include a black rectangular border that contains the marker's content. The background is white to increase the contrast and improve recognition. In terms of symbols, we plan to use a marker design similar to the often used and approved Hiro marker[1], which depicts the word "Hiro". As opposed to the Hiro marker, which is designed for a general use, the use for our markers is known in advance, i.e. landscape and application models.

[1] https://commons.wikimedia.org/wiki/File:Hiro_marker_ARjs.png

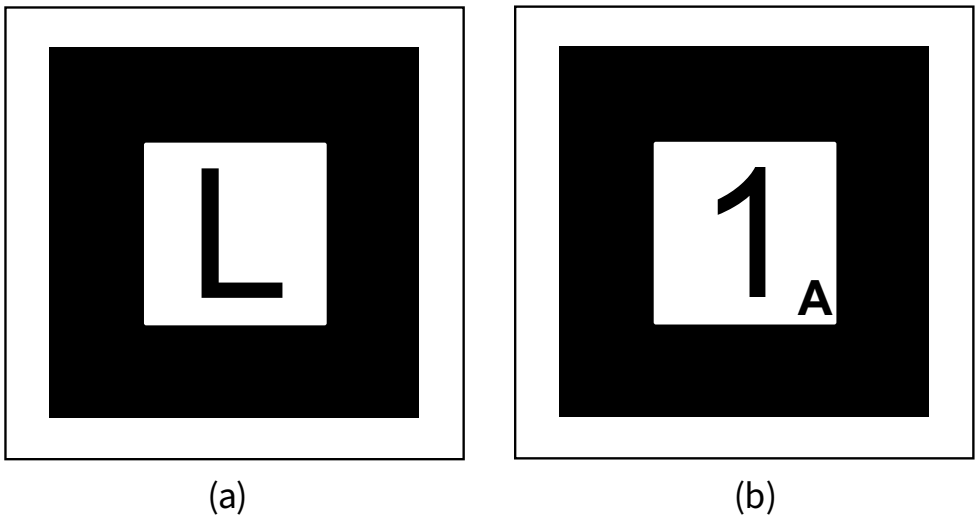



**Figure 5.1.** Marker concepts for landscape (a) and application models (b). The outer black border represents the edges of the paper on which the marker shall be printed.

Thus, we choose to depict semantically meaningful letters and numbers on the markers to aid users. We use the letter "L" to indicate that a marker is used for landscape models and refer to this kind of markers and *landscape markers*. The landscape model is required to open applications and gives an overview of the running applications.

Complementing the landscape markers are markers on which application models are placed, referred to as *application markers*. As software landscapes often contain multiple applications, multiple markers with different content are required. Therefore, we choose to display a number prominently on application markers and opt for a one-to-one mapping of numbers on application markers and different application models. We suppose that allowing multiple identical application models on different markers could lead to confusion. To remind the user of the semantic meaning of the application markers, a small letter "A" is printed on the bottom right. A marker design that illustrates the above-mentioned considerations for a landscape and the first opened application can be seen in Figure 5.1.

Next, we take a look at the suitable materials for the markers. As a user should be able to produce markers easily at home or at his or her workplace, printable paper is the only viable option for our approach. However, preliminary tests have shown that the usual paper with 80 grams per square meter has some drawbacks. Firstly, it is easily damaged by regular use. Secondly, thin paper is rather flimsy such that it tends to be uneven which could negatively impact the accuracy of marker recognition. At last, thin paper can not be easily picked up or moved when it lays flat on a surface without bending it. Thus, we

recommend the use of thicker paper which is still printable, i.e. in the range of 200 to 400 grams per square meter. Such paper exhibits improved durability, keeps its shape well, and can be picked up and moved easily with one hand.

The last design decision is about the size of the markers. We suggest that all markers should have the same dimensions to simplify the creation, storage, and usage of markers. On the one hand, markers should not be too large because they could become bulky or even tend to be only partly visible on a tablet's camera feed. On the other hand, markers should be easy to read for the user and reliably recognized by the employed software framework. In the same manner as the other design decisions, we aim to use widely available products. Concluding, we choose a rough format of 10 cm x 10 cm for the black rectangle which contains the marker symbols. As a white background around the black square improves recognition accuracy, we aim to print the markers on DIN A5 paper. DIN A5 is half the size of DIN A4 and thus can easily be acquired by cutting a regular DIN A4 paper in half. As DIN A5 is not a square shape, the printed markers have significantly more white space in one direction compared to the other. We assume this to be helpful for users in order to assess the correct orientation of the marker.

### 5.3.3 Deployment Infrastructure

The backend of ExplorViz with our contributions needs to run on a computer. As ExplorViz can be deployed with *Docker* [2] containers, a wide support of devices is ensured. However, it should be considered that both ExplorViz and the monitored application need to be executed. This can be done on a single device but requires sufficient compute power and main memory. For technical details about ExplorViz we refer to Section 3.5. For the remainder of this section, we assume that ExplorViz is deployed on a server and can be accessed easily through modern web browsers.

## 5.4 Representation

In this section, we present our concept for the employed visualization and the corresponding user interface.

For visualization in AR, we would like to adapt the existing visualization of ExplorViz. By splitting it into a landscape model, which gives an overview of the instrumented applications, and several application models, the visualization of ExplorViz is already designed to be modular. It therefore certainly makes sense to place the corresponding 3D models onto different markers. This way, the landscape would have its own marker. But since there can be any number of applications, they should not be automatically placed on markers. We, therefore, think it makes sense that by interacting with the landscape, applications can be opened and placed on a marker, which is still free. Here, we consider

[2] https://www.docker.com/

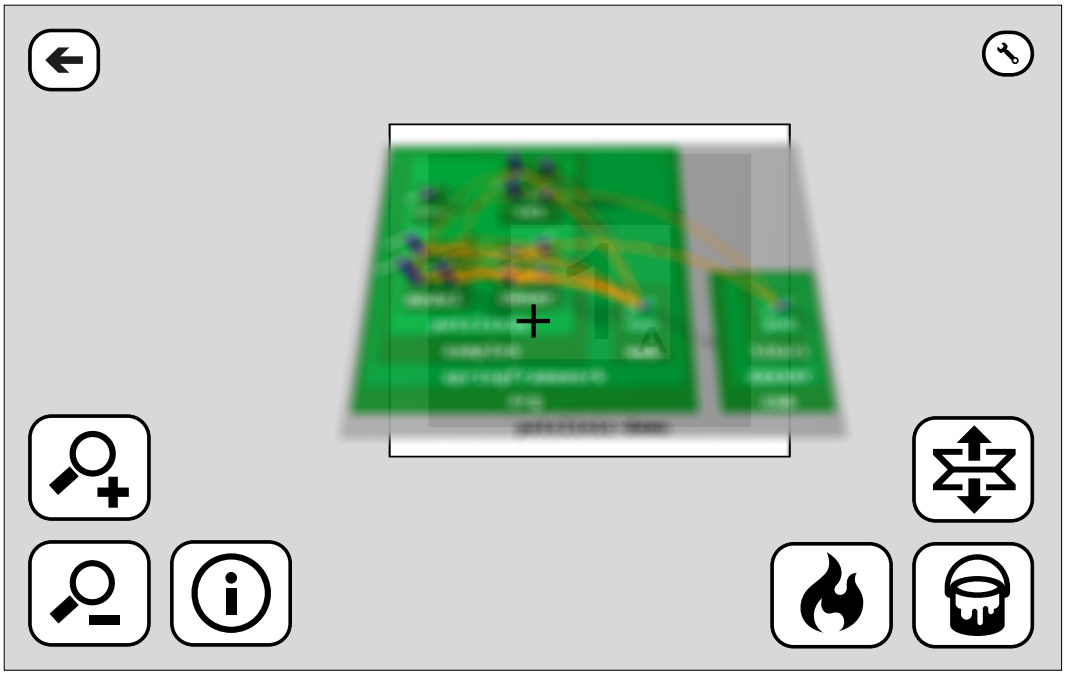


**Figure 5.2.** The concept for the user interface. A crosshair in the middle is used to aim at the desired entity within models which are placed on a marker. Buttons in the bottom left and right corner allow for various interactions. A button in the top right corner enables a user to exit the visualization while a button in the top right corner opens a window for settings.

the number of five markers for applications to be sufficient for the time being. If all markers are occupied, however, another application would have to be closed first.

For interaction with the software models, mobile devices do not have a mouse as an input device, but only a touchscreen. Tapping with a finger can be quite inaccurate, which is why we assume this to be unsuitable for interaction with small classes and closely packed communication lines. Furthermore, regular touch inputs could be impeded by the fact that only one hand is then available to hold the mobile device. Depending on the size, this could cause it to wobble, and the camera image with markers and models located on the markers would also move accordingly. We, therefore, aim to have users hold their device steadily with two hands and still be able to interact well with the software models. At last, we envision to provide the user with various settings to configure the visualization. For example, the size and spacing of the buttons should be configurable such that they can be adapted to different screen sizes and user preferences.

A first draft of the visualization can be seen in Figure 5.2. It can be seen that models are projected onto recognized markers. In the middle of the screen, we plan to use a crosshair, which allows the targeting of entities and prevents the models from being obscured by the

user's fingers. Buttons for interaction are placed in the lower left and lower right corners so that they can be easily operated with the thumbs during two-handed use.

As features for interaction, we would like to adopt the features known from the frontend for opening and closing applications and packages as well as the highlighting feature. A toggleable heat map should also provide a quick overview of properties of classes, e.g. regarding their number of outgoing method calls. Since applications can be very large, making it difficult to read class names, for example, it should be possible to adjust the size of the 3D models. Finally, the popups known from the frontend should also be added to display information.

## 5.5 Incorporation with existing Implementation

To ensure maintainability and compatibility of the developed approach, the later implementation should integrate with the existing implementation. Since our approach for the visualization has the most similarities with the VR approach of ExplorViz, our approach should be added to the existing VR extension. Effectively, this would turn the VR extension into an XR extension.

In terms of collaboration, the existing VR service should also be used. This service can be employed for session management. In addition, the VR service can synchronize data about the opened applications, the state of packages within applications, and the information about currently highlighted entities. Additions to the VR service may need to be made here to ensure proper synchronization for our approach or enable new features to be used collaboratively.

## 5.6 Software Technologies

In this section, we discuss our choice of software technologies.

WebXR is an API which enables the use of AR and VR devices for web browsers. ExplorViz employs WebXR already for its VR extension. However, WebXR is not yet supported by all modern browsers. For example, the mobile Safari browser does not support WebXR. As Safari is the only browser on iOS devices which is allowed to use the camera, this would interfere with our goal to develop an approach for a wide variety of mobile devices. In addition, we rely on markers for the placement of the software models. WebXR does not provide an implementation for the recognition of markers. Moreover, since we are not using specialized hardware for AR, WebXR does not offer features that we could make use of.

As opposed to WebXR, AR.js is a Javascript library which offers the detection of markers in live camera feeds and brings support for the use of three.js. Therefore, AR.js provides the feature set that is required by us and is supported by all modern browsers, including mobile browsers. The use of AR.js is presented in the upcoming chapter.

Chapter 6

# Implementation

In this chapter, we present the implementation of our approach for collaborative program comprehension. The presented implementation reflects the state which is used for our case study. Minor adaptions which result from a preliminary pilot study are already incorporated in the upcoming sections.

## 6.1 AR.js

AR.js is a Javascript library for AR on the web. For managing such a dependency we prefer to use *npm*[1], a package manager for Javascript applications. However, there does not exist an up-to-date version of AR.js for npm. Therefore, we include the current version of AR.js as a Javascript file in the frontend of ExplorViz. AR.js is provided in different variants. We choose the variant which can be used in conjunction with three.js. Another variant is meant to be used with *A-Frame*[2], a web framework for building AR and VR environments.

AR.js provides three main classes which need to be initialized in order to make use of AR.js in conjunction with three.js. These classes are the *ArToolkitContext*, *ArMarkerControls*, and *ArToolkitSource*.

**Listing 6.1.** Initialization of ArToolkitContext

```
1  let arToolkitContext = new THREEx.ArToolkitContext({
2     detectionMode: 'mono',
3     cameraParametersUrl,
4  });
```

The initialization of the *ArToolkitContext* class is presented in Listing 6.1. The ArToolkit-Context is the main engine which detects a marker within the desired images, e.g. a live camera feed. Since we rely on black and white markers, we set the detection mode to 'mono' (line 2). The use of colored markers is also possible but the accuracy of the marker recognition is more sensitive to the encountered lighting conditions as these can distort the colors. In addition to the detection mode, we need to pass the path to a file which contains parameters for the employed camera (line 3). This is a binary file that contains

[1] https://www.npmjs.com/

[2] https://aframe.io/

technical data about a camera like the aspect ratio or focal length. We include such a file for a 4:3 and a 16:9 aspect ratio and achieve satisfying results for the visualization. The use of a file which contains mismatched camera parameters could lead to visual distortions or misplacements of the 3D models.

**Listing 6.2.** Initialization of ArToolkitSource

```
1  let arToolkitSource = new THREEx.ArToolkitSource({
2       sourceType: 'webcam',
3     sourceWidth: width,
4     sourceHeight: height,
5  });
```

The initialization of the *ArToolkitSource* class is displayed in Listing 6.2. The ArToolkitSource manages the image source which should be used for the recognition of markers. As we are interested in a live camera feed, we use the images which are produced by the device's camera or webcam (line 2). However, the use of image or video files is also an option. Furthermore, we specify the desired width and height (line 3 and 4). Notably, it is not possible to request the actual camera resolution of a device through a web browser due to privacy concerns. The provided width and height (in pixels) can only inform a browser about the ideal camera resolution for the given application. It could be that a lower resolution is provided if the device's camera does not support the desired resolution. Therefore, we use a resolution of 640 x 480 pixels as a default camera resolution. This resolution can be met by all modern camera systems and reduces the computational load for the detection of markers.

**Listing 6.3.** Initialization of ArMarkerControls

```
1  let arMarkerControls = new THREEx.ArMarkerControls(
2  arToolkitContext,
3  landscapeContainerObject3D,
4  { type: 'pattern',
5     patternUrl: 'ar_data/pattern-angular_L_thick.patt'}
6  );
```

The initialization of the *ArMarkerControls* class is presented in Listing 6.3. The ArMarkerControls class is responsible for the correct alignment of 3D models on top of a recognized marker. Therefore, it relies on the ArToolkitContext (line 2) and expects a three.js object which it centers on the marker (line 3). Whenever a marker is recognized within the provided image, the specified three.js object is set to be visible. Thereby, also all child objects of that three.js object, e.g. the 3D landscape model, become visible. When a marker is not recognized anymore, the specified three.js object is turned invisible again. In addition to a three.js object, ArMarkerControls needs the path to a pattern file which contains data for the recognition of the corresponding marker (line 4 and 5). The contents

of the pattern files and the process to generate them is explained in more detail in the upcoming section.

## 6.2 Markers

In this section, we introduce the process of creating and detecting a marker. Thereafter, we use the gathered knowledge to report about the design process which resulted in our final marker design.

AR.js provides an online tool[3] that can be used to transform images into corresponding marker images and pattern files. For the resulting images, the uploaded image is placed within a thick black border, which characterizes markers. The resulting pattern files can be added to the frontend of ExplorViz and are needed to initialize the marker controls.

Using our concept for markers as a starting point, we thoroughly tested different marker designs. It turns out that the envisioned marker design (see Figure 5.1) has drawbacks in terms of recognition accuracy. Even though the first marker design is recognized in most cases, bending the marker or non-optimal lighting conditions had a great impact on the correct placement of the 3D models. During our regular tests, models had a tendency to be missing even though the marker was within the camera image. Furthermore, it could happen that models were visible in some seemingly arbitrary location of the image even though no marker was visible.

The unwanted behavior led us to investigate the cause of this issue. While taking a look at the content of the pattern files we recognized that the original images for a marker are transformed into 16 times 16 numerical values which range from 0 to 255. In other words, the marker recognition is based on recognizing a low-resolution image with gray values which range from 0 (black) to 255 (white). This approach is very similar to that of SkyscrapAR (see Section 4.2). If the resolution of input images is larger than 16 times 16, the average color value is calculated for the corresponding area. Therefore, an input image for a marker which only contains black and white pixels could result in a marker with many different gray values.

We assumed that the recognition accuracy could be improved by providing an input image whose pixels are a better match for the resulting pattern file. Thus, we went through several design iterations to come up with a marker design which consists of black and white elements, is aligned in a 16 by 16 grid, and still is semantically meaningful. We introduced a horizontal at the bottom of each marker to clarify the intended orientation. Furthermore, we kept the letter "L" to represent the landscape marker. In contrast to application markers, the font of the landscape marker is thicker to indicate the importance of that marker, as every software exploration in ExplorViz starts with the landscape model. For application markers, we retained the approach for numbering those markers but neglected to include additional visual hints since the landscape marker already stands out. An example of the

[3] https://jeromeetienne.github.io/AR.js/three.js/examples/marker-training/examples/generator.html

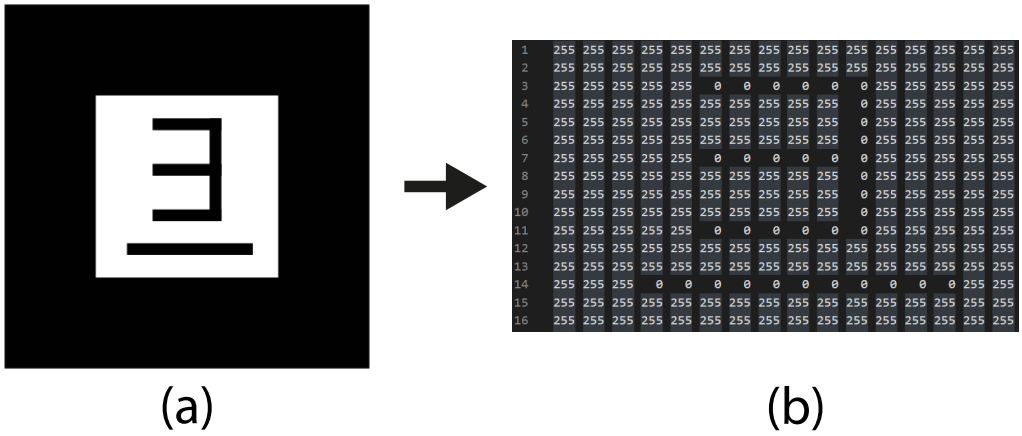



**Figure 6.1.** A marker for the third application which has been opened (a) and a snippet from its corresponding pattern file (b).

new marker design and its color values in the corresponding pattern file can be seen in Figure 6.1. The design of all markers is presented in Appendix 8.2.

We tested the new marker design and noticed that the beforementioned issues are alleviated. In our experience, it does not happen anymore that models are placed on the wrong marker or placed when no marker is present at all. Still, markers may not be recognized correctly due to poor printing quality, especially if colors are not printed accurately.

Regarding the paper on which the markers should preferably be printed, we use matt paper with a weight of 300 grams per square meter in accordance with our concept for markers (see Section 5.3.2). In the upcoming section, we present how the markers are combined with the 3D software models and a user interface.

## 6.3 Visualization

In this section, we give an overview on the developed visualization and the user interface which accommodates it. For features that are added in addition to the core implementation of ExplorViz, we refer to upcoming sections.

Figure 6.2 displays an example of the developed visualization. It shows three markers which are placed on a table. The landscape model is always displayed on the marker which contains the letter "L". Both landscape (Figure 6.2 - A) and application models (Figure 6.2 - D) can be set to different transparency values such that the underlying marker is readable and virtual and real environments blend together. Applications which include transparent


**Figure 6.2.** Overview of the developed visualization. Three printed markers are placed on a table such that a landscape model (A), a highlighted application (B) with additional information (C), and a second application (D) are visualized. A centered crosshair (E) is the reference point for several interactions with the models which can be triggered via buttons in the bottom left (E) and bottom right (F) corner of the screen. Settings (H) can be accessed to customize the visualization.

entities for other reasons, e.g. because of a highlighted package, are temporarily set to opaque (Figure 6.2 - B). Besides the transparency, we included various customization options which can be accessed via a designated settings button (Figure 6.2 - H). We give an overview of all available settings in Section 6.7.

In general, the developed visualization closely resembles the draft for the visualization which we envisioned in our concept (Figure 5.2). We removed the button to return to the 2D landscape view. A context menu can be accessed through a long press on the camera image or 3D models to access this and other features which are not frequently used. Through this, we freed up additional screen space for the visualization and reduced accidental button presses. The elements on the top of the screen, including the logo of ExplorViz and a blue identifier of the currently selected landscape, are carried over from the default visualization of ExplorViz as those take up little space and are unintrusive. Furthermore,

we deviated from our concept by not employing the classic crosshair design and updating it to a hollow circle (Figure 5.2 - E). This has the advantage that the color of the currently targeted entity is still visible in the center and not covered. Even though the form differs, we keep the name crosshair for this visual selection aid. In the given figure, the crosshair could seem small as the screenshot was taken on a tablet computer with a screen diagonal of 12.9 inches. It is important to us that the user interface is suitable for most screen sizes and therefore elements of the user interface exhibit different sizes depending on the used device.

In addition to the crosshair, we adapted the icon design, arrangement, and features of some buttons. The scaling of software models is now accomplished through touch gestures (see Section 6.4). The remaining button with a magnifier (Figure 5.2 - F) is repurposed to enable a temporary enlargement of a small area around the crosshair. This zoom feature is described in more detail in Section 6.5. Beside the button for zooming, there is a button to toggle the heat map. This feature is presented in more detail in Section 6.6. The remaining button in the bottom left corner, called info button, is used to call up additional information about an entity. When activated, popups (Figure 5.2 - C) that contain data about the targeted model, e.g. a class, package, or application, can be opened. The popups are inherited from the frontend's core implementation. However, we added a wrapper around the underlying code which enables us to initially place popups above the crosshair and allow a user to drag and place them freely on the screen. By default, at most one popup can be opened at a time but popups which have been dragged by the user are not removed by subsequent clicks on the info button. Still, whenever an opened popup is of no interest anymore, it can be individually closed via a blue button above the popup. In addition, all popups can be closed at once through a long press of the info button or the selection of a corresponding entry within the context menu.

The remaining buttons of the visualization are placed in the bottom right corner of the screen (Figure 5.2 - G). The button with two arrows enables the main interaction with the visualized models. Through this button, target applications and packages can be opened or closed. Therefore, it resembles the features which are accomplished by a double click within the default visualization. The button with a colored paintbrush icon is used to highlight packages, classes, or class communication in the depicted color. It, therefore, inherits the features which are accomplished through a single mouse click in the default visualization. The last remaining button contains an icon which should resemble a laser. It triggers the temporary display of a colored sphere. Thus, users can ping at a point within an application or landscape. This collaborative feature is presented in more detail in Section 6.8.2.

## 6.4 Touch Gestures

In our concept, we planned to realize the interaction with the displayed 3D models mainly through the use of HTML buttons. As these are placed in the bottom right and left corner,

it would enable a user to hold the tablet computer steadily with two hands during the use of our approach. However, during the development of our approach, the number of desirable features increased such that the addition of more buttons would be required. Due to the limited screen space, we want to avoid this. Thus, we add the (multi-)touch gestures pinch, rotate, and pan to our implementation. The gestures can be used to change the scaling, rotation and position of the models which are targeted by the crosshair.

### 6.4.1 Pinch

We noticed that scaling the landscape or application models through a button press is not practical. First of all, scaling a model like this can take up an extended amount of time. Buttons can only register the number of performed touch actions or the duration of a button press and therefore a user would need to click the button multiple times or press it for quite a long time to change the size of a model by multiple magnitudes. Scaling a model faster would inevitably make it harder for a user to make small adjustments to the scale of a model.

As tablet computers are reliant on touchscreens for user inputs, we decided to incorporate touch gestures into our implementation. The frontend of ExplorViz already uses Hammer.JS, a library to detect and process touch gestures, in its core implementation. However, ExplorViz only uses the *tap recognizer* of Hammer.js which can detect the number of consecutive taps or clicks which a user performed. The tap recognizer of ExplorViz' core implementation is configured such that a double-click within a certain timeframe only triggers a corresponding event for a double-tap but no event for a single-tap. Thereby, ExplorViz realizes double-click actions like opening a sofware package without triggering a single-click action which could for example result in an unintended highlighting of a package.

We, on the other hand, want to realize the scaling of the models through touch gestures. Since scaling a model can have the same visual effect as changing the zoom of a camera, we employ the commonly used pinch-to-zoom gesture. For a pinch-to-zoom gesture, a user places two fingers on a touchscreen. The gesture is then performed by changing the distance between the two fingers, i.e. moving the fingers apart or closer together. The change in distance between the used fingers can be measured and for example, be used to change the size of an object accordingly.

Hammer.js provides a pinch recognizer. This recognizer can trigger an event when the pinch gesture is performed and contains a numerical value which represents the distance of the two corresponding fingers. We apply some calculations to retrieve a percentage which we apply directly to the scale of the targeted model. For example, by doubling the distance between the two fingers, the targeted model is doubled in size with respect to all three dimensions. Moving the fingers more closely together has the inverse effect.

Thus, a user can, depending on the initial placement of fingers, make both small and large changes to a model's scale very rapidly. For example, initially placing the fingers closely together and then increasing the distance between them can dramatically increase

a model's size. This is a useful feature for devices with small screens. On other hand, an initial finger placement with a greater distance causes only smaller changes to the scale of a model with respect to the absolute difference of the finger's distance.

### 6.4.2 Rotate

In our concept, we do not include the option for rotating a model on a marker since the model can be rotated intuitively together with the underlying marker. However, through early tests, we recognized that switching between the tablet computer and a marker could be disruptive. Thus, we add the option to rotate models through touch gestures and let the user decide which way of interaction suits him or her the best.

Hammer.JS also provides a recognizer for rotation gestures. Therefore, a user can place two fingers on the screen and turn those in a circular motion. As the pinch-to-zoom gesture also requires two fingers, the two gestures can be combined seamlessly for improved usability. Additionally, we decided against the employment of gestures which include three or more fingers as those are hard to perform on small screens. Furthermore, on some operating systems there exist gestures for three fingers which can take precedence over our implementation.

### 6.4.3 Pan

Since the size of applications depends on the number of packages and classes, there is no upper limit for the size of the displayed models. Thus, scaling models can be crucial but could result in models which do not fit on the screen. The part of the models which is currently visible could be changed by moving the tablet computer and therethrough change the position of the marker in the current image of the camera. However, this could also cause a marker to be outside the camera's field of view. As a result, we incorporate a pan gesture to move landscape and application models within the plane which is defined by the marker.

Hammer.JS provides a pan recognizer. As the movement of objects is usually achieved with one finger and we since dragging a popup is also achieved by a pan gesture with one finger, we enable the movement of models by placing and moving one finger on the screen. Just as the other touch gestures, the pan gesture can be performed anywhere on the screen (excluding buttons) such that the users can perform the gesture beside the buttons. Therefore, it is not necessary for a user to regularly change the position of his or her hands and travel large distances on the screen just to target the desired model.

The scaling, rotation, and position of software models can be reset via an entry within the context menu.


**Figure 6.3.** The visualization of an application. The zoom feature is activated (A). Thus, a rectangular region around the crosshair is magnified (B).

## 6.5 Zoom Feature

Scaling through touch gestures, as presented in the previous section, allows a user to change the overall size of a model. However, when a user wants solely to increase the readability of some text or more closely inspect the crosshair to determine if the wanted entity is targeted, scaling the whole model is impractical. By increasing the scale of a model one can lose the overview of its dimensions and overall structure.

Therefore, we introduce a zoom feature which enables a user to temporarily increase the size of the center part of the visualization. The zoom feature can be toggled by the zoom button (Figure 6.3 - A), whereby a green color of the button indicates that the zoom is activated. As a result, the crosshair and the parts of the targeted model which surround the crosshair are visually enlarged. By default, we set the magnification such that the displayed parts are three times larger than usual. We assume this to be a suitable magnification for the selection of classes or communication. However, the magnification can also be configured in the settings.

**Listing 6.4.** Rendering of the zoomed Area

```
1 renderer.setScissorTest(true);
2
3 renderer.setViewport(zoomPos.x, zoomPos.y, zoomSize.x, zoomSize.y);
4 renderer.setScissor(zoomPos.x, zoomPos.y, zoomSize.x, zoomSize.y);
5
6 renderer.render(scene, this.zoomCamera);
7
8 renderer.setScissorTest(false);
```

For the implementation, we first need to calculate the position and size of the zoomed region within the overall canvas which depends on the desired magnification. To render the desired portion of the scene, three.js brings features to render another image section within the default image. The core of this implementation is presented in Listing 6.4. First, the renderer is instructed to only render within a restricted region by activating the scissor test (line). Following, the position and size of the zoomed region is declared (lines 3 and 4). At last, the scene containing all 3D models is rendered (line 6). As opposed to the rendering of the image for the remaining scene, a camera which is configured in accordance with the desired magnification is used.

## 6.6 Heat Map

Especially for small screen sizes, a heat map can help to improve the overview of the displayed entities. The heat map of ExplorViz, as introduced in the concept, provides a coloring for classes which is applied with respect to different metrics.

However, we could not simply adopt the existing implementation for the heat map and use it for our visualization approach. The existing heat map implementation is based on a deprecated data model. It also relies on a backend service which calculates and persists data for different metrics. The existing service cannot be included as is in the current backend infrastructure.

As the frontend is already capable and responsible for processing incoming trace data, we opt to update the existing implementation and to introduce a working heatmap visualization which resides in the frontend. The foundation for the heat map visualization are metrics. We calculate those concurrently in a designated Web Worker. In addition to the already calculated count of newly created class instances, we introduce metrics for incoming, outgoing, and overall requests. Hereby requests for a class are method calls which originate from or target any object which is an instance of that class.

The resulting visualization is displayed in Figure 6.4. The heat map visualization can be toggled by the click of a button (Figure 6.4 - A) which turns green when the heat map is active. The components and classes of the targeted applications become transparent. Thereby classes and their corresponding heat map colors, which are projected on the

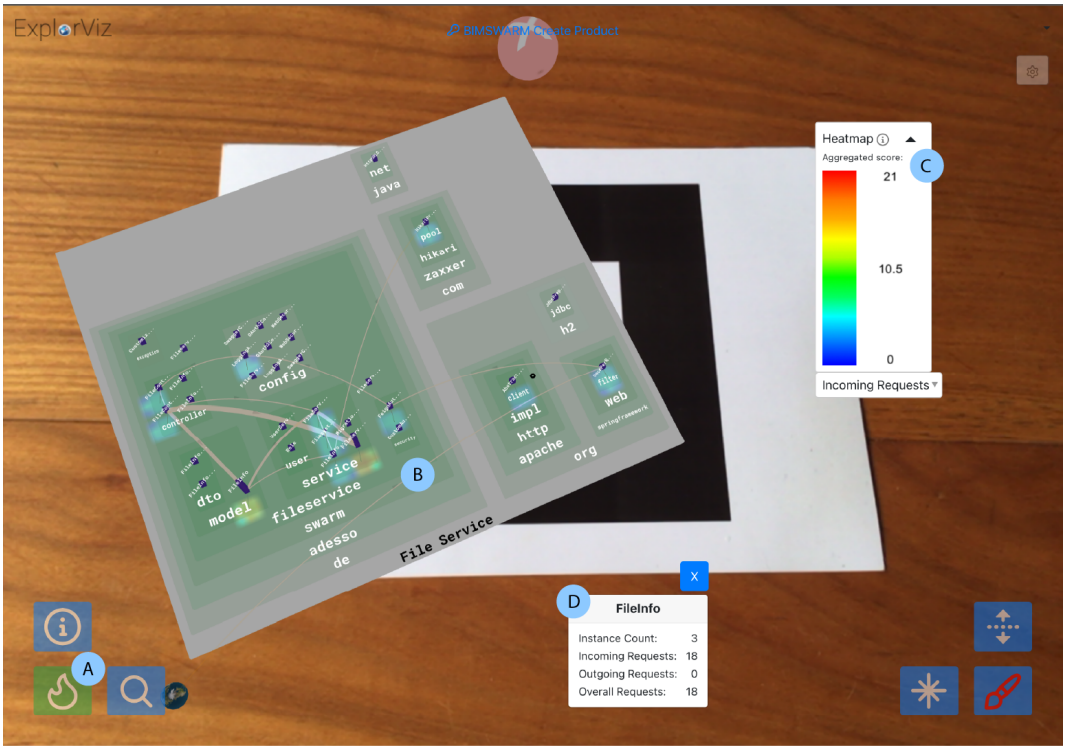



**Figure 6.4.** An example of the heat map visualization. The heat map is toggled via a button (A) and projected on the foundation of an application (B). A collapsable color legend with a metrics selector (C) and precise values about the metrics in class popups (D) accompany the heat map.

opaque foundation, are visible (Figure 6.4 - B). Small black lines connect classes and their corresponding colored area on the foundation such that their association can be accessed independently of the current camera perspective.

When the heat map is activated, a labeled legend (Figure 6.4 - C) is opened. It can be dragged and placed freely on the screen, just like popups. In addition, a small triangle can be clicked to collapse the legend such that it takes up less space on small screens. Under the legend is a selector which shows the currently active metric and allows to switch seamlessly between the available metrics.

The data values for the metrics are calculated when the heat map is activated. Since the colors only give a rough estimate about the absolute value of a metric, popups for classes contain the absolute values of the available metrics (Figure 6.4 - D).

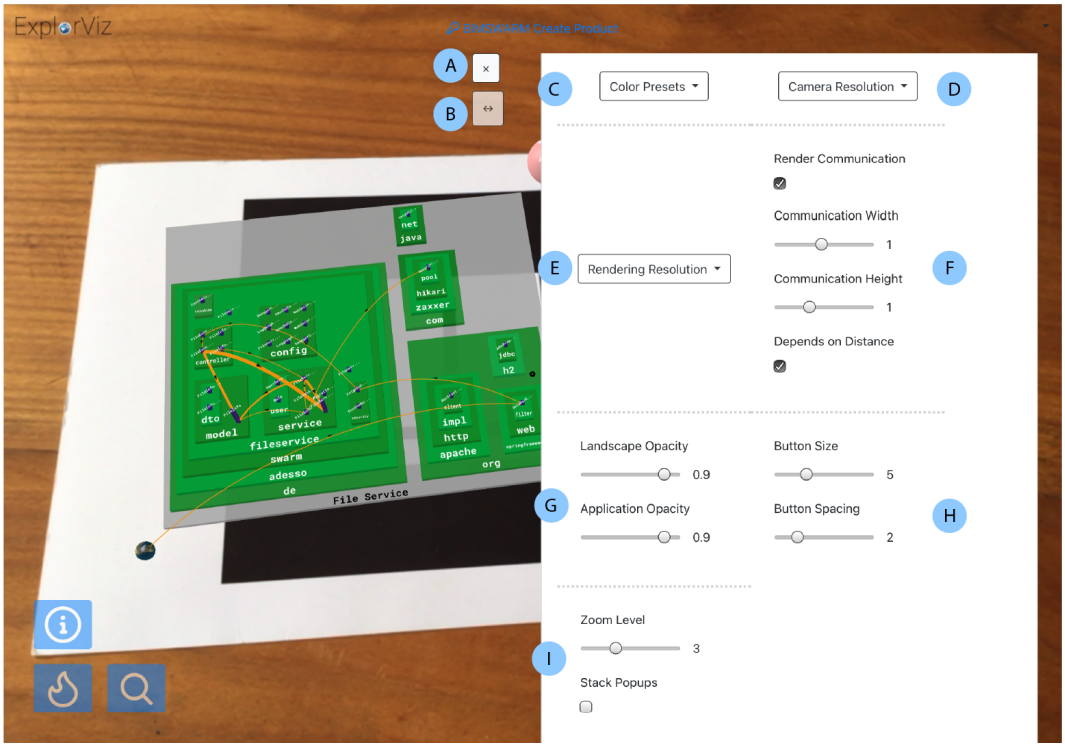



**Figure 6.5.** An overview of the available settings (C to I). The settings panel can be closed (A) and horizontally resized (B).

## 6.7 Settings

In order to support a variety of devices in the best possible way, we implement a number of configuration options, which we present below.

As soon as the user clicks on a button to open the settings, a sidebar opens and reveals the configuration options (Figure 6.5). The sidebar can be varied in width via a button (Figure 6.5 - B). In the options, the color scheme can be adjusted (Figure 6.5 - C), e.g. if the default color scheme does not exhibit good contrasts on a user's device. In addition to the default color scheme, color schemes with blue, dark, as well as pastel colors are provided.

To calibrate quality and performance, a user can adjust the resolution of the camera and the resolution of the rendered models (Figure 6.5 D - E). By default, the models are displayed in the best quality, whereas the camera is set to a low resolution to save resources. However, high-definition resolutions are also available.

Since a large number of communication lines can be difficult to tell apart, we have introduced some options to customize their visualization (Figure 6.5 - F). The communication

can be completely hidden, if only structural data is of interest. In addition, the thickness of the communication lines can be configured in order to make differences between the various communication lines more obvious. The height of the communication specifies how great the curvature of the rendered lines should be. A height of 0 corresponds to a straight communication line. In addition, it can be configured whether the height of the communication depends on the distance of the involved classes, such that far away classes are connected by a higher communication line.

In order not to lose sight of the markers and to improve immersion, both the landscape model and the application models are slightly transparent. The degree of transparency can be adjusted to the user's preferences (Figure 6.5 - G). Next, the size and spacing of the buttons can be adjusted (Figure 6.5 - H). The buttons already scale with the screen size, however, especially on small devices, it may be helpful to further reduce the size of the buttons to free up space for the visualization.

Lastly, we offer the option to adjust the magnification of the zoom feature as well as the option to let popups stack (Figure 6.5 - I). By default, one popup is displayed at a time. Popups can be moved at will, after which they are not automatically removed again. However, the displayed option allows the user to call up several popups subsequently and then view them one after the other.

All settings made via the sidebar are also updated in realtime within the visualization to improve usability. The sidebar does also provide a component for collaboration that we present in the upcoming section.

## 6.8 Collaboration

In this section, we present the implementation of collaborative features.

The basis for the collaborative software exploration is the synchronization of the displayed models such that users can exchange their thoughts about entities which are currently visible to them. The VR extension for ExplorViz makes extensive use of collaborative features. There exists a designated backend application, called VR Service, to persist and synchronize the current state of the visualization amongst different users. It is our goal to make use of existing technologies and implementations. Therefore, we employ the VR Service to synchronize various aspects of the visualization for collaborative use.

### 6.8.1 Collaboration Interface

The VR extension of ExplorViz uses custom canvas-based menus which are placed within the 3D scene of three.js to allow users to host and join a collaborative session. These menus are optimized for use with VR controllers. We, on the other hand, opt for the use of native HTML elements and therefore developed a component for collaboration. The component is accessible by a click on the settings icon which opens a sidebar.

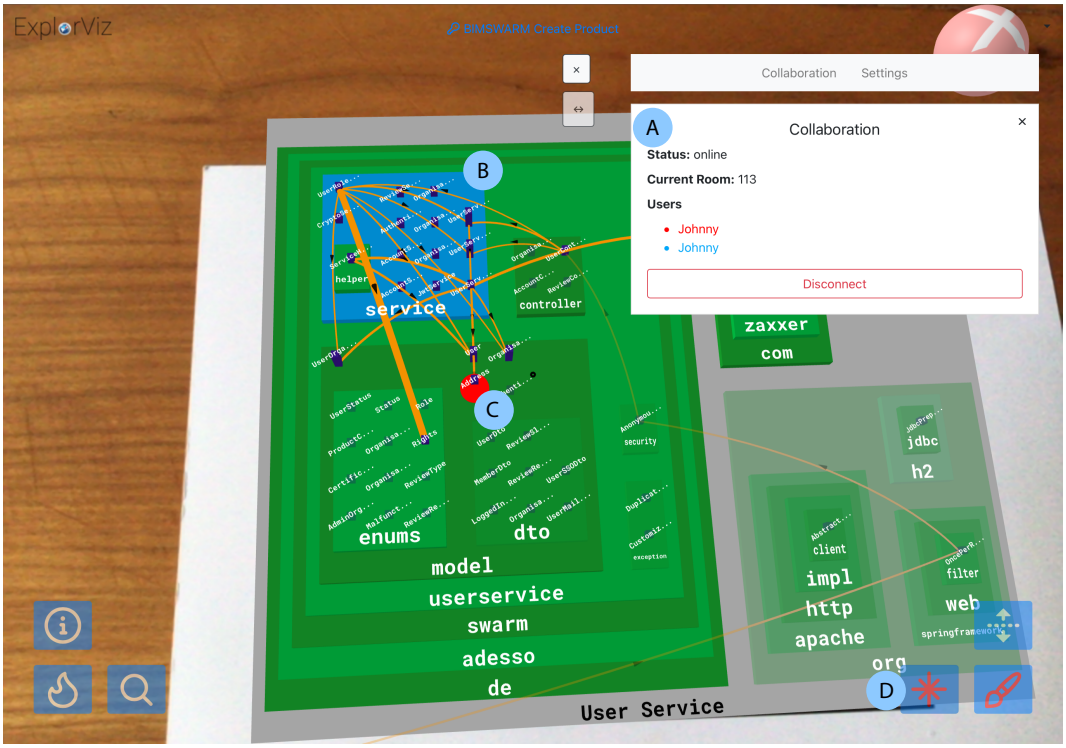



**Figure 6.6.** A Snippet of an application model during the collaboration of two users (A). One user highlighted a package (B) while another user pinged a class (C) by clicking the corresponding button (D).

The collaboration component enables users to host a collaborative session, referred to as a room, or join an existing room. Available rooms are assigned with a unique number. When a user joins a room, the collaboration component displays the current room number and a list of all users who are currently assigned to that room (Figure 6.6 - A). A unique color is assigned to every user by the VR Service which we display in the user list. As the assigned color is used for highlighting, the icon of the highlighting button also does represent the current color of a user. The highlighting is synchronized amongst users (Figure 6.6 - B) and can be used on packages, classes, and communication between classes.

### 6.8.2 Ping Feature

Highlighting is intended for the analysis of communication between packages or classes. Also, only one entity can be highlighted per application such that a subsequent highlighting action does override any existing highlighting within an application. We also noticed in a conducted pilot study that users who use our approach collaboratively might talk about

a certain package or class which exists in more than one application. For example, the "model" package is a common name for a package throughout various applications. Thus, there exists the possibility of confusion.

To facilitate the collaboration of users, we introduce the ping feature. A ping is represented by a colored sphere which bears a resemblance to the visual effect of a laser pointer. Users can activate the temporary ping via the corresponding button (Figure 6.6 - D). The ping is placed just above the entity which is targeted with the crosshair. It is displayed for two seconds for each user whereby pings of multiple users could be present at once.

## 6.9 Deployment

In this section, we describe the necessary steps to test our approach during the development and the steps that follow in order to deploy the developed approach.

The backend of ExplorViz does provide Docker containers such that the services can be started easily. There do also exist various sample applications with corresponding configurations for inspectIT Ocelot such that live monitoring of an application can be established for development purposes. In addition, most of the implemented features can be tested on a local desktop computer since our visualization approach does only require printed markers, a modern web browser, and a camera.

However, testing the AR visualization locally on a mobile device turned out to be more challenging. As a first step, the addresses of the backend services are set to local IP addresses as opposed to *localhost* within the frontend. This is necessary since a mobile device which accesses the frontend would otherwise expect that the backend services of ExplorViz are executed on the mobile device itself. With this configuration, mobile devices can use the regular visualization of ExplorViz. However, it is not possible to use our approach like this. In order to use the camera within a browser, WebRTC requires a secure connection with the only exception being the localhost domain.

To establish a secure connection locally, we generate a self-signed certificate for the localhost domain. Since modern web browsers do not allow mixed content, i.e. the combination of secure and insecure communication on websites, all connections from and to the frontend need to be secured by that certificate. This could be achieved by supplying every backend service of ExplorViz with a configuration for a secure connection. However, we choose to use nginx as a reverse proxy. With nginx, requests from the frontend can be sent securely to nginx which in turn can forward the requests to the desired backend service. The connection between nginx and the backend services do not need to be secure. This is a preferable solution as long as all microservices of ExplorViz and nginx are deployed on the same computer or server.

With nginx and the self-signed certificate in place, we can test our implementation locally on an Android smartphone with Chrome as a mobile browser. Chrome displays a warning since the certificate is only self-signed but allows a user to enter the website after a confirmation. In addition, the camera can be accessed through WebRTC. However, we

observe that this solution is not viable for iOS devices. The mobile *Safari*[4] browser does not allow users to enter websites with an untrusted certificate. The use of Chrome is no viable solution either since Safari is the only browser on iOS which is allowed to use the device's camera. It is possible to add trusted root certificates to iOS. However, certificates cover a domain and no IP address. Thus, the only way to test our developed approach locally on iOS devices would require the installation of a local DNS server. The DNS server could route requests for the desired domain to the local IP address of the desktop computer on which ExplorViz is executed. This setup is impractical for us and thus we stick with testing our implementation locally on a computer and an Android device.

To test our approach with iOS devices and for the evaluation of our approach, a deployment of ExplorViz on a publicly accessible server with a secure connection is required. During development and also for the later evaluation, we are granted access to a server of adesso. The server uses *CentOS*[5], a linux distribution, as an operating system. Since ExplorViz can be executed within Docker containers, the deployment process is straightforward. Just like for the local deployment, we use nginx as a reverse proxy on the server of adesso. Only the configuration of nginx needs to be adapted because a development version of BIMSWARM is also executed on the given server and already uses nginx. In the end, both BIMSWARM and ExplorViz are publicly accessible and use trusted certificates for secure communication from and to the frontend.

As the last step for the deployment, we need to combine our approach for software visualization in AR with BIMSWARM. In the upcoming section, we describe the necessary steps for the instrumentation which collects runtime data of BIMSWARM and sends it to the backend infrastructure of ExplorViz for further processing such that the resulting data can be visualized in with our visualization approach.

## 6.10 Instrumentation of BIMSWARM

To execute BIMSWARM on our local machine, we build its frontend and configured nginx such that incoming requests on port 80 are redirected to the frontend. Additionally, nginx is responsible for routing requests which originate in the frontend to the corresponding backend services. For the backend there exist gradle tasks to start the services. First, the service which employs Eureka is started such that services can register with Eureka on their startup. Then, the Gateway service and PCSSO service are started to enable the distribution of communication and authorization of service requests. At last, the remaining services are started in an arbitrary order.

As BIMSWARM employs a microservice architecture with services which are written in Java, we can use inspectIT Ocelot as a Java agent to extract runtime data. The Java Virtual Machine (JVM) enables agents to attach themselves to a running process of an application. Therefore, we intended to use this feature and add the inspecIT Ocelot agent to the already

[4] https://www.apple.com/safari/

[5] https://www.centos.org/

running services of BIMSWARM. However, this approach has few advantages since there is no option to remove the previously added agent from the instrumented process. Most importantly, attaching inspectIT Ocelot as a Java agent during the runtime of a Java process yielded an exception much more frequently than adding the agent during the startup of the same application.

Therefore, as recommended by the documentation[6] of inspectIT Ocelot, we started it together with the application which we want to instrument. This can be achieved with the following command:

**Listing 6.5.** Execution of the Product Service Application with inspectIT Ocelot

```
1  java -Dinspectit.config.file-based.path="./" -javaagent:inspectit-ocelot-agent
       -1.8.1.jar -jar productservice.jar
```

For this command execution, it is expected that a configuration file, the inspectIT Ocelot agent in version 1.8.1, and the JAR of the product service are in the current directory.

**Listing 6.6.** Excerpt of the Instrumentation for the Product Service

```
1  scopes:
2  s_erroneous_instrumentation:
3  type:
4  name: de.adesso.swarm.productservice.feign
5  matcher-mode: STARTS_WITH_IGNORE_CASE
6  s_allClasses:
7  type:
8  name: de.adesso.swarm.productservice
9  matcher-mode: STARTS_WITH_IGNORE_CASE
10 exclude:
11 s_erroneous_instrumentation: true
```

The configuration file, called *inspectit.yml* by default, is a *YAML*[7] file which is used to specify which data shall be collected and where the data is to be sent. For the most part, we adapt the configuration files which ExplorViz already uses for other Java applications. However, we need to specify a custom scope. Instrumentation without restrictions would also include method calls of classes from employed libraries. Since we are interested in the program behavior of BIMSWARM, we naturally narrowed the scope down to classes which are developed and maintained by adesso. Doing so resulted in inconclusive runtime errors as soon as we triggered actions through the frontend. We used the approach of binary search, e.g. by including only half of the packages in our scope at first, to isolate the origin of the errors. Finally, we could observe that the instrumentation of classes which use *Feign*[8], a Java to HTTP client binder, results in the previously noticed errors. We suspect

[6] https://inspectit.github.io/inspectit-ocelot/docs/getting-started/installation

[7] https://yaml.org/

[8] https://github.com/OpenFeign/feign

that this could be caused by the extensive use of annotations in Feign, including custom Feign and *JAX-RS*[9] annotations, which could interfere with the code which is added by the Java agent.

Concluding, we excluded the classes which use Feign from our instrumentation. A resulting scope for the product service can be seen in Listing 6.6. Here, line two to five describe the scope which causes errors during instrumentation. This scope is excluded from the overall scope, which is specified in the remaining lines of the Listing 6.6. The scope of the instrumentation is representative, as we included all packages which are developed by adesso and excluded the feign package for those services, too.

[9] https://jakarta.ee/specifications/restful-ws/3.0/

Chapter 7

# Evaluation

In this chapter, we present the evaluation of the implemented approach for collaborative program comprehension by means of a case study.

## 7.1 Goals

In this section, we describe our goals for the evaluation of the developed visualization approach for program comprehension. Collaborative program comprehension through an AR visualization for mobile devices, i.e. tablet computers and smartphones, is a new field of research. It is our goal to ascertain which potentials and drawbacks such a visualization has. We specify our goals in terms of four research questions:

- Is the employed visualization suitable for mobile devices?
- Are the implemented features suitable for mobile devices in terms of usability?
- Is the developed approach suitable to solve tasks in the context of dynamic program analysis?
- Is the developed approach suitable to work collaboratively?

## 7.2 Methodology

In order to assess the potentials and drawbacks of our approach, we conduct a case study. Our study design is inspired by the experiments of Hansen [2018], König [2018], and Brück [2020] about the collaborative VR approach of ExplorViz. Notably, the collaborative nature of our approach also requires that two probands can evaluate the developed approach together. Hereby we can evaluate whether the implemented set of features and the use of mobile devices are suitable for collaborative work.

Due to the COVID-19 pandemic, we cannot design and prepare an experiment setup which requires probands to undertake the case study in a designated lab environment. Additionally, the collaborative use of mobile devices for program comprehension is a new field of research. Therefore, our study is qualitative in nature. We strive to find answers to our proposed research questions and collect substantial feedback which can be a foundation

for further development of our approach. For the communication and interactions with probands during the study, we are guided by established best practices of Hove and Anda [2005].

## 7.3 Experiment

We evaluate our approach by means of a case study. In the following, we start by presenting the study setup, which includes probands, employed equipment, and the chosen use case for the study. In the remainder of this section, we present the phases of our case study, namely introduction to ExplorViz, introduction to BIMSWARM, assignments, and the survey.

### 7.3.1 Setup

It is our goal to evaluate the potentials and drawbacks of the developed visualization approach for collaborative program comprehension. In the following, we describe the experimental setup which includes the background of participants, the required equipment, the employed use case, and the configuration of ExplorViz.

**Probands**
Based on the concept for the possible use cases, people with an IT background are our target group. This includes students of computer science and closely related subjects, as well as researchers in computer science and IT professionals. We do not give probands financial incentives but rely on volunteers. The technical background of the probands allows us to give a quick introduction to the principles of software visualization. This frees up time such that probands can test the developed approach more extensively. Furthermore, the technical background allows us to set practical tasks and not only to evaluate the design and usability of the visualization.

**Equipment**
To meet our requirements for the equipment, the probands need a mobile device, preferably a tablet computer, which has a touchscreen and a camera. In addition, an active internet connection is necessary to use ExplorViz with a mobile browser. At last, probands require the printed markers for the AR visualization. If no printer is available or the printing quality is inadequate, we also allow that markers are displayed on a monitor.

Usually, it is the goal of an evaluation to provide a controlled environment such that all probands have comparable conditions during the conduct of the case study. However, due to the remote nature of the experiment, our influence on the setup which probands use to conduct the study is limited. Due to the COVID-19 pandemic, we cannot design and prepare an experiment setup which requires probands to undertake the case study in a designated lab environment. Notably, the collaborative nature of our approach requires that

two probands can evaluate the developed approach together. Consequently, the probands need to conduct the case study remotely and it is not feasible to bring or send equipment to each proband individually. The remote setting also requires the probands to own a microphone which can be used to communicate during the conduct of the case study.

Our only influence on the equipment of the probands lies in the communication of requirements during the acquisition process. We presume that the acquisition of participants for a collaborative case study could be challenging and thus do not require the use of specific devices, a modern smartphone or tablet computer suffices. Consequently, the employed equipment could be quite diverse. This has potential impacts on our results which we consider and reflect in Section 7.5 and Section 7.7.

**Use Case**

We would like to present a realistic use case to the participants of the study. As described before, BIMSWARM is a current and Java-based software which is suitable for the analysis. We would like to capture the runtime behavior of a use case of BIMSWARM to employ it as an example for the case study. The employed use case should be easy to understand. BIMSWARM offers some domain-specific functionalities, which however would require more background knowledge and thus a more detailed introduction.

Since we want to focus the study on the use of the approach we developed, we decided to use the creation of a product as a use case. Organizations which are registered at BIMSWARM can create a product with various features. This use case is easy to understand after a brief introduction to BIMSWARM. However, several services of the backend are involved during the creation of a product. Worth mentioning are the Product Service, User Service, and File Service. The File Service is involved by uploading a logo for the product.

**ExplorViz**

It is our goal to present the same visualization to all probands, especially the same runtime data. The use of the backend of ExplorViz for live visualization does not allow us to ensure that the same runtime data is always displayed. For example, the method calls of the use case could be split and show up in two different snapshots of ExplorViz. Also, when employing live trace analysis, the appropriate timestamp would need to be selected for the visualization, which seems impractical to us for a robust execution of a study.

Therefore, we execute the described use case of BIMSWARM on our local machine and save the JSON data that is sent to the frontend. ExplorViz has an application to mock the backend, called Demo Supplier. The Demo Supplier is a NodeJS server that can run in a Docker container and respond to requests from the frontend with previously deposited JSON files. We use the Demo Supplier to deposit the PetClinic as a sample application. In addition, we saved the recorded use case of BIMSWARM such that it is persisted for our probands to ensure consistency of visualized data across multiple experimental runs.

Another point concerns the user management of ExplorViz. ExplorViz uses *Auth0*[1] to

[1] https://auth0.com/

identify users, associate persisted data to them, and to manage access rights to stored data. An account with *Google*[2] or *Github*[3] is required to log in to ExplorViz. We see it as a potential hurdle that probands need to have their credentials for either of those services on hand. Also, in the free variant of Auth0, the domain of Auth0 is accessed for authentication. This is blocked by some browsers and can prevent the login. In order to avoid difficulties in the execution of the study due to errors in the authentication, we configure ExplorViz for our study in such a way that no registration of the probands is necessary. Consequently, each proband is assigned with an example account by the name of "Johnny Doe". Distinguishing users from each other during the collaborative use of our approach is still possible due to a unique color assignment.

### 7.3.2 Introduction to ExplorViz

Before the study begins, probands are provided with the markers (see Appendix 8.2) such that they can print them in advance. After an appointment has been made to conduct the study and the probands have joined a meeting in a video conferencing system of their choice, the study begins. At the beginning of the study, we familiarize the probands with ExplorViz and the developed approach. This phase also serves to clarify general questions and to resolve technical problems. Specifically, probands are asked to visit the website where ExplorViz has been deployed for our study.

The introduction is done with PetClinic as an example application because it exhibits an overseeable structure and at the same time there are enough classes and corresponding communication for the explanation of the visualization. We also want to ensure that the probands have similar starting conditions for solving the upcoming tasks about BIMSWARM. We assume that different amounts of time will be needed for the introduction and thus the probands could also gather different amounts of information about the displayed application.

Once the introduction to the visualization and the user interface is completed, the presentation of the use case begins, which is then to be analyzed.

### 7.3.3 Introduction to BIMSWARM

Before the probands analyze the runtime behavior of the selected use case on BIMSWARM, we provide an introduction to BIMSWARM. To keep this introduction efficient, we introduce BIMSWARM via screen transfer. This has the advantage that all probands receive the same information and participants do not have to create a user account with BIMSWARM.

Besides the presentation of BIMSWARM as a marketplace, we also give a brief introduction to the domain of BIM to outline the domain-specific background. However, we refrain from presenting more complex functionalities such as the combination of different

[2] https://www.google.com/

[3] https://github.com/

products. Hereby, we facilitate that the information relevant for the use case remains memorable. Finally, we present the dialog for the creation of a product within the frontend of BIMSWARM.

### 7.3.4 Assignments

The probands are asked to perform tasks collaboratively for the prepared use case of BIMSWARM. We do not focus on a quantitative evaluation of the assignments, but rather want to encourage the probands to interact with the developed visualization approach. Therefore, we want to foster the use of various features across all assignments.

Beyond the mere stimulation of interaction, the tasks should also show practical relevance. Where possible, we want to encourage semantic analysis of the visualized data. The tasks are roughly based on the tasks chosen for the evaluation of the VR extension of ExplorViz [Brück 2020].

T1: Software System Overview and Structure

- T1.1: Name all applications which are communicating with other applications.
- T1.2: Name all packages which only contain classes (no other packages) within the User Service and are developed by adesso.
- T1.3: Which package does contain more classes: *de.adesso.swarm.fileservice.service* or *de.adesso.swarm.toolchainservice.service*?
- T1.4: Name all classes (full name) in the package *de.adesso.swarm.productservice.model.dto.pagination*.

T2: Software System Dynamics

- T2.1: Which class has the highest number of newly created instances within the User Service? The number of instances is represented by the height of a class but can also be estimated by using the heat map.
- T2.2: Which pair of classes in the Product Service has the highest number of requests between them? The number of requests correlates with the thickness of the depicted communication. Name also the direction of the communication and the exact number of requests.
- T2.3: Which class within the Product Service has the highest overall request count? Can you explain why this class might have so many requests? You may use the highlighting feature on the class to identify which other classes are directly communicating with the found class.
- T2.4: Which classes within the Product Service have more than 150 outgoing requests but less than 20 incoming requests?

T2.5: Imagine you wanted to make extensive changes to the code of the FileInfo class, located in the package *de.adesso.swarm.fileservice.model*. Which other classes could be affected by your changes (based on the current use case)?

We designed the assignments such that it should take participants around 30 minutes to solve the tasks, depending on their background knowledge. Therefore, on the assumption that no major technical issues occur, the overall study should take no longer than 60 minutes. We expect it to be challenging to recruit probands for a longer period of time without financial incentives.

The assignments about system overview and structure should ascertain that the probands understand the visualization metaphor correctly. We include straightforward questions which concern applications, packages, and classes. Assignment A1.3 is meant to remind probands that multiple applications can be opened at once. However, we do not specify how the assignments should be solved. Therefore, opening and closing applications one at a time such that no more than one application is open would be an acceptable way to solve this assignment. By asking questions about various applications we also want to encourage collaboration and communication. For example, the probands could determine who is responsible for opening applications or packages. In addition, since the state of applications and components is synchronized, they could use the ping feature for mutual guidance in unfamiliar applications.

The assignments about the dynamics of a software system are regarding the runtime behavior, i.e. the requests between classes. Some of the assignments require the use of the heat map to compare classes or even different metrics. In addition, assignment A2.3 asks for the class with the most overall requests within the Product Service. The solution is the class *ProductDto*, a *data transfer object* (DTO). To explain the origin of these requests, probands would need to know what a DTO is or inspect the communication as shown in Figure 7.1 more closely to make assumptions about the role of the class.

### 7.3.5 Survey

Following the study, we ask the probands to take part in a survey. Since the study takes place remotely, we decided to conduct an online survey. The choice for the survey tool falls on *SurveyMonkey*[4] as it is widely used and supports both desktop computers and mobile devices. We do not collect personal data such as age or the IP address.

Our survey is structured in such a way that we ask about the user experience first since we thereby minimize the time gap to the completion of the tasks. Only further down the line, we ask more general questions, e.g. about the professional background or the environment in which the proband conducted the study.

First, we ask the probands how they perceived the difficulty of the assignments about software structure and software system dynamics respectively. The study probands can select *very difficult*, *difficult*, *intermediate*, *easy*, and *very easy*. We can use the results to assess

[4] `https://www.surveymonkey.de`

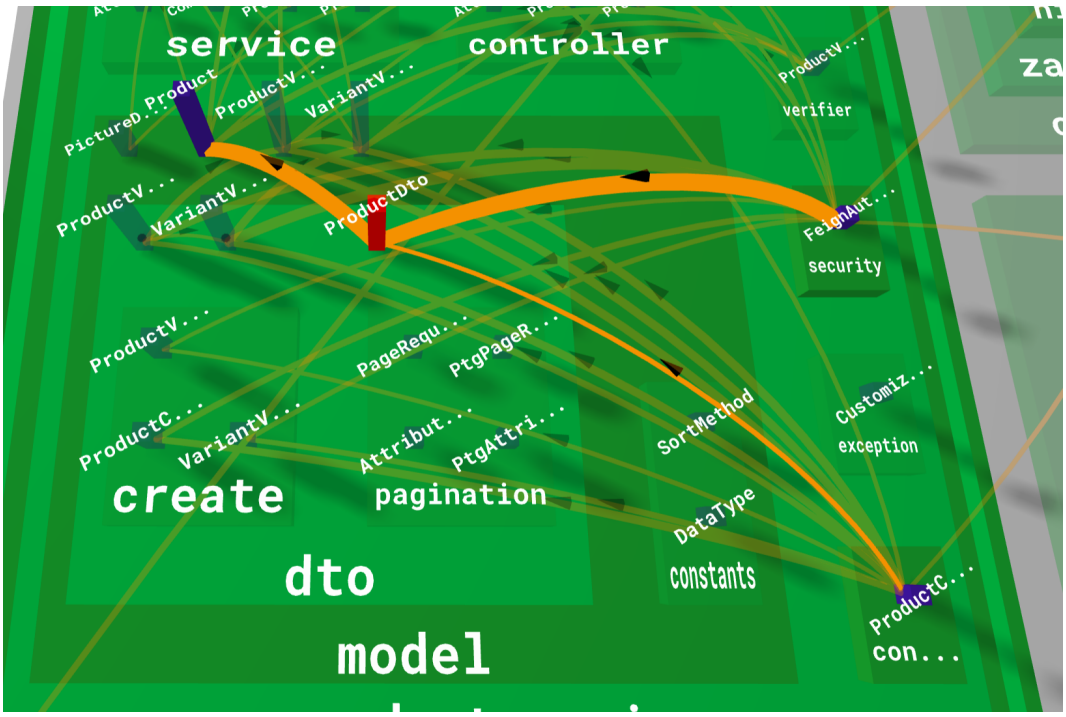



**Figure 7.1.** Excerpt of the Product Service with the highlighted class ProductDto.

whether the difficulty of the assignments is chosen appropriately. In the following, we present the questions of the survey which are most relevant to, later on, answer our research questions.

**Professional Background**
Our target group includes persons with a computer science background. Nonetheless, the level of expertise and practical knowledge might vary from person to person. Therefore, we ask the probands which of the following terms describes their professional background the best: *student*, *researcher*, *software developer*, *software architect*, *project manager*, or *other*. In addition, we ask the probands to self-assess their experience in different topics regarding software development, software analysis, and software visualization. The probands can choose from *none*, *beginner*, *intermediate*, *advanced*, or *expert* to categorize their level of expertise.

**Usability**
We provide various features to interact with and manipulate the displayed 3D software models. Therefore, we ask the probands to rate the usability of those features on a scale

which includes *very bad*, *bad*, *undecided*, *good*, and *very good* as options. The features which can be rated are displayed in Table 7.1.

**Table 7.1.** Features which probands could rate in Terms of Usability

| ID | Rated Feature |
|---|---|
| A1 | Touch gestures (pan, rotate, zoom) |
| A2 | Opening & closing of packages |
| A3 | Highlighting |
| A4 | Ping Feature |
| A5 | Heat map (e.g. request metrics) |
| A6 | Popups (movable window with information) |
| A7 | Settings / Configurability |
| A8 | User interface |

**Visualization**
We are interested if the established visualization approach of ExplorViz is suitable for mobile devices, too. To the best of our knowledge, there does not exist research on the applicability of the 3D city metaphor for such devices. Therefore, we ask the probands how much they agree with the statements in Table 7.2. For each statement, a participant can choose either *disagree*, *rather disagree*, *undecided*, *rather*, or *agree* as an answer.

**Table 7.2.** Statements on Visualization

| ID | Statement |
|---|---|
| B1 | The package structure can be easily accessed. |
| B2 | Component and class names can be read with ease. |
| B3 | Communication between classes can be distinguished easily. |
| B4 | The layout and visualization are suitable for the mobile device which I used. |

**Collaboration**
As we are interested in collaborative program comprehension, we present probands with various statements about the collaborative use of our approach. The statements are presented in Table 7.3 and are evaluated by the participants just like the statements about visualization.

**Table 7.3.** Statements on Collaboration

| ID | Statement |
|---|---|
| C1 | I know the other team member very well. |
| C2 | I already worked with the other team member before the study. |
| C3 | I felt working on the tasks as a team was helpful. |
| C4 | I felt I could communicate and interact well with my team member. |
| C5 | The AR mode of ExplorViz can be used to improve program comprehension. |
| C6 | The AR mode of ExplorViz is suitable for working in a team. |

**Markers**
Since we rely on markers to achieve the AR visualization, we ask the participants about their employed markers, the lighting condition, and the resulting recognition rate. The questions are displayed in Table 7.4.

**Table 7.4.** Questions and Statements on Markers

| ID | Statement |
|---|---|
| D1 | Which type of markers did you use? |
| D2 | Please rate the lighting conditions of your environment during the study. |
| D3 | Please rate the recognition rate of the markers. |

**Comments and Text Inputs**
In addition to the selection of predefined values, we give participants of our survey the opportunity to submit textual inputs on various topics. Table 7.5 displays the labels of the text boxes which we use in our survey. The topics include usability, visualization, collaboration, and markers. In addition, we also ask the probands about their mobile device since a predefined selection is not feasible due to the wide variety of supported devices.

**Table 7.5.** Additional Questions about further Comments

| ID | Statement |
|---|---|
| E1 | Do you have remarks about the usability of certain features? |
| E2 | Do you have further remarks about the visualization of applications (for mobile devices)? |
| E3 | Do you have further feedback about the collaborative aspects? |
| E4 | Which device(s) did you use? Be as specific as possible. |
| E5 | Additional comments about markers |
| E6 | Suggestions for improvements or other remarks |

## 7.4 Pilot Study

In order to gather early results and incorporate preliminary feedback into our implementation and experimental setup, we conduct a pilot study. The pilot study is comprised of two computer science students without prior knowledge about software visualization. Those students conduct the experiment as presented in Section 7.3 except that they are given the additional assignment of naming all classes which show communication with other applications. We exclude that question from the case study because it is of little relevance and we realize that the remaining questions will already require the targeted 30 minutes to answer.

The participants of the pilot study employed an older tablet computer and a modern smartphone for the pilot study. We observed that the older tablet exhibited performance issues when multiple applications were opened. As a result, the opening and closing of applications were performed with a noticeable delay and eventually, the browser stopped the execution of ExplorViz due to limited resources. We suspect that the performance issues originate from the 1.5GB of RAM. Even though the modern smartphone of the other proband was warm to the touch, there were no such performance issues. Concluding, we require that devices should own at least 2GB of RAM to avoid such performance issues during the conduct of the case study.

In terms of implementation, the ping feature (see Section 6.8.2) is not included in the pilot study as this feature is a result of the hereby gathered feedback. The students of the pilot study experienced the situation that they talked about different packages which share the same name and are present in multiple applications. Thus, the ping feature is added to allow users to easily point at parts of the software model which they want to talk about. Furthermore, the probands occasionally panned the software models so far off the marker that the model was barely visible anymore. As a result, we introduce an entry in the context menu which allows resetting the position, rotation, and scaling of all opened landscape and application models. At last, the legend of the heat map can take up valuable

screen space on smartphones. Hence, we add the option to collapse the heat map legend such that only a small header and the name of the selected metric are displayed.

As soon as we had incorporated the feedback from the pilot study, we conducted the case study. The results of the case study are presented in the upcoming section.

## 7.5 Results

In this section, we present the results of the conducted case study. We only present accumulated results from our survey in this section. Comments and additional remarks are not presented as they are manifold. However, the complete dataset is publicly accessible [Hansen 2021].

**Probands**
20 participants conducted our case study, out of which 7 are students, 8 are researchers, 2 are software developers, and 3 are software architects. All participants have experience in either object-oriented programming or web development, i.e. they have at least a basic knowledge of software development.

9 of the participants used a smartphone and 11 participants used a tablet computer. Out of the 9 smartphones, 7 smartphones run Android, 2 run iOS, and for one smartphone the operating system is not specified. Out of the 11 tablet computers, 5 run iOS, 4 run Android, and 2 devices run *Microsoft Windows*[5] as an operating system.

**Usability**
In order to accumulate the collected data, we assign integer values to the given answers, the mapping is declared in Table 7.6. We choose the mapping such that the mean value over several answers can result in values between 0 and 4, whereby a value of 2 would indicate that answers that indicate a bad usability and answers that indicate a good usability balance each other out.

**Table 7.6.** Integer mapping for answers to questions about usability

| Answer | Value |
|---|---|
| Very Bad | 0 |
| Bad | 1 |
| Undecided | 2 |
| Good | 3 |
| Very Good | 4 |

[5] https://www.microsoft.com/windows/

The accumulated results for the usability of our results are presented in Table 7.7. In addition to the mean value we added the value for standard deviation (SD) as an indicator for the distribution of the given answers.

**Table 7.7.** Rating of the usability features

| ID | Mean (overall) | SD (overall) | Mean (tablets) | SD (tablets) |
|---|---|---|---|---|
| A1 | 2.75 | 1.26 | 3.33 | 1.25 |
| A2 | 3.15 | 0.73 | 3.27 | 0.62 |
| A3 | 2.65 | 1.01 | 3.09 | 0.67 |
| A4 | 3.15 | 0.79 | 3.45 | 0.50 |
| A5 | 2.70 | 1.00 | 2.72 | 0.86 |
| A6 | 2.65 | 1.24 | 3.09 | 0.90 |
| A7 | 2.80 | 0.98 | 3.09 | 0.79 |
| A8 | 3.00 | 1.00 | 3.27 | 0.62 |

**Visualization**
To evaluate aspects of the visualization and collaboration, probands were asked to state their level of agreement with a given statement. In order to allow the accumulation of results, we again use a mapping to integer values that is analogous to the previous mapping. We display the mapping in Table 7.8.

**Table 7.8.** Integer mapping for answers to statements about visualization and collaboration

| Answer | Value |
|---|---|
| Disagree | 0 |
| Rather disagree | 1 |
| Undecided | 2 |
| Agree | 3 |
| Rather agree | 4 |

The accumulated results for the statements concerning the visualization are presented in Table 7.9.

**Table 7.9.** Rating of the visualization

| ID | Mean (overall) | SD (overall) | Mean (tablets) | SD (tablets) |
|---|---|---|---|---|
| B1 | 3.50 | 0.81 | 3.72 | 0.62 |
| B2 | 1.85 | 1.19 | 2.09 | 0.90 |
| B3 | 2.75 | 1.18 | 2.64 | 1.30 |
| B4 | 2.70 | 1.35 | 3.45 | 0.89 |

**Collaboration**
The accumulated results concerning statements about collaboration are presented in Table 7.10.

**Table 7.10.** Rating of collaborative aspects

| ID | Mean (overall) | SD (overall) | Mean (tablets) | SD (tablets) |
|---|---|---|---|---|
| C1 | 3.45 | 1.20 | 3.45 | 1.16 |
| C2 | 3.30 | 1.45 | 3.45 | 1.23 |
| C3 | 3.10 | 1.18 | 3.18 | 1.03 |
| C4 | 3.70 | 0.56 | 3.55 | 0.78 |
| C5 | 3.25 | 0.83 | 3.45 | 0.50 |
| C6 | 3.60 | 0.58 | 3.73 | 0.45 |

**Markers**
Regarding question D1, 5 probands used markers on thick DIN A5 paper that we provided, 5 participants printed markers themselves, 9 participants displayed markers on a display, and 1 participant used different kinds of markers throughout the study. With respect to question D2, 7 probands described the lighting conditions during the study as very good, 11 as good, and 2 stated that their lighting conditions are intermediate. For question D3, 8 probands described the recognition rate of the markers as very good, 9 as good, and 3 participants stated that the recognition rate is intermediate.

## 7.6 Discussion

This section discusses the abovementioned results. Therefor, we take a look at four different aspects which correspond with our research questions, namely assignments, visualization,

usability, and collaboration.

**Assignments**
The assignments were largely solved correctly by the probands. At times an incorrect answer was given at first but this was usually noticed and corrected by the other proband. Incorrect answers could also be explained by a communication problem and due to our additional notes the assignments could be solved without great difficulty. In the survey, no probands indicated that the assignments were perceived as very difficult either. Moreover, only one person found the assignments on software structure as difficult, for software dynamics it are also only two persons.

For task T2.3 the probands were asked how the high number of method calls for the previously identified class *ProductDto* can be explained. This semantic explanation requires prior knowledge about data transfer objects (DTO), which was not present for all probands. However, since this is an open question that cannot be answered by data from the visualization alone, we do not see this as a shortcoming for our approach.

Probands gave question C5, which asks about the suitability of our approach to improve program understanding, an overall rating of 3.25. Since the rating scale goes from 0 to 4, we see this as a very positive sign. It can therefore be assumed that the developed approach is in general suitable for tasks in the context of dynamic program analysis.

**Visualization**
Regarding B1, a question about the package structure, an overall rating of 3.50 and a rating of 3.72 for tablets indicates that the package structure of applications is also easy to understand on mobile applications. However, there were individuals who had no previous experience with software visualizations and found the order of the packages confusing at first. In the visualization, top-level packages, which therefore contain many other packages, appear visually at the bottom. From development environments, one usually expects that the name of these packages is at the top. However, this confusion was short-lived for the people concerned, as the opening and closing of packages in the visualization quickly revealed the underlying principles which are employed by the 3D city metaphor. Overall, the presented package structure, as known from the frontend core of ExplorViz, seems to work well on mobile devices.

Question B2 deals with the readability of labels for classes and packages. With an overall rating of 1.85 and a rating of 2.09 for tablets, this aspect received the lowest rating within our survey. The feedback shows that the names for packages are perceived as sufficiently large, while the readability of class names can still be improved. There are suggestions that the labels should already be displayed completely as soon as the user aims at a class with the crosshair or that the labels are not shortened when a class is magnified by the zoom feature. We consider these change requests to be very reasonable. However, when implementing them, the possible performance impacts should be monitored. Detecting if a class is currently targeted by the crosshair requires frequent calculations.

Question B3 asks about the distinguishability of communication lines. With an overall rating of 2.75 and a rating of 2.64 for tablet computers, respectively, this is rated rather good, but there is room for improvement here. It was noted, for example, that many lines of communication lying on top of each other are difficult to distinguish. It was suggested that the communication lines should be distinguished not only in thickness but also in color. This corresponds to a heat map visualization for communication lines. We consider this change to be very useful, which also has advantages for the regular visualization of ExplorViz.

Question B4 asks in general about the suitability of the layout and visualization for the employed device. With 2.70, the overall rating is still rather good, but for tablets it results in a rating of 3.45. Also considering the comments of the probands during the study, we consider smartphones are best suited for the mere visualization of the 3D models. On the other hand, tablet computers have a larger screen and are clearly better suited for an analysis of the visualized data. In our estimation, visualization on tablet computers thus proves to be very promising. The visualization on smartphones should continue to be supported as best as possible, but we suspect the practical use cases for smartphones are constrained.

**Usability**

The results for the questions A1 to A8 on the topic of usability are promising. From a possible overall rating between 0 and 4, the average values are between 2.65 and 3.15. However, it is noticeable that the rating of probands who have used tablet computers is better for every aspect. For results for tablet computers achieve values between 2.72 and 3.45 for the questions in this category.

We, therefore, conclude that usability is better achieved for tablet computers, as already assumed from the beginning. The larger screen of tablets allows more space to display the UI elements and significantly less scaling or zooming are required to explore the software. In addition, the selection of entities, especially for classes and lines of communication between classes, is significantly more reliable with a larger display.

During the study, some probands expressed that they would like to interact with the displayed models of the visualization with their finger or that they actually tried so without success. Depending on their current assignment, the probands expressed that the ping feature should be activated at the touched location or that a popup should be opened for the targeted entity. We consider the implementation of a ping to be the most sensible since precise targeting is not so relevant for that feature. If a class is meant to be targeted, but the package under the class is hit by the imprecise touch input, the semantics of the displayed ping will remain understandable.

When looking at the individual features, the popups, highlighting, and heat map gained the lowest ratings. In the following, we will take a closer look at the weaknesses and the reasons for these lower ratings in order to provide a basis for future improvement.

In the case of popups, individual probands with smartphones had the problem that

popups were not displayed on their screen or were only shown in landscape format. We suspect that the popups were displayed outside the viewport due to the employed browser. In addition, the sometimes difficult targeting of classes was criticized. One could counter this by offering the option to only display information on either packages, classes, or communication. With this addition, inaccurate targeting by the users could be adapted such that it is mapped to the closest class, package, or communication line.

For highlighting, there were individual comments that multiple entities within an application should be highlightable at the same time. We could imagine this, but we are not sure if the semantics regarding transparent and displayed communication is still preserved. Otherwise, we observed that some probands could only unhighlight an entity with difficulty. Unhighlighting an entity is currently only possible by highlighting an already highlighted entity again. This limitation originates from the frontend core and will soon be lifted so that the highlighting can be removed by targeting the background.

Regarding the heat map, there were two main criticisms. First, the heat map is projected onto the gray foundation of an application, which creates a gap between the class and the corresponding colored area of the heat map. When viewed in three dimensions in AR, this can lead to perspective shifts, such that colors are not aligned with the corresponding class. The second point of criticism concerns the gradient of the colored areas that belong to a class. In this gradient, the centered color indicates the actual value of the heat map. Even if this visualization should help to differentiate the colors more easily, some users have asked about the semantics of this coloring and have been unsure how to estimate the values using the heat map legend. The mentioned problems could be solved by displaying the colors of the heat map directly under a class and giving the class itself the assigned heat map color as well.

Overall, we conclude that the usability of the implemented features to be good. However, the aforementioned feedback should be implemented alongside other minor improvements to make the use of our approach more intuitive.

**Collaboration**

For the topic of collaboration, we note that most probands from the study know each other and have worked together before. However, this would also apply to people who develop software together in a work environment. Therefore, we assume this to be a viable precondition for the evaluation of collaborative features.

Question C3 asks whether working together was beneficial. With an overall rating of 3.10 most probands agreed with this statement. One proband noted that the tasks were too easy and therefore the collaborative work was unnecessary. On the other hand, another respondent, whose partner had more technical knowledge than him or her, described the collaborative teamwork as very helpful.

Furthermore, we can state that the communication and interaction worked very well, question C4 reaches a score of 3.70. Throughout the conduct of the study, there was also only one proband, for whom a weak Internet connection led to issues regarding communication.

C6 deals with the question of whether the developed approach is suitable for collaborative work in general. We find that with an overall rating of 3.60 and a rating for tablets of 3.73, this question was almost exclusively answered positively. The additional comments in the survey on collaboration are also positive and highlight collaborative working in ExplorViz as a particular plus.

The positive results regarding collaboration are certainly due in part to the fact that the probands mostly knew each other. According to our observations, the probands who already knew each other well also exchanged ideas more intensively during the study. Overall, however, we see confirmation that the chosen approach is very promising for collaborative program comprehension.

## 7.7 Threats to Validity

In this section, we present the most important threats to the validity of our evaluation. These threats regard the probands, study setup, and relevance of the assignments.

**Probands**
20 probands conducted our case study. This number makes it hard to generalize our results. Also, only 4 probands have currently a practical background like software engineer or software architect. 16 probands are associated with academia. Therefore, the gathered results provide first insights into the potential of our approach, but there is too little data to state whether the developed approach could be adopted by software professionals for practical applications. Another bias could be introduced due to the fact that we know almost every proband personally. However, we stressed that open and honest feedback is most beneficial to us. In addition, we encouraged probands to call out the issues they encounter during the study and stated that such feedback is very helpful for the further development of the evaluated approach.

**Study Setup**
As mentioned before, the setup of the study does not allow for a controlled experiment. Therefore, there exist various factors which influence the gathered results aside from the professional background of the probands. These factors include the diversity of employed devices, employed markers, lighting conditions, available desk space, internet bandwidth, as well as microphone and headphone quality. However, we anticipated most of these factors and designed our study such that qualitative results are attained. It is also notable that the conduct of such a study in a lab environment would probably improve the gathered results because the study setup could be extensively tested and optimized in advance.

**Relevance of the Assignments**
The assignments for our case study are kept short and we mostly request the names of one

or more classes. Even though we asked participants to reflect their findings in assignment T2.3 and add a practical use case to assignment T2.5, we expect real-world tasks to be more complex. However, complex assignments that ask probands to analyze the runtime behavior of a given software more extensively would require much more time. Our focus is on the evaluation of the feasibility of our approach. In addition, we argue that the assignments of our case study can be the building blocks for more complex tasks.

Chapter 8

# Conclusions and Future Work

In this chapter, we summarize our developed approach for program exploration using AR. We then take the gathered results from our evaluation to assess the potentials and drawbacks for the collaborative use of our approach in practical applications. At last, building upon our previous findings, we identify promising topics for future work.

## 8.1 Conclusions

In the course of this thesis, we developed an approach for an AR software visualization which can be used collaboratively. For this purpose, we use mobile devices, i.e. tablet computers and smartphones, and combine their camera images with virtual software models. The 3D models are aligned within the camera image with the help of printed markers.

In a concept, we have laid the foundation for the resulting implementation. The landscape model and application models of ExplorViz are placed on different markers. The user is also provided with a variety of interaction options, such as a heat map, for analyzing the visualization. Collaborative work is also enabled by synchronizing the status of applications and providing further collaborative features.

In a case study, probands solved given tasks collaboratively and subsequently evaluated the developed approach. The tasks were concerned with the analysis of the Java-based BIMSWARM software system. The study was conducted in a location-independent manner and feedback was collected for a variety of different devices. The results of the study indicate that an AR visualization approach for mobile devices, especially tablet computers, is suitable for improving program comprehension. In particular, the collaborative aspects of our approach received positive feedback and should be further developed.

To our knowledge, no comparable approach exists that combines tablet computers and an AR software visualization to collaboratively improve program comprehension. Therefore, we see the necessity for further research regarding this promising topic.

## 8.2 Future Work

In this section, we discuss ideas for future work which originate from the conducted case study.

**Extension of developed Approach**
The results of the conducted case study are very promising. Therefore, the collected feedback from the evaluation should be used to further develop and extend our approach. For usability, for example, the control with touch gestures could be even more intuitive and versatile. For the visualization a few ideas were collected, such as the different coloring of communication lines, which can also benefit the other visualization approaches of ExplorViz. Collaboration was positively received, but additional visual indicators can be introduced to further simplify the location-independent use.

**Further Case Study**
We have already conducted a case study and were able to gain initial insights into our approach. However, our case study has been exposed to a lot of influences due to its remote execution. For example, the employed devices and markers are hardly comparable from one group of probands to another. In order to obtain more meaningful results about the developed approach, it would be helpful to conduct a study in a laboratory environment. In such a study, the employed devices and markers can be selected in a controlled manner.

In addition, such a study could be used to explore collaborative work among probands who are in the same room or sit at the same table. Our study indicates that communication by means of videoconferencing tools is already suitable, but we hypothesize that face-to-face communication, including gestures and facial expressions, can further benefit collaborative work. Last, it would also be desirable to have both an increased number of subjects and an increased proportion of active software developers among the probands.

**Cross-Plattform Collaboration**
Our case study provides a strong indicator that collaborative work can help to improve program comprehension. We, therefore, recommend leveraging the platform independence of ExplorViz to also explore collaboration when using different visualizations simultaneously. Since our approach uses the VR service for synchronization, AR and VR can already be combined without many modifications. Collaborative work between AR, VR, and the visualization for computers would still be very interesting. For this, however, it would be strongly recommended that the visualization of ExplorViz allows the simultaneous display of the landscape and multiple application models or at least enables fast switching between applications. If the visualization for computers is adapted accordingly, these three visualizations could be combined to enable a variety of novel use cases.

# Bibliography


[Aires et al. 2018] M. Aires, M. López-Alonso, and M. Martinez-Rojas. Building information modeling and safety management: A systematic review. *Safety Science* 101 (Jan. 2018), pages 11–18. (Cited on page 5)

[Anthes et al. 2016] C. Anthes, R. García Hernandez, M. Wiedemann, and D. Kranzlmüller. State of the Art of Virtual Reality Technologies. In: Mar. 2016, pages 1–19. (Cited on page 9)

[Azhar 2011] S. Azhar. Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. *Leadership and Management in Engineering* 11 (July 2011), pages 241–252. (Cited on page 7)

[Azuma 1997] R. T. Azuma. A Survey of Augmented Reality. *Presence: Teleoper. Virtual Environ.* 6.4 (Aug. 1997), pages 355–385. (Cited on page 8)

[Billinghurst et al. 2001] M. Billinghurst, H. Kato, and I. Poupyrev. The MagicBook - Moving seamlessly between reality and virtuality. *Computer Graphics and Applications, IEEE* 21 (June 2001), pages 6–8. (Cited on page 8)

[Borrmann et al. 2018] A. Borrmann, M. König, C. Koch, and J. Beetz. Building Information Modeling: Why? What? How?: Technology Foundations and Industry Practice. In: Sept. 2018, pages 1–24. (Cited on pages 5, 6)

[Brück 2020] J. Brück. Collaborative Program Comprehension based on Virtual Reality. Bachelorarbeit. Kiel University, Apr. 2020. URL: http://eprints.uni-kiel.de/49581/. (Cited on pages 47, 51)

[Chotisarn et al. 2020] N. Chotisarn, L. Merino, X. Zheng, S. Lonapalawong, T. Zhang, M. Xu, and W. Chen. A systematic literature review of modern software visualization. *Journal of Visualization* 23.4 (2020), pages 539–558. (Cited on page 7)

[Dieberger and Frank 1998] A. Dieberger and A. Frank. A city metaphor for supporting navigation in complex information spaces. *Journal of Visual Languages and Computing - VLC* (Jan. 1998). (Cited on page 13)

[Fittkau et al. 2017] F. Fittkau, A. Krause, and W. Hasselbring. Software Landscape and Application Visualization for System Comprehension with ExplorViz. *Information and Software Technology* 87 (July 2017), pages 259–277. (Cited on pages 10, 15)

[Fittkau et al. 2015] F. Fittkau, S. Roth, and W. Hasselbring. ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes. In: *23rd European Conference on Information Systems (ECIS 2015)*. May 2015. (Cited on page 12)

[Hansen 2018] M. Hansen. Collaborative Software Exploration with the HTC Vive in ExplorViz. Bachelor thesis. Kiel University, Sept. 2018. (Cited on pages 15, 47)

[Hansen 2021] M. Hansen. *Evaluation results - Colaborative Program Comprehension based on Augmented Reality (Master's Thesis)*. Zenodo, July 2021. URL: https://doi.org/10.5281/zenodo.5075126. (Cited on page 57)

[Häsemeyer 2017] T. Häsemeyer. Kollaboratives Erkunden von Software mithilfe virtueller Realität in ExplorViz. Bachelor thesis. Kiel University, Sept. 2017. (Cited on page 15)

[Hasselbring et al. 2020] W. Hasselbring, A. Krause, and C. Zirkelbach. ExplorViz: Research on software visualization, comprehension and collaboration. *Software Impacts* 6 (Nov. 2020). (Cited on pages 1, 10, 15)

[Hove and Anda 2005] S. Hove and B. Anda. Experiences from Conducting Semi-structured Interviews in Empirical Software Engineering Research. In: volume 2005. Oct. 2005, 10 pp.-. (Cited on page 48)

[Jiang et al. 2016] Y. Jiang, X. Liu, F. Liu, D. Wu, and C. Anumba. An Analysis of BIM Web Service Requirements and Design to Support Energy Efficient Building Lifecycle. *Buildings* 6 (Apr. 2016), page 20. (Cited on page 5)

[König 2018] D. König. Collaborative Software Exploration with the Oculus Rift in ExplorViz. Bachelor thesis. Kiel University, Sept. 2018. (Cited on pages 15, 47)

[Krause 2015] A. Krause. Erkundung von Softwarestädten mithilfe der virtuellen Realität. Bachelor thesis. Kiel University, Sept. 2015. (Cited on page 15)

[Maletic et al. 2002] J. Maletic, A. Marcus, and M. Collard. A task oriented view of software visualization. In: Feb. 2002, pages 32–40. (Cited on page 21)

[*MDN Web Docs WebXR Device API*]. *MDN Web Docs webxr device api*. https://developer.mozilla.org/en/docs/Web/API/WebXR_Device_API. Accessed: 2021-01-09. (Cited on page 10)

[Meins-Becker et al. 2019] A. Meins-Becker, A. Kelm, M. Kaufhold, M. Quessel, and M. Helmus. BUILDING INFORMATION MODELING AND OPERATION. *Proceedings of International Structural Engineering and Construction* 6 (May 2019). (Cited on page 5)

[Milgram et al. 1994] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies* 2351 (Jan. 1994). (Cited on page 8)

[Ohzawa 1998] I. Ohzawa. Mechanisms of stereoscopic vision: the disparity energy model. *Current Opinion in Neurobiology* 8.4 (1998), pages 509–515. (Cited on page 8)

[Robertson et al. 1997] G. Robertson, M. Czerwinski, and M. van Dantzich. Immersion in Desktop Virtual Reality. In: *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*. UIST '97. Banff, Alberta, Canada: Association for Computing Machinery, 1997, pages 11–19. (Cited on page 8)

[Schreiber et al. 2019] A. Schreiber, L. Nafeie, A. Baranowski, P. Seipel, and M. Misiak. Visualization of Software Architectures in Virtual Reality and Augmented Reality. In: *2019 IEEE Aerospace Conference*. 2019, pages 1–12. (Cited on pages 18, 19)

[Seipel et al. 2019] P. Seipel, A. Stock, S. Santhanam, A. Baranowski, N. Hochgeschwender, and A. Schreiber. Speak to your Software Visualization—Exploring Component-Based Software Architectures in Augmented Reality with a Conversational Interface. In: *2019 Working Conference on Software Visualization (VISSOFT)*. 2019, pages 78–82. (Cited on page 18)

[Souza et al. 2012] R. Souza, B. da Silva, T. Mendes, and M. Mendonça. SkyscrapAR: An Augmented Reality Visualization for Software Evolution. In: Jan. 2012. (Cited on pages 17, 18)

[Sutherland 1968] I. E. Sutherland. A Head-Mounted Three Dimensional Display. In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: Association for Computing Machinery, 1968, pages 757–764. (Cited on page 8)

[*W3C Chrome Hardware Support*]. *W3C Chrome Hardware Support*. `https://immersiveweb.dev/chrome-support.html`. Accessed: 2021-01-09. (Cited on page 15)

[Wettel and Lanza 2007] R. Wettel and M. Lanza. Visualizing Software Systems as Cities. In: June 2007, pages 92–99. (Cited on page 13)

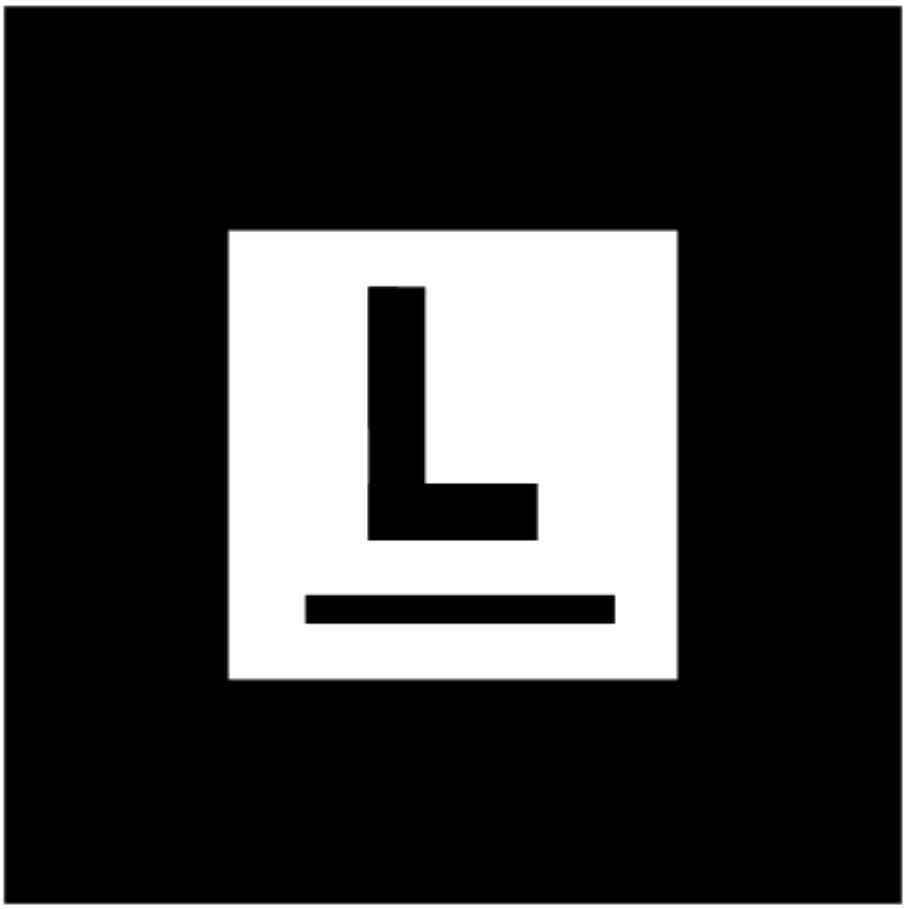
[wu et al. 2017] Z. wu, R.-D. Chang, and Y. Li. Building Information Modeling (BIM) for green buildings: A critical review and future directions. *Automation in Construction* 83 (Nov. 2017), pages 134–148. (Cited on page 5)

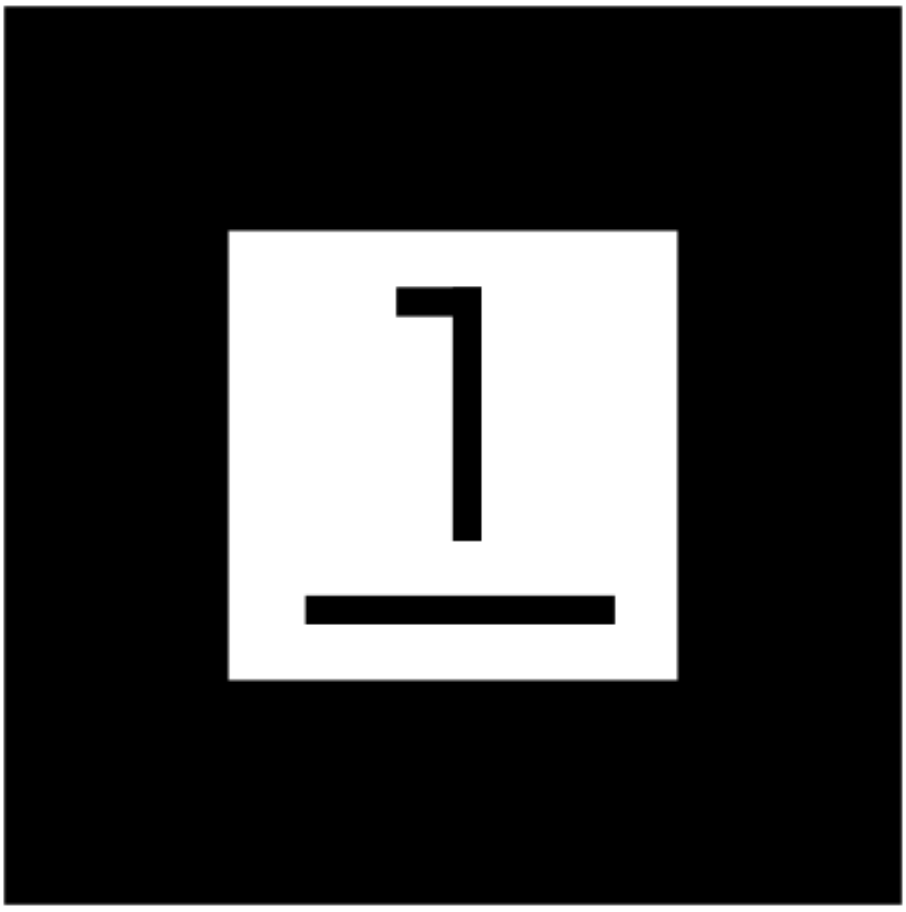
[Zirkelbach 2021] C. Zirkelbach. *Collaborative Reengineering and Modularization of Software Systems*. Kiel Computer Science Series 2021/4. Dissertation, Faculty of Engineering, Kiel University. Department of Computer Science, Kiel University, 2021. (Cited on page 15)

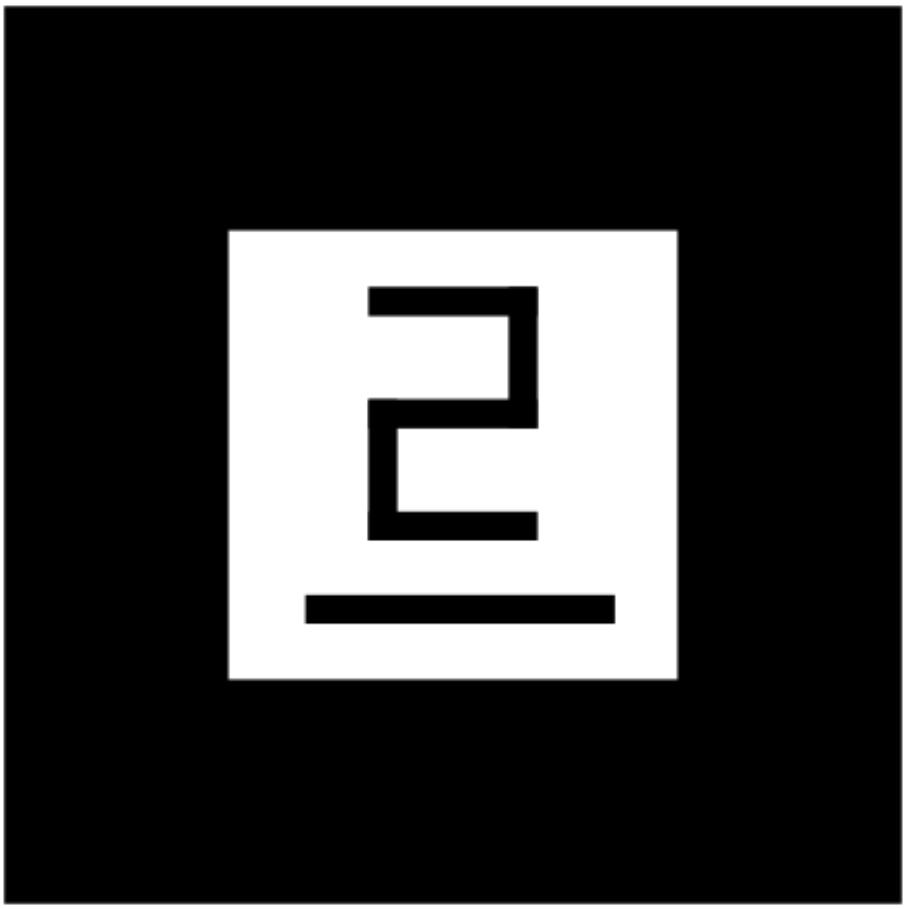
[Zirkelbach et al. 2018] C. Zirkelbach, A. Krause, and W. Hasselbring. On the Modernization of ExplorViz towards a Microservice Architecture. In: *Software Engineering*. 2018. (Cited on page 11)

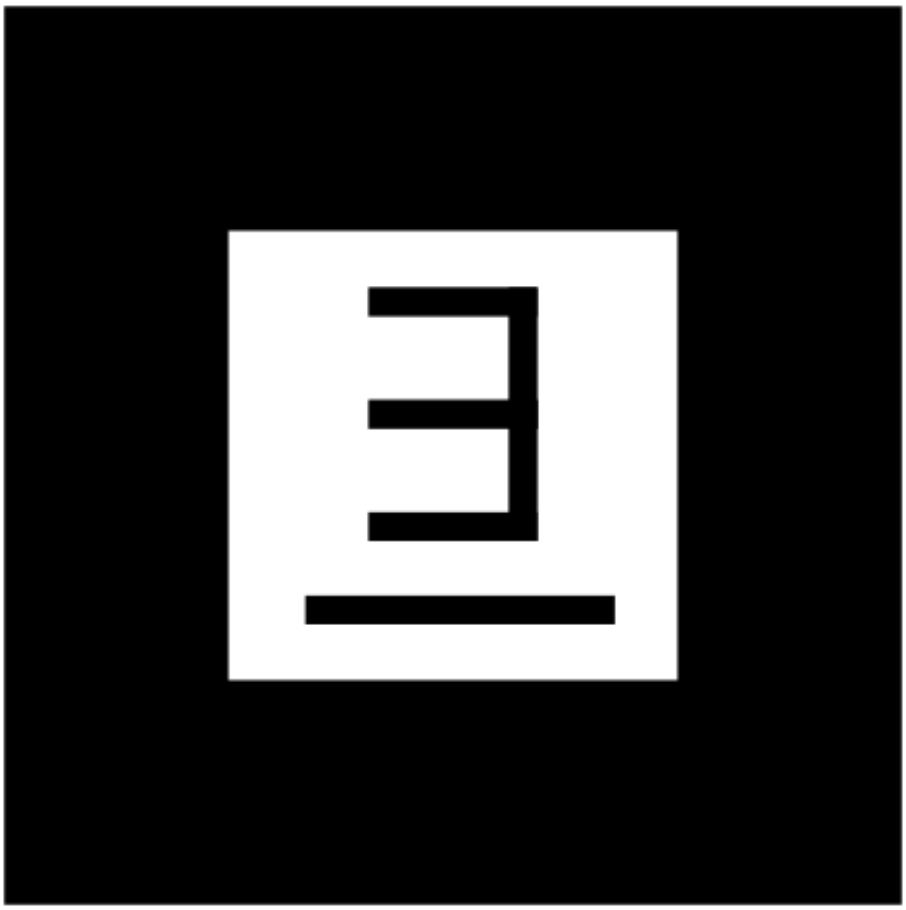
[Zirkelbach et al. 2019] C. Zirkelbach, A. Krause, and W. Hasselbring. Modularization of Research Software for Collaborative Open Source Development. In: *The Ninth International Conference on Advanced Collaborative Networks, Systems and Applications (COLLA 2019)*. June 2019, pages 1–7. (Cited on page 12)

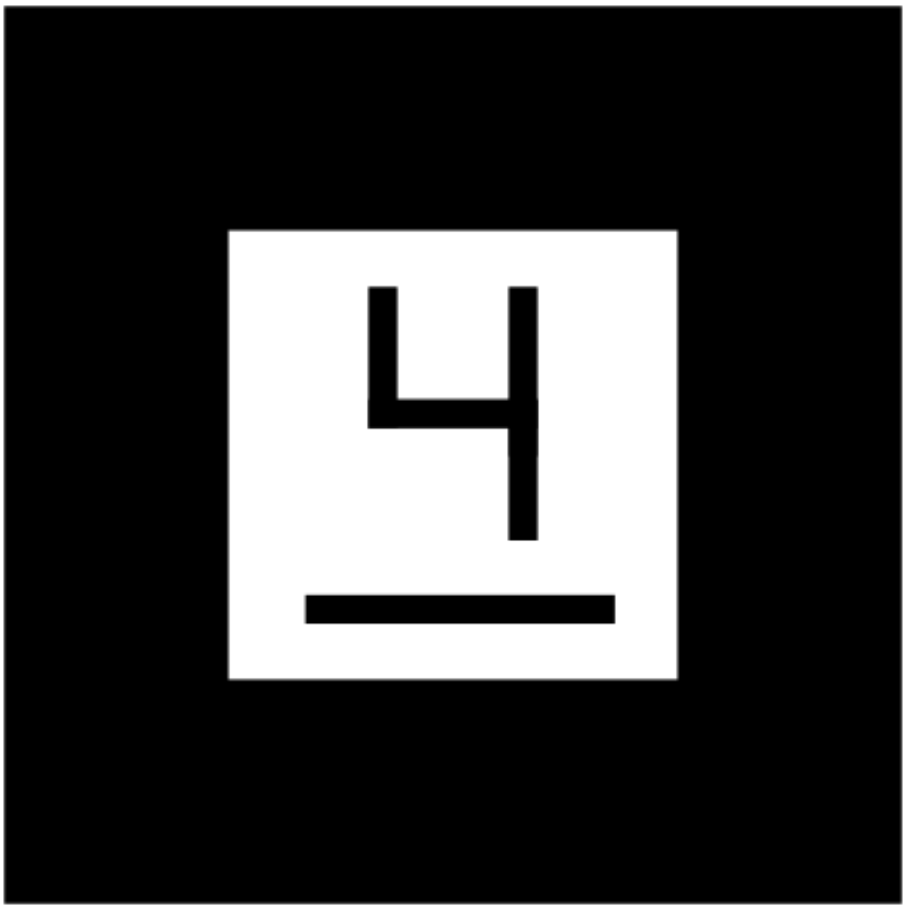
# Appendix A

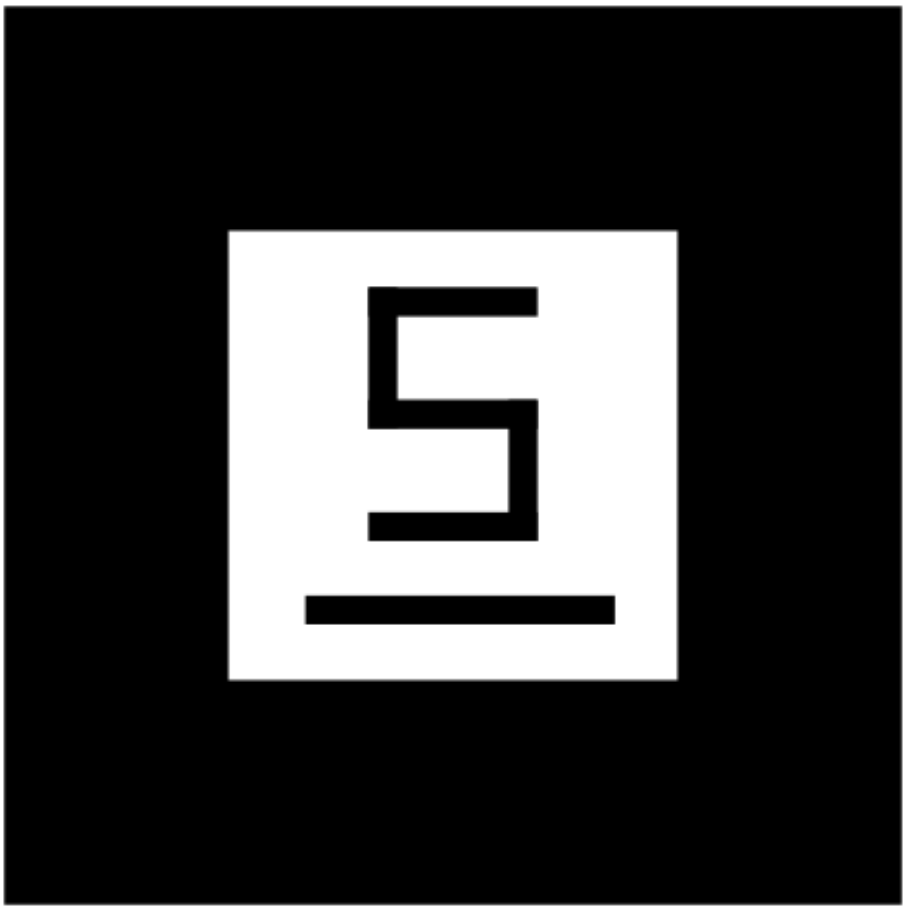












# Appendix B

ExplorViz

## Collaborative Augmented Reality (Program Comprehension Study)

* 1. Please enter your team name (name on which you and the other team member secretly agreed upon).

* 2. How did you perceive the difficulty of the tasks from the different categories?

| | Very Difficult | Difficult | Intermediate | Easy | Very Easy |
|---|---|---|---|---|---|
| Software Structure (concerning packages and classes) | ○ | ○ | ○ | ○ | ○ |
| Software Dynamics (concernings number of instances and requests) | ○ | ○ | ○ | ○ | ○ |

* 3. Rate following features or aspects in terms of usability

| | Very Bad | Bad | Undecided | Good | Very Good |
|---|---|---|---|---|---|
| Touch gestures (pan, rotate, zoom) | ○ | ○ | ○ | ○ | ○ |
| Opening & Closing of packages | ○ | ○ | ○ | ○ | ○ |
| Highlighting (coloring of classes / packages) | ○ | ○ | ○ | ○ | ○ |
| Ping (colored dots) | ○ | ○ | ○ | ○ | ○ |
| Heatmap (e.g. Request metrics) | ○ | ○ | ○ | ○ | ○ |
| Popups (movable window with infos, e.g. about classes / packages) | ○ | ○ | ○ | ○ | ○ |
| Settings / Configurability | ○ | ○ | ○ | ○ | ○ |
| User Interface (blue buttons etc.) | ○ | ○ | ○ | ○ | ○ |

4. Do you have remarks about the usability of certain features?

* 5. Regarding the layout and visualization of applications, how much do you agree with the following statements?

| | Disagree | Rather disagree | Undecided | Rather agree | Agree |
|---|---|---|---|---|---|
| The package structure can be easily accessed. | ○ | ○ | ○ | ○ | ○ |
| Component and class names can be read with ease. | ○ | ○ | ○ | ○ | ○ |
| Communication between classes can be distinguished easily. | ○ | ○ | ○ | ○ | ○ |
| The layout and visualization is suitable for the mobile device which I used. | ○ | ○ | ○ | ○ | ○ |

6. Do you have further remarks about the visualization of applications (for mobile devices)?

* 7. How much do you agree with the following statements?

| | Disagree | Rather disagree | Undecided | Rather agree | Agree |
|---|---|---|---|---|---|
| I know the other team member very well. | ○ | ○ | ○ | ○ | ○ |
| I already worked with the other team member before the study. | ○ | ○ | ○ | ○ | ○ |
| I felt working on the tasks as a team was helpful. | ○ | ○ | ○ | ○ | ○ |
| I felt I could communicate and interact well with my team member. | ○ | ○ | ○ | ○ | ○ |
| The AR mode of ExplorViz can be used to improve program comprehension. | ○ | ○ | ○ | ○ | ○ |
| The AR mode of ExplorViz is suitable for working in a team. | ○ | ○ | ○ | ○ | ○ |

8. Do you have further feedback about the collaborative aspects?

9. Which device(s) did you use? Be as specific as possible.

* 10. Which type of markers did you use?

- ( ) Provided markers (thick DIN A5 paper)
- ( ) Self-printed markers
- ( ) Markers displayed on a monitor
- ( ) Combination of different types

* 11. Please rate the lighting conditions of your environment during the study:

- ( ) Very Good (bright and even illumination for markers)
- ( ) Good
- ( ) Intermediate
- ( ) Bad
- ( ) Very Bad (uneven and low level of illumination for markers)

* 12. Please rate the recognition rate of the markers (reliable placement of models on correct marker).

- ( ) Very Good
- ( ) Good
- ( ) Intermediate
- ( ) Bad
- ( ) Very Bad

13. Additional comments about markers:

* 14. Which of the following statements apply to your environment during the study?

- [ ] I was in the same room as my team member.
- [ ] I was at home.
- [ ] I was at my workplace.
- [ ] I had a sufficient amount of space (desk space, room to move around).
- [ ] It was a distracting environment (e.g. noisy).
- [ ] None of the above.

* 15. What describes your professional background best?

- Student
- Researcher
- Software Developer
- Software Architect
- Project Manager
- Other

* 16. Rate your experience in the following fields:

| | None | Beginner | Intermediate | Advanced | Expert |
|---|---|---|---|---|---|
| Java, C++, or similar | | | | | |
| Web Development | | | | | |
| Software Development in Teams | | | | | |
| Static Software Analysis | | | | | |
| Dynamic Software Analysis | | | | | |
| Program Comprehension | | | | | |
| Reengineering and Reverse-Engineering | | | | | |
| Augmented Reality | | | | | |
| Software Visualization (Tools like ExplorViz) | | | | | |
| ExplorViz | | | | | |

17. Suggestions for improvements or other remarks: