Collaborative Software Visualization for Program Comprehension

Alexander Krause-Glau
Software Engineering Group
Kiel University
Kiel, Germany
akr@informatik.uni-kiel.de

Marcel Bader
Software Engineering Group
Kiel University
Kiel, Germany
stu216826@mail.uni-kiel.de

Wilhelm Hasselbring
Software Engineering Group
Kiel University
Kiel, Germany
wha@informatik.uni-kiel.de

***Abstract*—In the context of program comprehension, learning and working in teams, e.g., via pair programming, shared documentation, and discussions, can facilitate the comprehension tasks. So far, team collaboration is a relatively unexplored aspect in software visualizations, in particular approaches which are designed and explored to enable collaborative program comprehension.**

In this paper, we introduce our collaboratively usable software visualization environment for program comprehension. Related approaches are often limited to single-user modes, therefore neglect the advantages of multi-user collaboration, or allow only the use of a single type of device. Our approach addresses this topic and allows users to collaboratively explore software visualizations in a device-heterogeneous environment. User events, e.g., sharable pop-up information windows, are synchronized between each session participant, regardless of the employed device.

To the best of our knowledge, this is one of the first approaches that combines on-screen, virtual reality, and augmented reality modes in a single web-based SV tool for program comprehension. We conducted a user study to collect initial results regarding the perceived usefulness and enjoyment of co-explored software cities In that study, 20 participants collaboratively solved program comprehension tasks while using each mode consecutively. The results indicate that the majority of participants find our approach useful and enjoyable, with AR being the least favored mode. We provide each participant's video recording, the study's raw results, Jupyter Notebooks, and all steps to reproduce our evaluation as supplementary package. Furthermore, a live demo of our tool is available online.[1] We invite other researchers to extend our open-source software and jointly research this novel approach.

Video URL: https://youtu.be/MYAkRMWLVD8

***Index Terms*—program comprehension, software visualization, collaboration, software as a service, extended reality**

I. Introduction

Scientific works in the research field of software visualization (SV) often address program comprehension. There are various approaches [1] that overall investigate how to decrease the required resources for program comprehension such as time spent [2]–[4]. Most of these approaches and tools can be used only by a single user and / or do not convey a sense of collaboration. However, learning and working in teams, e.g., via pair programming [5], shared documentation [6], and face-to-face communication [7], can facilitate the comprehension tasks. In times of remotely working from home, co-located collaboration requires new approaches to enable location independence [8], but still must be enjoyable and useful. In the context of SV, team collaboration is a relatively unexplored aspect.

In this paper, we present our device-heterogeneous and collaborative SV approach for program comprehension. Our resulting tool ExplorViz enables location-independent co-exploration of 3D software cities [9], [10]. Users can employ our on-screen, virtual reality (VR), or augmented reality (AR) modes using common of-the-shelf devices. To the best of our knowledge, this is one of the first approaches that combines on-screen, VR, and AR to visualize shared software cities in a single web-based tool. For that reason, we conducted a initial user study to investigate our approach's enjoyment, sense of collaboration, and perceived usefulness. In that study, 20 participants collaboratively solved program comprehension tasks while using each mode consecutively. Despite the rather small number of participants, the results already show a promising future research opportunity for SV.

The remainder of this paper is structured as follows. We present envisioned usage scenarios of collaborative SV in Section II. Section III introduces our approach that has been evaluated by a initial user study as shown in the following Sections IV and V. The results of our study are discussed in Section VI. Section VII discusses related work to our contribution. Finally, we conclude this paper and present future work in Section VIII.

II. Envisioned Usage Scenarios for Collaborative Program Comprehension

This paper introduces our approach and initial evaluation for collaborative SVs used in program comprehension. We first introduce potential usage scenarios such that readers understand our vision for this emerging research. General properties of our approach (see Section III), such as device heterogeneity and location-independent collaboration, apply for all the following scenarios. Some of the following features, can be combined, are work in progress, or depend on the availability of specific hardware, such as commercially available AR glasses.

[1]https://explorviz.krause-sh.de

A. Ubiquitous Visualization for Software Development

In SV research, approaches are often developed to answer visualization-specific research questions. Without doubt, this is essential. However, the actual real world application for program comprehension is often incidental. In fact, we have to ask how to increase the acceptance and utilization of SVs (in professional environments), so that they become a considerable alternative to text-based tools in the comprehension task [4], [6], [11]. We should investigate properties such as a tool's deployment, ubiquity, and performance. These are properties which are usually difficult to achieve and maintain in a research-focused setting, but they have significant impact on the success of SV for program comprehension in the professional field. We envision SVs to be commonly included in development processes, comparable to tools like integrated development environments or version control systems. For example, data collection could be automatically triggered from within a continuous integration pipeline and would not require a manual feed-in or record procedure. Developers could then collaboratively explore the latest visualization snapshot of their program without the need to re-setup the required software stack.

B. Teaching and Knowledge Transfer

Teaching purposes are also mentioned in related works that investigate collaborative SVs (see Section VII). We envision a SV that can be used by new developers in a team to comprehend for example the use cases of the visualized software system. To facilitate this, they can select a single use case and explore its unfolding throughout the system. The SV might be enriched with text boxes that transfer additional knowledge such as source code or explanations. At any time, collaborators, e.g., instructors, can be invited or freely join the collaborative SV session to help the new developers with the comprehension task.

C. Team Meetings

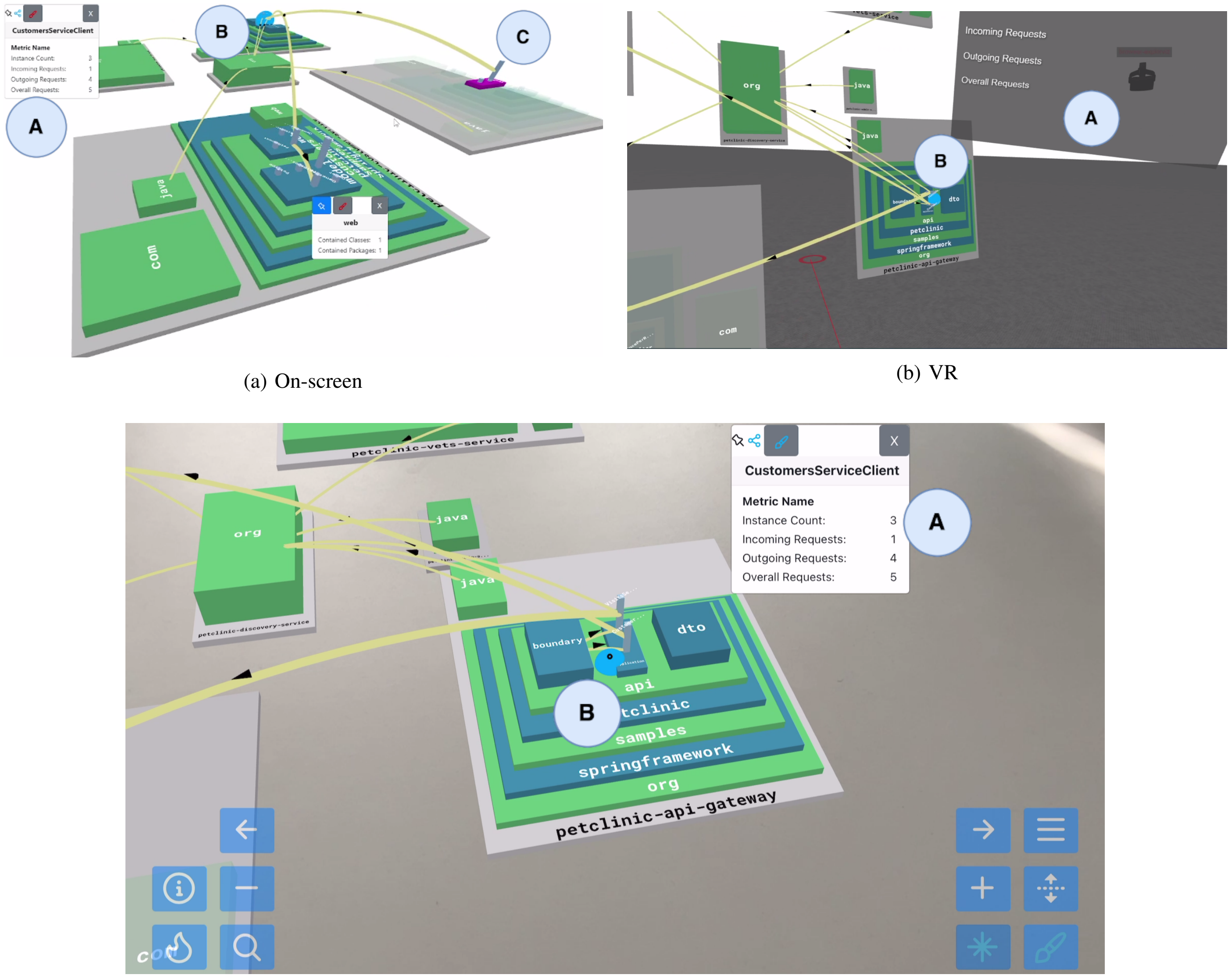
Meetings are used to discuss for example potential restructurings and feature realizations in the software as well as the evolution of a software system's code base. Those meetings could benefit from collaborative SVs, since nowadays they are often conducted as virtual meeting, therefore location-independent. As a result, they require some support for online discussions. We envision that collaborators use a SV as such basis. For example, they could interactively insert new classes, source code packages, and relationships, e.g., method calls, inside of the SV. Based on architecture conformance checking [12], the new architectural model could then be automatically transferred to tickets in an issue tracking system, e.g., as feature requests.

III. Approaching Collaborative Software Visualization - ExplorViz

ExplorViz was initially developed as monolithic Java application, in which users could separately explore the same SV using dynamic analysis as source for the visualization [13], [14]. From day one, ExplorViz was and still is a publicly available open source software.[2] We approached collaborative program comprehension via 3D-printed software cities in the past [15] and presented one of the earlier usages of VR [16]. Since 2020, we redevelop most of ExplorViz' functionality with modern cloud-native technologies to focus on collaborative SVs that are provided as SV as a Service (SVaaS) [17]. With SVaaS, we aim to achieve a ubiquitous SV environment, similar to the ideas of Section II-A. In the following, we introduce ExplorViz' new functionalities from a user's perspective and outline the most important implementation details. Therefore, we focus on the ExplorViz frontend component as it provides the actual SV and user interface. We refer readers to Krause-Glau and Hasselbring [17] for further technical information regarding ExplorViz' backend. In [17], we introduce its dynamic analysis procedure, microservice-based architecture, and employed cloud technologies. A demo of ExplorViz is online available and can be explored,[3] for example with collaborators over the internet.

Figure 1 depicts our collaborative SV modes that are provided by the web-based ExplorViz frontend component. The *modes* are *on-screen* (Figure 1a), *VR* (Figure 1b), and *AR* (Figure 1c). We chose to develop a browser application, since we can easier deploy new versions of our tool in the future. For example, there is no need to use an app store. Each visualization is rendered using Three.js[4], i.e., a JavaScript library for WebGL content. To begin, users navigate to a deployed version of the frontend component via their web browser, e.g., Google Chrome or Oculus Browser on the Meta Quest 2. After login, they can choose to open the SV of one of their (currently being) monitored software systems [17]. Alternatively, they can join a publicly available collaborative session. Regardless of the choice, users are forwarded to the on-screen SV of ExplorViz that can be seen in Figure 1a. If a collaborative session was initially selected by the user, the SV will be synchronized according to the session state. This means that for example shared information windows (popups) will be opened and pinned (Figure 1-A). By default, those windows are depicted only for a local user and depend on their interaction. For example, a single user might be interested in a specific method call and wants to comprehend further information, e.g., number of invocations during runtime, through a popup. If the user decides that this information is worth sharing, the popup can be opened for all remaining collaborators by clicking a single button. Figure 1 also shows the other collaborative features, namely pinging (Figure 1-B) and entity selection (also called highlighting) (Figure 1a-C). The ping is a temporary and in size fluctuating sphere that attracts the remaining collaborators' attention towards a visualization detail. The entity selection can be used to color a visualization detail based on the executing user's color. Furthermore, entities that are incoherent to the selected entity

[2]https://github.com/ExplorViz
[3]https://explorviz.krause-sh.de
[4]https://threejs.org


CustomersServiceClient

Metric Name	
Instance Count:	3
Incoming Requests:	1
Outgoing Requests:	4
Overall Requests:	5

web

Contained Classes: 1

Contained Packages: 1

(a) On-screen

(b) VR

(c) AR

Fig. 1: Different perspectives of employed devices during a collaborative session in a single room. The collaboration is location-independent, therefore, is also remotely usable over Internet.

are faded out.

The on-screen version of ExplorViz is also the gateway to our VR (Figure 1b) and AR modes (Figure 1c). They can be accessed through a context menu from the on-screen SV. At any time, users can switch the device and proceed with the exploration (while still being connected to the same collaborative session). Both XR modes are developed using the WebXR device API.[5] Therefore, ExplorViz can be used with the most common VR and AR equipment. Additionally, no change in the source code is required to support future VR and AR devices. WebXR is made up of different modules, for example the AR module. Each module encapsulates a feature of the API that might be provided by a browser. As a result, not every module is included in each browser.[6] For example, the AR module is available in the Microsoft Edge browser on the Hololens 2, but not usable in the Oculus browser at the time of writing. Looking in the future, this will most certainly change with the increasing maturity of VR Pass-through features. Those features enable (standalone) head-mounted displays to pass through the camera feed, ultimately resulting in a mixed reality environment.

IV. Experiment Design and Demographics

Principally, the idea for our approach reposes on collaboration and device-heterogeneity. We expect that a wider range of supported devices influences SVs' acceptance in general, since users can freely select their favored device(s). The ability

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

[6]https://immersiveweb.dev/#supporttable

TABLE I: Program comprehension tasks, two questions per mode.

Set ID	Question ID	Category	Question
QS1	Q1	Metric Analysis	Name the top two applications that have the highest cumulative count of classes and packages, respectively.
	Q2	Tool Insight	In your understanding, what is the “petclinic-discovery-service” application used for?
QS2	Q3	Structural Understanding	Name the class that you think represents the start of the ”Read Pet Owner Data“ use case.
	Q4	Tool Insight	The veterinary clinic would also like to offer holiday care for the animals. The software system should take over the appointment management for this. How would you implement this new functionality by extending and/or creating one or more applications?
QS3	Q5	Metric Analysis	Name the top two classes that have the most incoming method calls.
	Q6	Structural Understanding	The software does not properly execute the use case “Show list of vet visits for all pets owned by a pet owner”. What needs to be done to ensure that the use case is implemented correctly?

to change the device at any time and obtain a comparable SV while being reconnected to the same collaborative session might contribute factors such as enjoyment, sense of collaborators' presence, and recollection. In fact, those and many other variables ultimately influence the effectiveness and experience of virtual content of such SV approaches [1], [18], [19]. The main **contributions** of our evaluation are:

- Further insights regarding the perceived usefulness of software-city-based on-screen, VR, and AR modes.
- First quantitative and qualitative results regarding the perceived enjoyment and usefulness of collaborative and device-heterogeneous SV using modern input and output devices.
- A supplementary package containing the evaluation's raw results, screen recordings of all participants, and detailed instructions as well as software packages for replication [20].

These contributions have been achieved by conducting a user study to collect initial user feedback as basis for future refinement and research. In this study, we strove for collaborative feature evaluation and observable enjyoment as well as usefulness. Later, we will quantify the effectiveness of each mode, again, to refine each mode. For those reasons, our posed research questions are not concerned about effectiveness, but focus on qualitative aspects of our approach in the context of program comprehension:

- **RQ1:** Are the collaborative features perceived as useful in our collaborative and device-heterogeneous SV?
- **RQ2:** Is one mode, i.e., on-screen, VR, and AR, or a combination of modes favored over the others?
- **RQ3:** Does the collaborative and device-heterogeneous SV achieve a feeling of collaboration?
- **RQ4:** Is the collaborative and device-heterogeneous SV enjoyable?
- **RQ5:** Is a collaborative and device-heterogeneous software visualization environment perceived as useful in the context of program comprehension?

Prior to the user study, we conducted a pilot study with three colleagues to evaluate our research design. The following paragraphs also introduce any alterations that have been made due to the pilot study's results.

A. Target System and Tasks

At the time of writing, ExplorViz visualizes results of dynamic analysis. As a consequence, the visualization changes every ten seconds based on the analyzed and aggregated execution traces. Therefore, the visualization is dependent on the runtime behavior of the software system and ultimately on the executed use cases within this system, e.g., a user books an appointment. Combined with the features that are under observation in this study (RQ1), the amount of impressions for new users is overwhelming. For that reason, our study design shifts the visualization focus towards a “static” visualization of runtime behavior as shown in the following.

We used the distributed version of the Spring PetClinic[7] as target system. This version features seven Java-based microservices with auxiliary software such as Netflix's Eureka[8] for service discovery. As preparation for the study, we started the complete software stack of the distributed PetClinic while each service was configured to be instrumented and monitored with the inspectIT Ocelot Java agent. Then, we executed a set of use cases within the PetClinic's frontend and exported the resulting execution traces, i.e., a single snapshot that ExplorViz uses to aggregate multiple execution traces. This snapshot was then provided to our frontend by a mocked analysis component, resulting in a steady SV, therefore a “static” visualization of runtime behavior.

Table I presents the program comprehension tasks that the participants had to solve during the study. These types of questions are commonly used in SV research [21], [22]. The depiction of Table I is influenced by the works of Wettel et al. [23] and Moreno-Lumberas et al. [24]. In our case, every question was collaboratively solved while each participant used a distinct mode, i.e., device, in comparison to the other collaborators of the study session. Each mode was used to solve two subsequent questions. Our pilot study showed that more questions caused a too long duration for the overall study.

[7]https://github.com/spring-petclinic/spring-petclinic-microservices
[8]https://github.com/Netflix/eureka

The participation with six tasks already required between one and two hours. Therefore, we decided against more tasks to prevent discouraging potential participants. The `Set ID` of Table I represents two questions that were subsequently solved in one mode.

Our pilot study indicated difficulties for some of the tasks in Table I. For example, the PetClinic's discovery service only contains a single class called DiscoveryServerApplication. Without knowledge about Eureka or in general service discovery, participants might not understand the functionality of this service. Therefore, we manually enriched the snapshot with self-explanatory class names and adjusted the method calls accordingly.

B. Participants

We publicly invited computer science students of the Kiel University to participate in our user study. The participation was voluntary. All probands could sign up for random group assignment or participate with up to two fellow students or companioned graduates as predefined group. Each group had the chance to win one out of five 100 € gift cards for an e-commerce shop [25].

Distribution: The following information were collected during the mandatory pre-questionnaire of our study. The general procedure of the study will be discussed in the upcoming Section IV-C. Overall, the user study included 19 computer science students and one master's graduate (excluding the pilot study participants). Therefore, the number of participants is greater than the median participant count (13) and ranks among the most usual count in the related literature body [1]. The group of participants consisted of two master and 15 bachelor students. The remaining two persons were bachelor's student with the goal of becoming secondary school teachers. Eleven participants stated that they see software development as future working field.

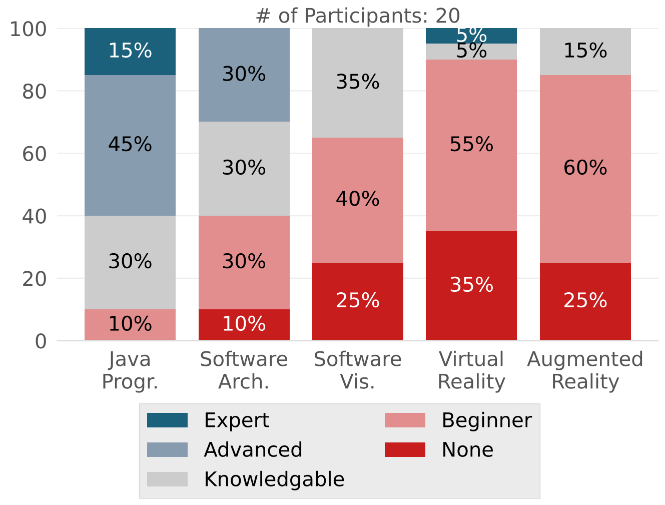


Fig. 2: Participants' reported experiences for different aspects.

TABLE II: Employed devices in the user study.

Mode	Device	Specifications
On-Screen	Desktop computer	CPU: Intel i5-6500 @ 3.2 GHz GPU: NVIDIA GeForce GTX 1070 Display Resolution: 1920x1080 Input device: Mouse
VR	Desktop computer & Oculus Rift S	Desktop specs as above, Rift S: Disp. Res.: 1280x1440 px per eye Input device: Oculus Touch v2
AR	iPad Pro 11 (2020)	Diplay size: 11 inches Display Resolution 1668x2388 px Camera Resolution: 12 MP Depth sensor: TOF 3D LiDAR Input device: Touchscreen

Experiences: Two participants completed an IT specialist apprenticeship prior to their studies. Seven participants developed open or closed-source software as free time activity and two persons worked as student employee. Eight participants used pair programming to develop software in the past. Due to their answer, these persons also had to fill out a Likert scale questionnaire to state their (dis)agreement toward pair programming. All eight participants strongly or at least agreed that this technique is useful and helps with knowledge transfer. Seven persons (strongly) agreed that this technique can improve a software's quality, while one person neither agreed nor disagreed. Two participants previously used ExplorViz. Further experiences are visualized in Figure 2, based on answering via semantic differential scales.

C. Procedure

The user study took place at the Kiel University and included two instructors (IN1 and IN2). Both, IN1 and IN2 develop ExplorViz, designed and conducted the user study, and authored this paper. We co-located all groups one after another in a single room to intervene if necessary. While our approach allows remote collaboration via internet, the study did not use a physical separation of participants and a voice chat software. Instead, all participants could speak out loud to their group members (voice communication). At any time, they could not see their group members screen. The employed devices are listed in Table II. Before an iteration of the study took place, all devices were put in a collaboration session, therefore ready to use. Upon arrival, all participants were welcomed and asked to take a seat in front of IN1. IN1 stated that questions were allowed to be asked at any time. Then, IN1 briefly explained what the participants were about to perform. IN1 and IN2 proceeded to introduce ExplorViz and its collaborative features based on the same SV that the participants were about to explore. For that, IN1 displayed the computer screen via video projector to all participants. Furthermore, IN1 briefly explained the purpose of the Spring PetClinic. This introduction did not explain profound insights of the software system, e.g., detailed use cases or task-related information.

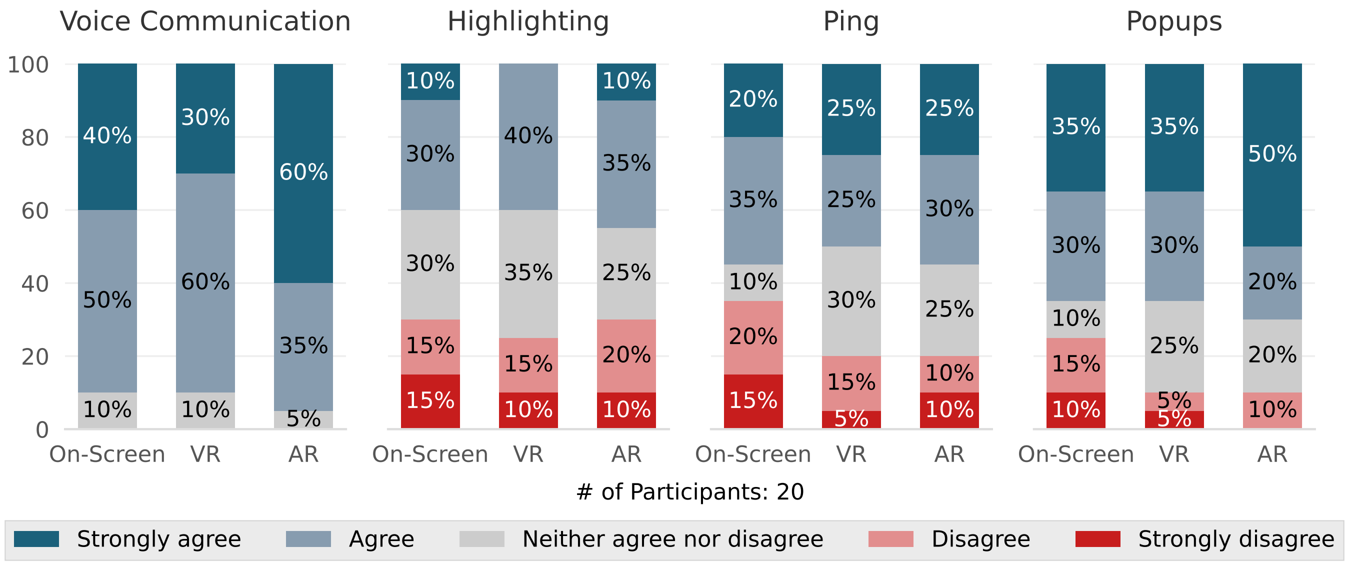


Fig. 3: Likert scale showing how much participants agree or disagree that a collaborative feature facilitated solving the tasks.

After the introduction, each participant consecutively selected an anonymized login credential for the survey app LimeSurvey.[9] LimeSurvey was used to collect the demography and user feedback during the study. The credential also determined in which mode a participant would start, i.e., On-Screen, VR, or AR. At any time, each mode was only used by one participant to focus on the device-heterogeneity. Therefore, the remaining participants of that group simultaneously used a different mode. Groups of two started with participants simultaneously using On-Screen and AR. The order in which the modes were consecutively used was On-Screen → VR → AR → On-Screen → ... Again, all modes were only used once for two tasks (see Section IV-A) by each participant while the remaining group members used a different mode, i.e., device-heterogeneity. IN1 asked all participants to go to the provided survey computers, login with the selected credential, and start to fill-out the pre-study (pre) questionnaire. After that, the first mid-study (mid) questionnaire indicated which mode a participant had to test first. Mid questionnaires were used to collect the users' feedback regarding the mode that was tested shortly before. We decided to collect this feedback after each mode, since there might be too many (new) impressions and emotions that participants might have forgotten in a post-study (post) questionnaire.

After all participants took a seat in front of their device, IN1 (On-Screen, AR) and IN2 (VR) helped all participants with the controls and collaboration features. IN1 asked all participants if they were comfortable with the controls and ready to answer the program comprehension tasks. Again, IN1 repeated that questions were allowed to be asked at any time. Then, IN1 read out aloud the first question. IN1 and IN2 used an online document to take notes during all tasks. Solutions were discussed between collaborators and also communicated aloud to the instructors, so that they could note down the answers in their online document. After two tasks, all participants were asked to leave their device and fill-out the mid questionnaire on their survey computer. This procedure was repeated twice. The last mid questionnaire was followed up by a post questionnaire that collected the participants' feedback regarding the collaboration approach in general. Finally, IN1 asked if there were any further remarks. If yes, those were again noted down in the instructors' memo. Overall, an iteration of the user study took about one and a half hours.

[9]https://www.limesurvey.org

V. Results

Our questionnaires included closed-ended and open-ended questions. Additionally, IN1 and IN2 made a note of their observations such as rational usages of a collaborative feature or emotions while the study took place. In the following, we present and use the quantitative and qualitative results of the mid and post questionnaire to answer our posed research questions (see Section IV).

RQ1: *Are the collaborative features perceived as useful in our collaborative and device-heterogeneous SV?*

Figure 3 visualizes the participants' evaluation of the collaborative features that they could use during the solution process. Each participant completed three mid questionnaires, i.e., one questionnaire per mode, therefore per task set and device (see Tables I and II). The results are accumulated over all three task sets, since we did not find unforeseen peculiarities for any feature in any combination of mode and task. Selectable answers were posed as 5-point Likert-type scale for the statement *The collaborative feature X helped me to solve the task set*, where *X* is one of the four collaborative features. Regardless of the evaluated mode, the stacked bar chart shows at least a slightly higher degree of overall agreement than disagreement or neutrality for the collaborative features voice communication (abbr. communication) and popups. In fact, Figure 3 presents that at least 90 % of the participants generally agree with the posed statement regarding communication, again regardless of the used mode. Not a single participant disagrees in this context. This is also recognizable in the feedback for the collaborative features that was collected during the mid questionnaires and asked for any further remarks. Looking at the instructors' memo, we also find multiple records of task-oriented discussions, therefore rational usages of communication, that took place among several groups.

We conclude that communication was the collaborative feature that was at most perceived as useful.

The results of the collaborative highlighting feature show an almost equal distribution of agreement, disagreement, or neither of both when comparing the modes. When focusing on a single mode, e.g., VR, we see that 40 % agree with the posed statement. In the qualitative data, some participants stated that they tend to use the ping feature instead of highlighting and therefore do not see a use case for the highlighting in the collaboration. The instructors' memo contains only few records of task-oriented collaborative highlighting usages, therefore correlates with Figure 3.

We conclude that the collaborative highlighting feature was perceived as not useful.

The collaborative ping feature shows a higher degree of disagreement for the on-screen mode in comparison to VR and AR. This is not perceptible in the qualitative data, since the users' feedback was entirely directed towards the usability of the different modes, rather than their usefulness for the collaboration. This is further discussed in Section VI. A higher degree of disagreement for the on-screen mode in comparison to VR and AR also applies for the popup feature. Once again, participants' evaluated the usability rather than usefulness for the collaboration. We recorded various requests to "share" or "ping" a visualization detail for both the collaborative ping feature as well as popup feature, respectively.

We conclude that both, the collaborative ping feature and the popup feature are perceived as useful.

RQ2: *Is one mode, i.e., on-screen, VR, and AR, or a combination of modes favored over the others?*

Figure 4 visualizes the results of a multiple choice question that asked all participants to state their favored mode(s). This question was included in the post questionnaire, therefore, after each participant consecutively evaluated each mode. Nine participants favored the collaborative on-screen version of the SV. Six different persons equally favored the collaborative VR mode. Three participants liked the VR mode the most. The combinations On-Screen + AR and VR + AR were each favored by one person. No participant liked the collaborative AR mode the most. Overall, the collaborative AR mode was the least favored one.

We conclude that the collaborative on-screen mode was favored at most.

This pattern of opinions can also be seen in the evaluated performance of each mode. Figure 5 depicts this evaluation, accumulated over the course of all tasks. Selectable answers were posed as 5-point Likert-type scale for the related statement *I had no problems with the performance of the used mode.* With 95 % the on-screen mode's performance was most positively perceived among all modes. One single participant strongly disagreed with the posed statement.

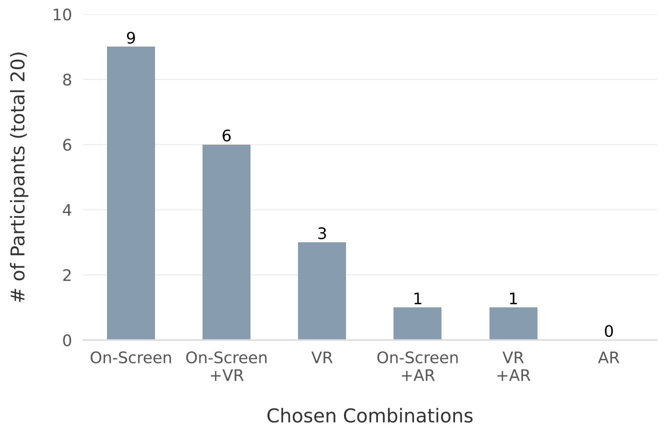


Fig. 4: Favored collaboration modes

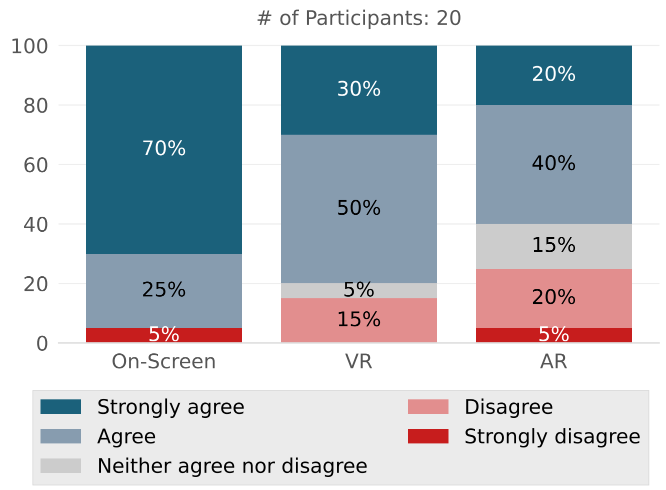


Fig. 5: Likert scale showing the participants' evaluation of the modes' performance.

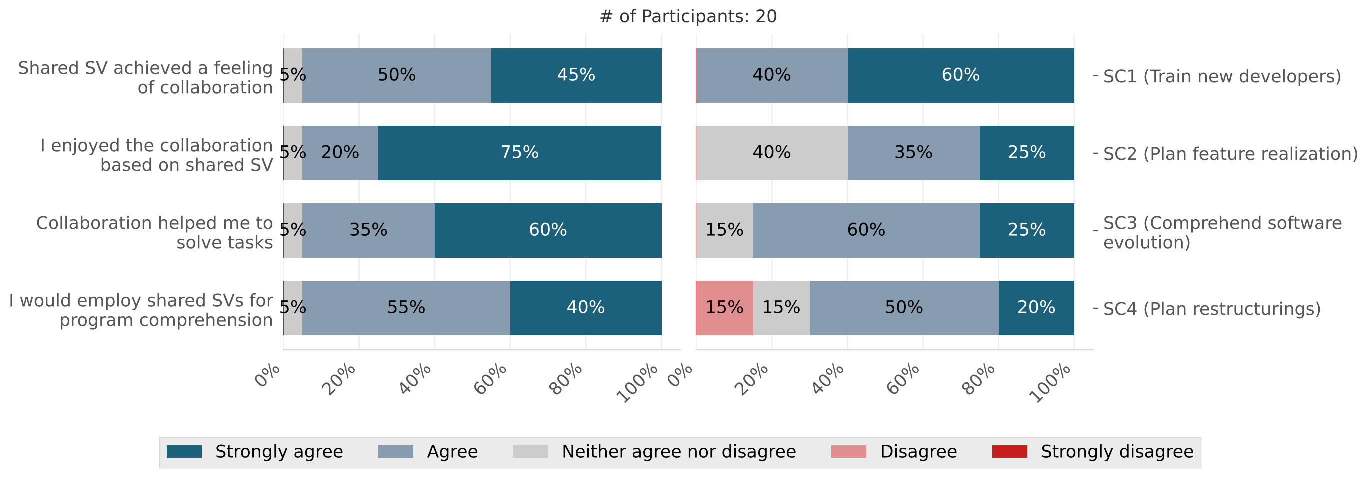


Fig. 6: Likert scale showing participants' evaluation of different statements (left side) and exemplary usage scenarios for collaborative SV (right side).

Regarding VR, 15 % of the participants disagree and 5 % neither agree nor disagree with the statement. AR has both a higher overall disagreement of 25 % and a higher count of neutral opinions with 15 %. As stated in the instructors' memo, some groups faced multiple crashes of the used WebXR Viewer, i.e., the Mozilla's WebXR-ready browser for Apple's iOS and iPadOS. This is also recognizable in some of the screen recordings of the employed iPad.

While there is a most favored collaborative mode, namely on-screen (see Section V), we overall conclude that our approach is enjoyable.

The remaining research questions are addressed by evaluating the results of the post questionnaire. It included four statements and four exemplary usage scenarios for collaborative SV that users had to evaluate. These were used to examine the perceived sense of collaboration, enjoyment, and usefulness. Potential answers for the statements and usage scenarios were again posed as 5-point Likert-scales. The results are depicted in Figure 6. The full textual descriptions of the usage scenarios are included in our supplementary package [20].

RQ3: *Does the collaborative and device-heterogeneous SV achieve a feeling of collaboration?*

Figure 6 depicts the statement *Shared SV achieved a feeling of collaboration* that participants had to evaluate. The result shows that 95 % generally agree with the statement. The distribution between strong and normal agreement is almost equal. One participant neither agrees nor disagrees with the statement.

RQ4: *Is the collaborative and device-heterogeneous SV enjoyable?*

Figure 6 shows that 95 % of the participants enjoyed working with the collaborative and device-heterogeneous SV. The instructor's memo includes only one recorded participant who often sighed due to wrong operation, especially in VR. There was not a single participant who thought that the collaborative and device-heterogeneous SV was not enjoyable.

RQ5: *Is a collaborative and device-heterogeneous software visualization environment perceived as useful in the context of program comprehension?*

The post questionnaire also included an optional open-ended question that asked all participants to state a potential usage scenario for collaborative SV. All 20 participants answered this optional question with their own ideas. The evaluation of the exemplary usage scenarios that are presented in Figure 6 followed afterwards, therefore were not known when answering the open-ended question. Most of the given answers can be mapped onto one or multiple of the posed usage scenarios. Regarding the latter, SC2 shows the least degree of overall agreement with 60 % and the highest degree of neither agreement nor disagreement with 40 %. Out of all scenarios, SC4 is the only scenario with 15 % disagreement.

We conclude that our envisioned potential of collaborative SV was also recognized by the participants in general.

VI. Discussion

In the following, we discuss the results of our user study.

A. Overall significance

Based on the quantitative and qualitative data, we evaluated different aspects such as usefulness and enjoyment of our approach. Since there are no replications of our evaluation so far, our initial results should be taken as indicators rather than evidences. With that being said, we require more comparable research in the field of collaborative SV. For example, an environment that simulates real world locations in which software can be comprehended might be more enjoyable (see Section VII). In that context, we could also conduct controlled experiments instead of mostly qualitative-focused user studies or use established measurements to quantify usefulness such as Davis' Technology Acceptance Model [26].

B. *Collaborative features*

Section V mentions that the qualitative data for the collaborative features ping and popups do not correspond to the results visualized in Figure 3. For example, the sole negative statement regarding the ping mentioned too big spheres in the on-screen version. Furthermore, the ping was often perceived as not visible in the VR and AR modes. The qualitative data of the evaluated popup feature shows that multiple participants were annoyed with the popup's trigger action in the on-screen version. While VR and AR used buttons to open a popup, the on-screen version uses mouse hovering. The screen recording of the on-screen mode visualizes the problematic behavior. When using the mouse, participants tend to follow the visualization entities with the cursor. This resulted in unwanted popups. Overall, we conclude that various participants qualitatively evaluated the usability or intuitiveness of the features rather than the perceived usefulness in the context of collaboration. This is also supported by assessing the qualitative data. We obtained various feature requests and potential improvements for the collaborative features that concern usability, not usefulness.

C. *Favored mode and the (users') performance*

The results show a correlation between favored mode(s) and the evaluated performance. We classify the one participant that showed a strong dislike of the on-screen mode as outlier. An equally strong dislike, but overall higher dislike, can be found in the AR performance. It seems that the AR mode's performance or possibly an unobserved variable, e.g., enjoyment while using a mode, heavily influenced the evaluation of the AR mode. This is also supported by assessing the qualitative data.

Various participants mentioned that the training period in VR took most of the time, compared to AR and VR. Furthermore, the instructors' memo includes several records of VR participants asking for the right button to enable a feature. Only a single participant stated that the iPad's size makes it unpleasant to hold in a short period of time. The evaluation of the AR mode might outcome differently with the future availability of commercially available AR glasses

D. *Threats to Validity*

Internal validity:

Testing: A preceding mid questionnaire might have affected the results of the subsequent mid questionnaires. For example, participants might have used the first mid questionnaire as baseline for the remaining two questionnaires. Overall, this might have affected the assessment of the collaborative features (Figure 3) that were used to answer RQ1. To mitigate this threat, we used a textual hint in the post questionnaire which pointed out that participants' could go back to their previous answers for any adaptions.

Experiences: Different experiences of the participant's might have influence the perceived usefulness of the approach (Figure 6). In fact, 75 % of the participants were undergraduates and the average experience level regarding SV (Figure 2) was 2.1 on a scale from one to five (one being used to indicate no experience at all). However, our study was intended to evaluate primarily qualitative aspects of our approach and therefore should be taken as indication rather than evidence (see the discussion in Section VI-A). With that being said, we conclude that more empirical research is required in this context.

Distribution: Different group compositions might have influence the outcome of our evaluation. To mitigate this threat, we initially invited all computer science students of Kiel University to participate in our study. Groups that were not randomly assigned showed different social interaction during the course of the study as was also observed in the instructors' memo [20].

External validity:

Sample Size: With 20 participants, our study's sample size is higher than the median participant count (13) and ranks among the most usual sample sizes in the related SV literature body [1]. However, since this our first study to examine device-heterogeneous and collaborative SV, the results have to be validated with larger samples rate (see also Section VI-A).

Participants: In practice, developers with diverse experience levels face the task of program comprehension. Our study uses mostly computer science students to evaluate our collaborative SV approach. The use of students in software engineering research is a suitable practice to evaluate approaches, although it is a laboratory simplification of reality [27], [28]. Since there are no prior results to build upon, we are initially concerned with qualitative aspects or our approach. However, professional developers might evaluate the approach differently. With reference to Merino et al. [1], we plan to conduct a case study in a real-world professional environment as future work.

VII. Related Work

The research field of program comprehension through SVs is profound. Therefore, related work can be divided into different categories. For example, there are various SV approaches that employ or originally introduced the same visualization metaphor [10], [29], [30]. Other works use comparable hardware, e.g., extended reality headsets [18], [24], [31], or similar data sources for the visualizations [32]–[34]. More recently, collaboration in extended reality has also been researched in different visualization topics [35].

Due to the large amount of work in the general research field of visualization, we focus on the single most related collaborative SV approach for program comprehension by Koschke and Steinbeck [36]. Their publication presents a detailed overview of previous works in the context of collaborative SV, e.g., [37],

[38]. Therefore, we point readers towards their work to review the history of collaborative SV for program comprehension. Other works [8], [39]–[42] can only be mentioned here due to space restrictions.

Koschke and Steinbeck presented their multi-user multi-purpose SV platform called Software Engineering Experience (SEE) [36]. SEE enables multiple users to collaboratively comprehend software in a shared and device-heterogeneous environment. Regardless of the real world location, users are depicted as avatars in a virtual room that includes multiple visualized software cities on top of rendered desks or tables. Here, each user sees other collaborators and can interact via voice chat. While the table-based visualization is comparable to for example IslandViz [39], SEE enhances this idea and introduces each table for a specific use case. As a result, the same software city can be differently layouted for architecture conformance checking, debugging, performance analysis, and quality assessment.

SEE and Explorviz share the same overarching idea, i.e., collaborative and device-heterogeneous SV for program comprehension. However, their implementations and designs are divergent. For example, Koschke and Steinbeck mention that they are also working on supporting AR devices. ExplorViz provides an AR mode that can be accessed via off-the-shelf mobile devices. Powered by WebXR, this mode should also be ready-to-use with for example Microsoft's HoloLens.[10] SEE is developed in C# and built upon the Unity game engine,[11] a common pick for SV approaches. The ExplorViz frontend component uses WebGL and WebXR such that the complete SV interaction can be executed in standard web browsers such as Google Chrome or Mozilla Firefox. Nowadays and for this type of software, it is most often the developers choice which rendering framework they employ. The reason behind this is that most of the time the same devices are supported by today's rendering frameworks. However, the resulting artifacts of the software are differently deployed. For example, we can rollout new versions of our tool without the need of an app store. The two tools also differ in terms of data source. ExplorViz is not limited to visualize a set of pre-recorded execution traces, but can also provide live trace visualization. For that reason, our backend is built upon the Quarkus framework. This enables us to horizontally scale out for example our analysis services, so that we can adopt to varying load [17] (see Section II).

Regarding the design, SEE's SVs are surrounded by a virtual environment that simulates real world locations such as a library with book shelves and chairs. Regardless of the employed device, namely computers using common screens, tablets, and VR equipment, the interaction with the SVs and collaborators takes place in this environment. When comparing those real-world-simulating environments, there are indeed very recent results that indicate more pleasurable VR experiences when using colorful interventions or nature elements [43]. ExplorViz's VR mode also introduces a visualization of collaborators in the VR space. However, our virtual environment is less-detailed. As with our on-screen and AR modes, the focus is the SV itself. For that reason, we decided that the remaining modes are not included in the same virtual space. Instead, ExplorViz promotes the sense of collaborators' presence via a set of globally defined events that are synchronized throughout all connected session devices. Overall, our modes follow the most common device-related application types and do not share a virtual environment.

VIII. Conclusions & Future Work

In this paper, we presented our approach for device-heterogenous and collaborative SV for program comprehension. We described our envisioned usage scenarios and introduced our approach's implementation. Our resulting tool ExplorViz provides web-based on-screen, VR, and AR modes that can jointly be used to explore software city-based visualizations. We expect that a wider range of supported devices influences SVs' acceptance in general. Free selection of employable devices while being reconnected to the same collaborative session contributes factors such as enjoyment, sense of collaborators' presence, and recollection. In fact, those and many other variables ultimately influence the effectiveness and experience of virtual content such as SV approaches [1], [18], [19].

We conducted an initial user study to collect qualitative feedback regarding our approach's perceived usefulness and enjoyment. The study was conducted at Kiel University and included 20 participants forming groups of two or three. The groups collaboratively explored a given SV to solve common program comprehension tasks. Additionally, they completed questionnaires to state their feedback. Equipped with the results, we see that for example 19 participants (greatly) enjoyed the collaboration, felt a presence of their collaborators, and would employ collaborative SVs for program comprehension.

Regarding future work, we use the collected results and feedback to refine our modes. The qualitative feedback of the participants included various feature requests and potential improvements. Those are visible in a compiled list that is also included in our supplementary package [20]. For example, arrows indicating the direction where a ping was executed were often requested. Also for future work, we strive for more empirical evaluation. For example, the study could be replicated with the enhanced tooling. Additionally, we want to follow the guidelines posed by Merino et al. [1] to evaluate the effectiveness of our approach by means of a case study with professional software developers. Other controlled experiments might compare each mode against each other or the overall approach against for example the work of Koschke and Steinbeck [36].

[10]https://docs.microsoft.com/en-us/windows/mixed-reality/develop/javascript/webxr-overview

[11]https://unity.com

References

[1] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz, "A systematic literature review of software visualization evaluation," *Journal of Systems and Software*, vol. 144, no. C, p. 165–180, oct 2018. DOI: 10.1016/j.jss.2018.06.027. [Online]. Available: https://doi.org/10.1016/j.jss.2018.06.027

[2] D. Robson, K. Bennett, B. Cornelius, and M. Munro, "Approaches to program comprehension," *Journal of Systems and Software*, vol. 14, no. 2, pp. 79–84, 1991. DOI: https://doi.org/10.1016/0164-1212(91)90092-K Software Maintenance. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016412129190092K

[3] K. Bennett, V. Rajlich, and N. Wilde, "Software evolution and the staged model of the software lifecycle," *Advances in Computers*, vol. 56, pp. 1–54, 2002. DOI: 10.1016/S0065-2458(02)80003-1

[4] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li, "Measuring program comprehension: A large-scale field study with professionals," *IEEE Transactions on Software Engineering*, vol. 44, no. 10, pp. 951–976, 2018. DOI: 10.1109/TSE.2017.2734091

[5] M. Bandukda and Z. Nasir, "Efficacy of distributed pair programming," in *Proceedings of the 2nd International Conference on Information and Emerging Technologies (ICIET 2010)*, 2010. DOI: 10.1109/ICIET.2010.5625667 pp. 1–6.

[6] R. Tiarks and T. Roehm, "Challenges in program comprehension," *Softwaretechnik-Trends*, vol. 32, no. 2, 2012. [Online]. Available: http://pi.informatik.uni-siegen.de/stt/32_2/01_Fachgruppenberichte/SRE_TAV/02-tiarks.pdf

[7] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, "How do professional developers comprehend software?" in *Proceedings of the 34th International Conference on Software Engineering (ICSE 2012)*. IEEE Press, 2012. DOI: 10.1109/ICSE.2012.6227188 pp. 255—-265.

[8] J. Dominic, B. Tubre, C. Ritter, J. Houser, C. Smith, and P. Rodeghero, "Remote pair programming in virtual reality," in *Proceedings of the 36th IEEE International Conference on Software Maintenance and Evolution (ICSME 2020)*, 2020. DOI: 10.1109/ICSME46990.2020.00046 pp. 406–417.

[9] C. Knight and M. Munro, "Comprehension with[in] virtual environment visualisations," in *Proceedings of the 7th International Workshop on Program Comprehension*, 1999. DOI: 10.1109/WPC.1999.777733 pp. 4–11.

[10] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *Proceedings of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 2007. DOI: 10.1109/VISSOF.2007.4290706 pp. 92–99.

[11] J. Siegmund, "Program comprehension: Past, present, and future," in *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2016)*, vol. 5, 2016. DOI: 10.1109/SANER.2016.35 pp. 13–20.

[12] F. Fittkau, P. Stelzer, and W. Hasselbring, "Live visualization of large software landscapes for ensuring architecture conformance," in *Proceedings of the 2014 European Conference on Software Architecture Workshops (ECSAW 2014)*. ACM, 2014. DOI: 10.1145/2642803.2642831 pp. 28:1–28:4.

[13] F. Fittkau, A. Krause, and W. Hasselbring, "Software landscape and application visualization for system comprehension with ExplorViz," *Information and Software Technology*, vol. 87, pp. 259–277, Juli 2017. DOI: 10.1016/j.infsof.2016.07.004

[14] W. Hasselbring, A. Krause, and C. Zirkelbach, "ExplorViz: Research on software visualization, comprehension and collaboration," *Software Impacts*, vol. 6, Nov. 2020. DOI: 10.1016/j.simpa.2020.100034

[15] F. Fittkau, E. Koppenhagen, and W. Hasselbring, "Research perspective on supporting software engineering via physical 3D models," in *Proceedings of the 3rd IEEE Working Conference on Software Visualization (VISSOFT 2015)*, 2015. DOI: 10.1109/VISSOFT.2015.7332422 pp. 125–129.

[16] F. Fittkau, A. Krause, and W. Hasselbring, "Exploring software cities in virtual reality," in *Proceedings of the 3rd IEEE Working Conference on Software Visualization (VISSOFT 2015)*, 2015. DOI: 10.1109/VISSOFT.2015.7332423 pp. 130–134.

[17] A. Krause-Glau and W. Hasselbring, "Scalable Collaborative Software Visualization as a Service," in *10th IEEE International Conference on Cloud Engineering (IC2E 2022)*, Pacific Grove, California, USA, Sep. 2022, (in press).

[18] L. Merino, A. Bergel, and O. Nierstrasz, "Overcoming issues of 3D software visualization through immersive augmented reality," in *Proceedings of the 6th IEEE Working Conference on Software Visualization (VISSOFT 2018)*, 2018. DOI: 10.1109/VISSOFT.2018.00014 pp. 54–64.

[19] D. Heidrich, A. Wohlan, and M. Schaller, "Perceived speed, frustration and enjoyment of interactive and passive loading scenarios in virtual reality," in *HCI International 2020 – Late Breaking Papers: Virtual and Augmented Reality*, C. Stephanidis, J. Y. C. Chen, and G. Fragomeni, Eds. Cham: Springer International Publishing, 2020. ISBN 978-3-030-59990-4 pp. 343–355.

[20] A. Krause-Glau, M. Bader, and W. Hasselbring, "Supplementary data for: Collaborative software visualization for program comprehension," 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6795801

[21] L. Merino, J. Fuchs, M. Blumenschein, C. Anslow, M. Ghafari, O. Nierstrasz, M. Behrisch, and D. A. Keim, "On the impact of the medium in the effectiveness of 3d software visualizations," in *Proceedings of the 5th IEEE Working Conference on Software Visualization (VISSOFT 2017)*, 2017. DOI: 10.1109/VISSOFT.2017.17 pp. 11–21.

[22] M. Pacione, M. Roper, and M. Wood, "A comparative evaluation of dynamic visualisation tools," in *Proceedings of the 10th Working Conference on Reverse Engineering (WCRE 2003)*, 2003. DOI: 10.1109/WCRE.2003.1287239 pp. 80–89.

[23] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities: a controlled experiment," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011)*, 2011. DOI: 10.1145/1985793.1985868 pp. 551–560.

[24] D. Moreno-Lumbreras, R. Minelli, A. Villaverde, J. M. González-Barahona, and M. Lanza, "Codecity: On-screen or in virtual reality?" in *Proceedings of the 9th IEEE Working Conference on Software Visualization (VISSOFT 2021)*, 2021. DOI: 10.1109/VISSOFT52517.2021.00011 pp. 12–22.

[25] C. F. Camerer and R. M. Hogarth, "The effects of financial incentives in experiments: A review and capital-labor-production framework," *Journal of Risk and Uncertainty*, vol. 19, no. 1, pp. 7–42, 1999. DOI: 10.1023/A:1007850605129. [Online]. Available: https://doi.org/10.1023/A:1007850605129

[26] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989. DOI: 10.2307/249008

[27] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects—a comparative study ofstudents and professionals in lead-time impact assessment," *Empirical Softw. Engg.*, vol. 5, no. 3, p. 201–214, nov 2000. DOI: 10.1023/A:1026586415054. [Online]. Available: https://doi.org/10.1023/A:1026586415054

[28] D. Falessi, N. Juristo, C. Wohlin, B. Turhan, J. Münch, A. Jedlitschka, and M. Oivo, "Empirical software engineering experts on the use of students and professionals in experiments," *Empirical Softw. Engg.*, vol. 23, no. 1, p. 452–489, feb 2018. DOI: 10.1007/s10664-017-9523-3. [Online]. Available: https://doi.org/10.1007/s10664-017-9523-3

[29] F. Pfahler, R. Minelli, C. Nagy, and M. Lanza, "Visualizing evolving software cities," in *Proceedings of the 8th IEEE Working Conference on Software Visualization (VISSOFT 2020)*, 2020. DOI: 10.1109/VISSOFT51673.2020.00007 pp. 22–26.

[30] C. L. Jeffery, "The city metaphor in software visualization," in *Proceedings of the 27th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2019)*, 2019. [Online]. Available: http://hdl.handle.net/11025/35620

[31] J. Dominic, B. Tubre, J. Houser, C. Ritter, D. Kunkel, and P. Rodeghero, "Program comprehension in virtual reality," in *Proceedings of the 28th International Conference on Program Comprehension*, ser. ICPC '20. New York, NY, USA: Association for Computing Machinery, 2020. DOI: 10.1145/3387904.3389287 pp. 391—-395.

[32] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 684–702, 2009. DOI: 10.1109/TSE.2009.28

[33] V. Dashuber and M. Philippsen, "Trace visualization within the software city metaphor: A controlled experiment on program comprehension," in *Proceedings of the 9th IEEE Working Conference on Software Visualization (VISSOFT 2021)*, 2021. DOI: 10.1109/VISSOFT52517.2021.00015 pp. 55–64.

[34] O. Benomar, H. Sahraoui, and P. Poulin, "Visualizing software dynamicities with heat maps," in *Proceedings of the 1st IEEE Working*

Conference on Software Visualization (VISSOFT 2013), 2013. DOI: 10.1109/VISSOFT.2013.6650524 pp. 1–10.

[35] D. Heidrich, A. Meinecke, and A. Schreiber, "Towards a Collaborative Experimental Environment for Graph Visualization Research in Virtual Reality," in *EuroVis 2021 - Posters*, J. Byška, S. Jänicke, and J. Schmidt, Eds. The Eurographics Association, 2021. DOI: 10.2312/evp.20211068. ISBN 978-3-03868-144-1

[36] R. Koschke and M. Steinbeck, "See your clones with your teammates," in *Proceedings of the 15th IEEE International Workshop on Software Clones (IWSC 2021)*, 2021. DOI: 10.1109/IWSC53727.2021.00009 pp. 15–21.

[37] P. Isenberg, N. Elmqvist, J. Scholtz, D. Cernea, K.-L. Ma, and H. Hagen, "Collaborative visualization: Definition, challenges, and research agenda," *Information Visualization*, vol. 10, pp. 310–326, 10 2011. DOI: 10.1177/1473871611412817

[38] C. Anslow, S. Marshall, J. Noble, and R. Biddle, "SourceVis: collaborative software visualization for co-located environments," in *Proceedings of the 1st IEEE Working Conference on Software Visualization (VISSOFT 2013)*, 2013. DOI: 10.1109/VISSOFT.2013.6650527 pp. 1–10.

[39] 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. DOI: 10.1109/AERO.2019.8742198 pp. 1–12.

[40] A. Henrysson, M. Billinghurst, and M. Ollila, "Face to face collaborative ar on mobile phones," in *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*, 2005. DOI: 10.1109/ISMAR.2005.32 pp. 80–89.

[41] F. Jung, V. Dashuber, and M. Philippsen, "Towards Collaborative and Dynamic Software Visualization in VR," in *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: IVAPP*, J. B. Andreas Kerren, Christophe Hurter, Ed. Portugal: SciTePress, 2020. DOI: 10.5220/0008945201490156 pp. 149 – 156.

[42] B. Scott-Hill, C. Anslow, J. Ferreira, M. Kropp, M. Mateescu, and A. Meier, "Visualizing progress tracking for software teams on large collaborative touch displays," in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2020)*, 2020. DOI: 10.1109/VL/HCC50065.2020.9127286 pp. 1–5.

[43] A. Batistatou, F. Vandeville, and Y. N. Delevoye-Turrell, "Virtual reality to evaluate the impact of colorful interventions and nature elements on spontaneous walking, gaze, and emotion," *Frontiers in Virtual Reality*, vol. 3, 2022. DOI: 10.3389/frvir.2022.819597. [Online]. Available: https://www.frontiersin.org/article/10.3389/frvir.2022.819597