Empirical Software Engineering (2022) 27:173
https://doi.org/10.1007/s10664-022-10196-5

The human experience of comprehending source code in virtual reality

James Dominic[1] · Brock Tubre[1] · Deborah Kunkel[1] · Paige Rodeghero[1]

Accepted: 15 June 2022 / Published online: 20 September 2022

Abstract
Virtual reality (VR) is an emerging technology used in various domains such as medicine, psychotherapy, architecture, and gaming. Recently, software engineering researchers have started to explore virtual reality as a tool for programmers. However, few studies examine source code comprehension in VR. This paper explores the human experience of comprehending source code in VR and compares it to source code comprehension in a desktop environment. We conducted a study with 26 graduate student programmers. We measured actual productivity, perceived productivity and used the NASA Task Load Index (TLX) survey to measure various factors such as mental demand, physical demand, temporal demand, performance, effort, and frustration. We found that the programmers experienced more physical demand, effort, and overall task load when reading and comprehending code in VR. However, we did not observe any statistically significant differences in the programmers' measured productivity or perceived productivity between VR and desktop comprehension.

Keywords Code comprehension · Human experience · Virtual reality

1 Introduction

Several domains such as education, medicine, and architecture use Virtual Reality (VR) (Taxén and Naeve 2002; Kurumi and Morikawa 2016; Pourchera et al. 2018; Nazligul et al.

Communicated by: Janet Siegmund

✉ James Dominic
domini4@clemson.edu

Brock Tubre
btubre@clemson.edu

Deborah Kunkel
dekunke@clemson.edu

Paige Rodeghero
prodegh@clemson.edu

[1] Clemson University, Clemson, SC, USA

2017; Racz and Zilizi 2018; Hayes et al. 2018). Software engineering researchers have recently started to study the benefits of VR for software developers and researchers (Gulec et al. 2018; Zirkelbach et al. 2019; Akbulut et al. 2018; Sharma et al. 2018). Ruvimova *et al.* found that VR reduces unnecessary distractions faced by software engineers (Ruvimova et al. 2020). Researchers have also started to explore how programmers can communicate and collaborate in VR (Bierbaum et al. 2001; Vincur et al. 2017; Aleotti et al. 2004).

For programmers, one of the most challenging and time-consuming tasks is comprehending source code (Steinmacher et al. 2016; Al-Saiyd 2017; Bacher et al. 2017; Khomokhoana and Nel 2019). Current solutions to address these problems include visualizations to display code, enhanced syntax highlighting, and code comprehension through games (Merino et al. 2018; Oberhauser and Lecon 2017; Wiese et al. 2019). Programmers also find that collaboration leads to better code comprehension (Arisholm et al. 2007).

The COVID-19 pandemic has forced many programmers to work from home. Brynjolfsson *et al.* has estimated that 50.2% of the workforce was remote during 2020 (Brynjolfsson et al. 2020). At the end of 2020, Upwork Inc. estimated that 26% of the total workforce would be working remotely in 2021, and an estimated 22% would continue to work remotely in 2025 (Upwork Study Finds 22% of American Workforce Will Be Remote by 2025: Upwork 2020). Ralph *et al.* found that employee well-being and productivity are closely related and are adversely affected by the sudden shift to work from home (Ralph et al. 2020). They suggest that employers should focus on improving employee well-being and the ergonomics of the home office space, such as providing proper office furniture.

In recent years, there has been a renewed interest in digital spaces. A metaverse is a virtual world that allows multisensory interactions through VR and Augmented Reality (AR) (Mystakidis 2022). Link Trainer, an analog flight simulator, is often considered the precursor to the present-day metaverse (Jeon 2015). Various metaverse platforms have already started offering virtual office spaces (Future of remote work? here's the reality of working in the metaverse 2022). Facebook launched Horizon Workrooms, a business meeting space accessed through their Oculus series of VR headsets (Workrooms: VR for business meetings n.d.). These developments and the future possibilities heighten the need to understand the programmer's experience during source code comprehension in VR. As the software development industry is exploring the use of VR for workspaces, we academics should explore this space more and provide research outcomes that further advances our understanding of this relatively new paradigm.

In this paper, we define the human experience of programmers during code comprehension as to how they experience frustration, mental effort, and physical effort. We study graduate student programmers comprehending source code in both a desktop computer setup and a virtual environment. We recruited 26 programmers to participate in this study. The program comprehension tasks included asking programmers to read source code, provide the code output, and write a plain English text summary of the code. In the study, 13 programmers worked in VR, and the other 13 programmers used the desktop computer setup. We asked programmers in both studies to complete as many comprehension tasks as possible from 8 different code snippets in 40 minutes. We found that programmers faced many challenges comprehending source code in VR. Programmers in VR reported lower concentration levels and higher levels of physcial demand, and effort. However, we found that the VR programmers' perceived productivity and measured productivity was not significantly different from that of the desktop programmers.

This paper is an extension of the Program Comprehension in Virtual Reality paper (Dominic et al. 2020a). In addition to the details discussed in the publication, we added four

more research questions, provided statistical analysis for eight more variables, performed Benjamini-Hochberg adjustment for the p-values, added Cohen's *d* effect sizes, and Spearman correlation analysis. We expanded the paper from four pages to 24 pages, including more details about the experiment setup and an in-depth discussion of the results and lessons learned while conducting this experiment. We discuss the results in the context of the added data and provide recommendations for future research in virtual environments for software engineering.

2 The Problem

We address a gap in the current program comprehension literature by exploring the productivity and perceived productivity of programmers comprehending source code in VR. As a community, we need to understand if the benefits of using VR as a tool for software developers outweigh the risks (e.g., fatigue). To better understand the human experience of comprehending source code in VR, we needed to first expand upon our previous work on program comprehension in VR (Dominic et al. 2020a; b). It is important to note that as a community, we are starting to explore various uses of VR (Ruvimova et al. 2020; Merino et al. 2018; Elliott et al. 2015a). For example, Merino *et al.* demonstrated the use of VR for explaining software architecture in the form of city buildings (Merino et al. 2018). Shepherd *et al.* studied the reduction of distractions in VR when used at the workplace (Ruvimova et al. 2020). Parnin *et al.* explored the affordances, possible applications, and challenges VR brings to software engineering (Elliott et al. 2015a).

This paper advances the SE research community's understanding of program comprehension in VR. We have not taken the risks that other fields have for years. Many fields have found benefits to using VR for training and education (Sattar et al. 2020; Martin-Gutierrez et al. 2017; Wang et al. 2018; Hayes and Johnson 2019). Other fields have found benefits from using VR for collaborations (Hoppe et al. 2018; Guerin and Hager 2017). Unfortunately, there is still little work on software engineers working in VR. Even though we recognize program comprehension as one of the most significant factors for success, we have not spent the time to explore the potential use of VR in enabling program comprehension.

Remote work has become significantly prominent in the software engineering field during COVID-19 (O Connor et al. 2022). This means some programmers never enter their company's physical office location. Remote programmers use Slack, Microsoft Teams, and various other tools for collaboration. Unfortunately, this solution is problematic. Remote programmers still find it challenging to collaborate with these tools (Miller et al. 2021). The more geographically dispersed a team gets, the more negative impact it has on team performance and software quality (Nguyen-Duc et al. 2015).

Various solutions, such as using agile practices, have been introduced in the past (Hossain et al. 2009). One solution is to use VR to bring programmers into the same environment for collaboration. If code comprehension and the human experience are similar or better in VR, it is worth further exploring for programmers. Software development teams could use VR as a tool for future training and onboarding of new programmers to software development teams. They could also use VR to facilitate pair programming sessions where an experienced programmer collaborates remotely with a new programmer to comprehend source code and program together.

3 Background and Related Work

In this section, we discuss the background and related work on program comprehension environments, virtual reality, VR in software engineering, and the human experience during comprehension.

3.1 Program comprehension environments

Many program comprehension environments have been built and explored (Williams et al. 2000). The focus has been on computer environments that display code in a unique fashion. Panas *et. al* proposed the software city metaphor to visualize software as a city for easy understanding of code structure (Panas et al. 2003). These visualizations were made on a 2D screen and was considered a high cost activity during its introduction in 2003. With the introduction of relatively affordable VR headsets such as the Oculus Rift, the research community replicated the software city metaphor in VR. Programmers found gesture based translation, rotation and selection of these software cities highly usable in virtual environments (Fittkau et al. 2015). Many researchers have developed visualizations of source code that help programmers quickly understand the source code at a high level (Kircher et al. 2001; Baheti et al. 2002). There are a few studies on the physical environment that programmers work in for comprehension (Schenk 2018). Johnson *et al.* studied distractions in the workplace. They found that programmers who were distracted and had to context switch were less productive than their counterparts (Johnson et al. 2003). However, Grubert *et al.* found that VR environments limit potential distractions in the workplace (Grubert et al. 2018). These characteristics of VR indicate that there is potential for VR to be used as an effective code comprehension platform.

3.2 Virtual Reality

Virtual Reality has entered many domains outside of software engineering. Many educators focus on teaching VR development to students as young as 7-14 years old and have found success in developing VR curriculum for students of all ages (Häfner et al. 2013). Some children with autism spectrum disorder responded positively towards the use VR and are able to learn new thing when information is presented using VR (Parsons and Cobb 2011). Hoffmann *et al.* desinged a VR system used to provide immersive tours of remote laboratories from a virtual theater (Hoffmann et al. 2016). This system provides the users a different perspective and opportunity to interact using natural means (using their hands and gestures). Pellas *et al.* conducted a literature review of 41 research studies using VR for education published between 2009 and 2019 (Pellas et al. 2020). Several of them reported positive learning outcomes with the use of VR. Students in one study were able to fix real-world electrical circuits using the knowledge from VR training. However, some studies also reported problems with using VR, such as a 3D interface that is not always well suited for learning applications and limited collaboration capabilities.

VR is also used in robotics and manufacturing to simulate as well as evaluate the behaviour of various robotic systems and humans. Matsas *et al.* used VR to evaluate the behaviour of humans while interacting with robots to perform manufacturing tasks (Matsas and Vosniakos 2017). Many fields such as medicine, architecture, and entertainment employ

VR technologies. Psychotherapists use VR to help clients overcome phobias, such as the fear of heights (Freeman et al. 2017). Hodges *et al.* demonstrated that participates exhibited acrophobia when exposed to increased heights in VR (Hodges et al. 1995). VR is used in medical schools to train students on surgical procedures (Ullrich and Kuhlen 2012). Architects use VR to create and model buildings. Architects can interact with their clients, discuss design, and preview a design before building (Bozgeyikli et al. 2016; Sampaio 2018; Johansson et al. 2014; Graham et al. 2018; Kreutzberg 2015). Businesses use VR for meetings with remote clients and employees (Boughzala et al. 2012). Pearlman *et al.* found examples of use of virtual worlds by companies such as IBM, Starwood and Forterra for product development and the American Cancer Society for annual fund raising events (Pearlman and Gates 2010). Not only has VR entered the workforce, but VR has also continued to become a common household technology as more gaming companies release VR games (Herz and Rahe 2020; van Berlo et al. 2021).

3.3 Virtual Reality in Software Engineering

VR is a realistic interactive environment created by computer graphics. Many research areas, including gaming, training, education, therapy, and more, have implemented VR systems (Lohse et al. 2014; Laver et al. 2017; Bordegoni and Ferrise 2013). Software engineering researchers have recently started to explore the uses of VR for programmers. Elliott *et al.* displays code as labeled piles on the floor and allows programmers to navigate the code piles by grabbing them for closer inspection to understand the functionality better (Elliott et al. 2015b). Takala *et al.* demonstrated the use of virtual environments to provide help with learning how to program software systems (Takala 2014). This environment included "3D User Interface Building Blocks," making it easier for programmers to interact with various peripherals.

Fittkau *et al.* researched "software cities" (Fittkau et al. 2015), which are 3D representations of the packages and classes in a software system. The city metaphor visualizes the object-oriented code as a city. The buildings represent classes, and the districts represent packages. The visual artifacts of the city represent the software metrics. Romano *et al.* found that programmers using the city metaphor did better during code comprehension (Romano et al. 2019a). Programmers using VR were able to complete an average of 5.78 successful code comprehension tasks, whereas those who used a desktop completed only 4.75 code comprehension tasks.

3.4 Human Experience During Comprehension

The human experience during source code comprehension is complex (Al-Saiyd 2017; Khomokhoana and Nel 2019; Tvarozek et al. 2016). Research has found that programmers have similar human experiences while comprehending source code, such as frustration (DeLine et al. 2005; Fritz et al. 2014). Researchers have used functional magnetic resonance imaging (fMRI) machines to understand better how the brain comprehends code (Floyd et al. 2017; Siegmund et al. 2014; Castelhano et al. 2019). Peitek *et al.* found that code comprehension activated the brain regions related to working memory, attention, and language processing (Peitek et al. 2018). Siegmund *et al.* found lower activation of brain areas when using semantic cues for code comprehension (Siegmund et al. 2017).

Although there are similarities in the way programmers comprehend source code, various factors like the complexity of the code and programmer experience affect code comprehension. Cognitive complexity is a complexity measure developed to control the complexity of Fortran programs. Various studies have shown the correlation between cognitive complexity and source code understandability during code comprehension (Campbell 2017; Muñoz Barón 2019). We examine programmers' perceived complexity and understanding of the source code while assessing their perceived productivity during our study. These factors will help us obtain a qualitative understanding of the effectiveness of using virtual environments for code comprehension. LaToza *et al.* compared the human experience of code comprehension in experts and novice programmers (LaToza et al. 2007). They found that experts explained the root cause of the issue, whereas novices could only explain symptoms. They also found that novices spent more time reading more methods and functions whereas experts did not. Romano *et al.* found that using city metaphor visualization in VR positively affected the programmers' feelings and emotions during code comprehension (Romano et al. 2019b).

Concentration and attention are often considered to be largely identical (Petermann 2011). However, various experts also consider concentration to be a part of the attention model (Schweizer 2006). Attention is the neurobiological concept that implies the concentration of mental power of a subject upon an object by careful observation and listening (Ma et al. 2002). Sorqvist *et al.* defines concentration as the shield that acts against distraction under increased task loads (Sörqvist et al. 2016). Various studies have shown decreased performance in reading comprehension with increased load (Kiger 1989; Keating 2008). We use the overall task load to indicate concentration during code comprehension. Our research questions and data analysis examines how task load affects code comprehension in both groups of programmers.

4 Virtual Reality Study Design

In this section, we describe our research objective, research questions, methodology, surveys, data collection, subject application, participants, statistical test, equipment, threats to validity, and replication package.

4.1 Research Questions

Our research objective is to better understand the human experience and how programmers comprehend source code when working in a VR environment. Therefore, we ask the following Research Questions (RQs):

RQ_1 To what degree does code comprehension in VR and desktop settings differ in terms of concentration?
RQ_2 To what degree does code comprehension in VR and desktop settings differ in terms of mental demand?
RQ_3 To what degree does code comprehension in VR and desktop settings differ in terms of frustration?
RQ_4 To what degree does code comprehension in VR and desktop settings differ in terms of physical demand?
RQ_5 To what degree does code comprehension in VR and desktop settings differ in terms of perceived productivity?

RQ_6 To what degree does code comprehension in VR and desktop settings differ in terms of measured productivity?

The rationale behind RQ_1 is that concentration is a weak predictor in reading comprehension. According to Wolfgramm *et al*, it may be a stronger predictor in program comprehension (Wolfgramm et al. 2016). Answering RQ_2 will help understand the difference in the intellectual effort necessary to comprehend code in VR versus on a desktop. The rationale behind RQ_3 is that participants in the VR group must use extra equipment to perform code comprehension. We are interested in observing how this additional requirement affects the participant's frustration level and code comprehension performance. RQ_4 tests the differ- ence in physical demand between the two programming environments. The rationale behind RQ_5 is that comprehension influences productivity (Rastogi et al. 2017). By testing the perceived productivity in different environments, we can explore the environment's influence on comprehension to optimize working conditions for programmers. Finally, by answering RQ_6, we can quantify the productivity differences between the two code comprehension environments and could begin to understand the viability of using VR for code comprehension.

4.2 Methodology

We designed a research methodology based on related work in program comprehension (Good and Brna 2004; Rodeghero et al. 2014a; Abbes et al. 2011) to study individual participant's reading and comprehending source code. We had two groups of participants. One group of participants wore a VR headset for the entire study. The other group of participants sat at a desk and used the desktop computer monitor to read the code and complete the comprehension tasks.

Participants sat in front of a desktop computer setup with a monitor, or they sat down in front of a desk and put on a VR headset. We assigned participants to desktop and VR groups alternating as they came in for the study. For the VR participants, we first introduced them to the VR system. The system consisted of a virtual environment with a Windows desktop mirrored in VR (see Fig. 1). The desktop mirror allowed participants to have full interaction with a computer. For more information on the software used, see Section 4.10. For the VR participants, we introduced them to the VR headset, the keyboard, and the mouse apparatus used for the experiment. We then assisted them with wearing the VR headset and made sure that they were physically comfortable. Participants were allowed to wear the VR headset over their corrective glasses when applicable. During the development of the VR system, we noticed that the virtual hands might not always overlap with the participant's hands. This was an unfortunate artifact introduced by combining HTC Vive and Leap Motion controllers. To remedy this situation, we designed a keyboard alignment software that the participants could use to align the virtual keyboard with their hands in VR. The keyboard alignment software used only the arrow keys to move and rotate the keyboard to make it easier to use. Participants could bring up and close this software using a space bar and "shift" key combination. They could toggle between moving and rotating the keyboard using the "x" key.

Once the participants were in the virtual environment, we introduced them to the keyboard alignment software and gave them as much time as required to make keyboard adjustments. However, this activity did not take more than 5 minutes for any participant during this study. The methodology is the same for both groups of participants throughout the rest of this section.

We provided the participants with 8 different Java projects. The presentation order of these projects was randomized for each participant. The participant read one piece of code at a time.

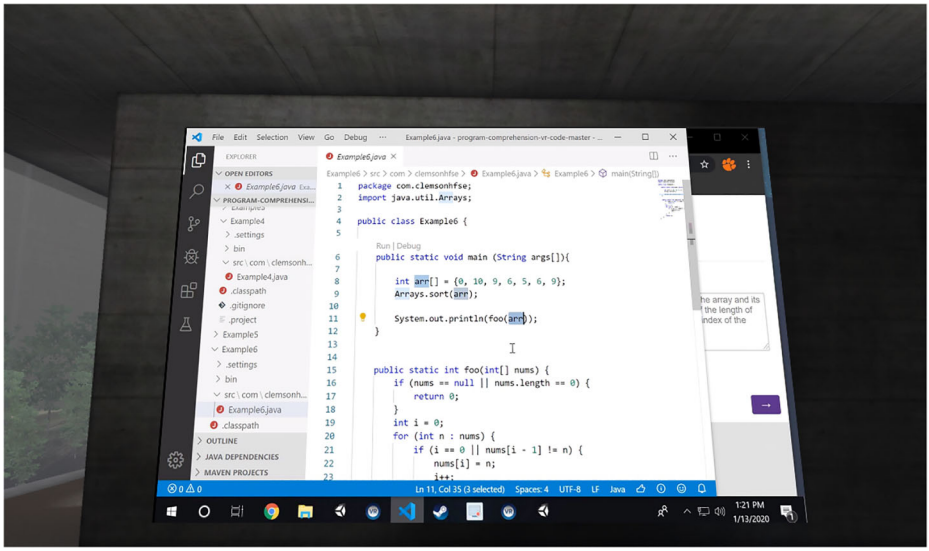

Fig. 1 The virtual reality environment used by the programmer to complete source code comprehension tasks. This figure shows the desktop mirror as the programmer comprehended source code

An image of the source code was presented during this time, accompanied by two text boxes below the image. They then provided the output and an English text summary of the code. The participants had 40 minutes to complete the study. The output was collected using the first text box below the source code. Participants could use the second text box below the source code to record their code summary. There were no formatting requirements for the code summary. During the study, the participants had internet access and could use any internet resource (e.g., Stack Overflow) in VR and desktop setup.

After the study, the participants completed a post-experiment survey, capturing their experiences during the code comprehension study. This survey captured data about how mentally and physically demanding the task was, along with more critical contributions to the overall task load. We also surveyed participants about their experiences using VR.

4.3 Surveys

We conducted three surveys. We collected a (1) demographics survey that asked questions about programming experience, including years of programming and Java experience. At the end of the study, we asked all participants to fill out the (2) NASA TLX survey and a (3) post-experiment survey.

The NASA TLX survey is a multi-dimensional questionnaire designed to determine the task load of a participant performing a task (Hart and Staveland 1988; Hart 2006). Various research communities have adopted it as a reliable measure of task load (Afridi and Mengash 2020; Cecil et al. 2021). Said *et al.* demonstrated that a modified NASA TLX survey accurately reflects the assumed influences of various covariates on perceived workload (Said et al. 2020). This survey measures mental demand, physical demand, temporal demand, performance, effort, and frustration via the question listed in Fig. 2. The participants record their answers on a 21 gradient scale. These gradient scales are converted into standard quantities using the official NASA TLX application after the experiment (TLX @ NASA

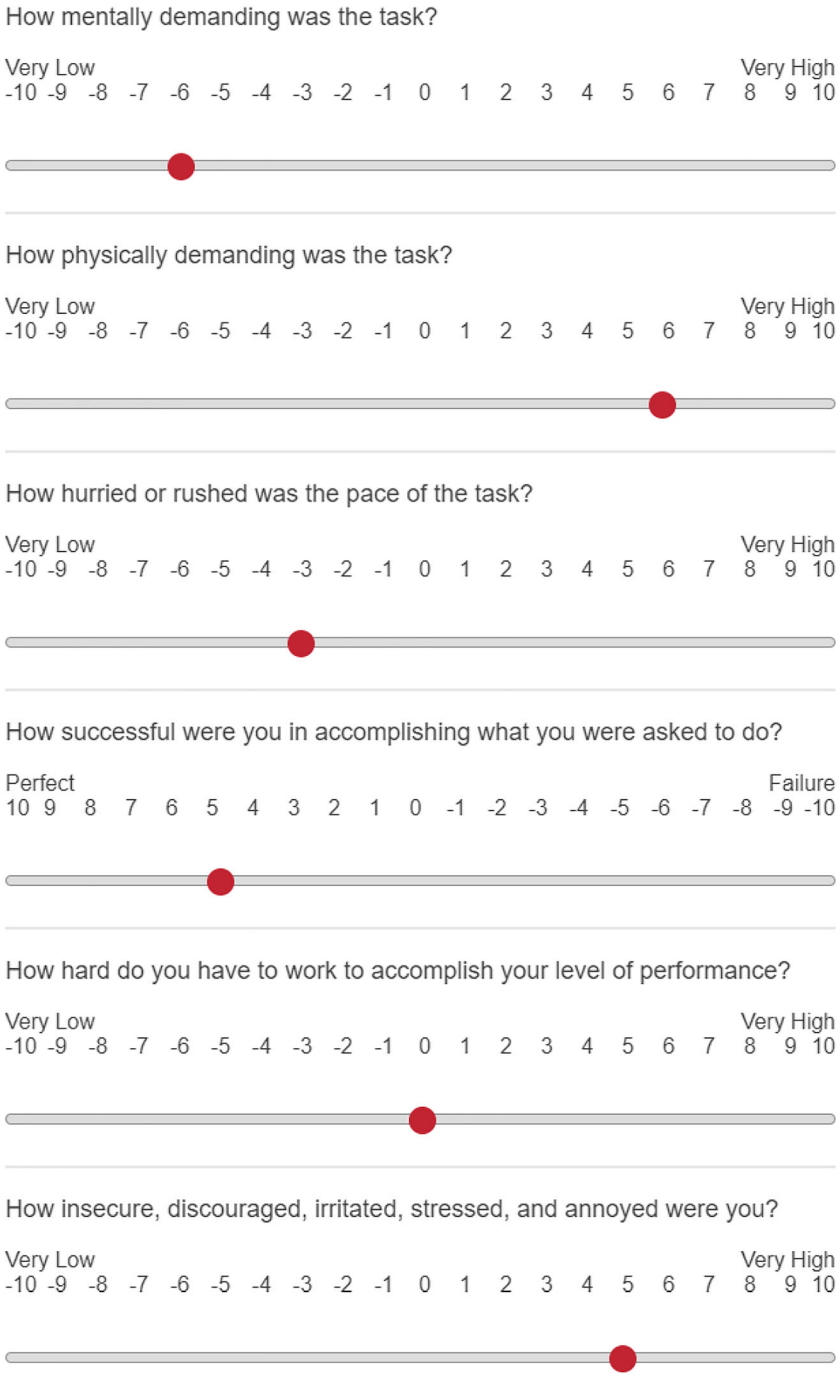

Fig. 2 A screenshot of the NASA TLX survey questions asked during the post-experiment survey. It lists all six questions and the 21 gradient scale

Ames - NASA TLX App. NASA n.d.). The calculations provide an adjusted score for each category and an overall task load index. Figure 3 shows sample measures obtained through NASA TLX survey. All surveys and anonymized data are available in our online appendix (see Section 4.10).

4.4 Data Collection

We collected screen recordings of all participants, and collected both the function outputs and source code summaries. We collected the participants' demographics and self-reported programming experience before the study. After the study, participants completed a questionnaire where they provided a self-reported productivity score. Finally, participants reported their mental demand, physical demand, temporal demand, performance rating, effort, and frustration through the NASA TLX survey during the experiment. We calculated the overall task load using the data collected using the NASA TLX survey. Participants also filled out the simulator sickness questionnaire and the IPQ presence questionnaire (Kennedy et al. 1993; Regenbrecht and Schubert 2002). Since we could not establish a control group that also used VR, we decided not to use the reponses for data analysis. After the study, the second author analyzed the output and code summary provided by the participants. First, the author analyzed the correctness of the output, followed by a review of the code summary in case the output was incorrect. We determined that the source code was successfully comprehended if the output or the summary was correct.

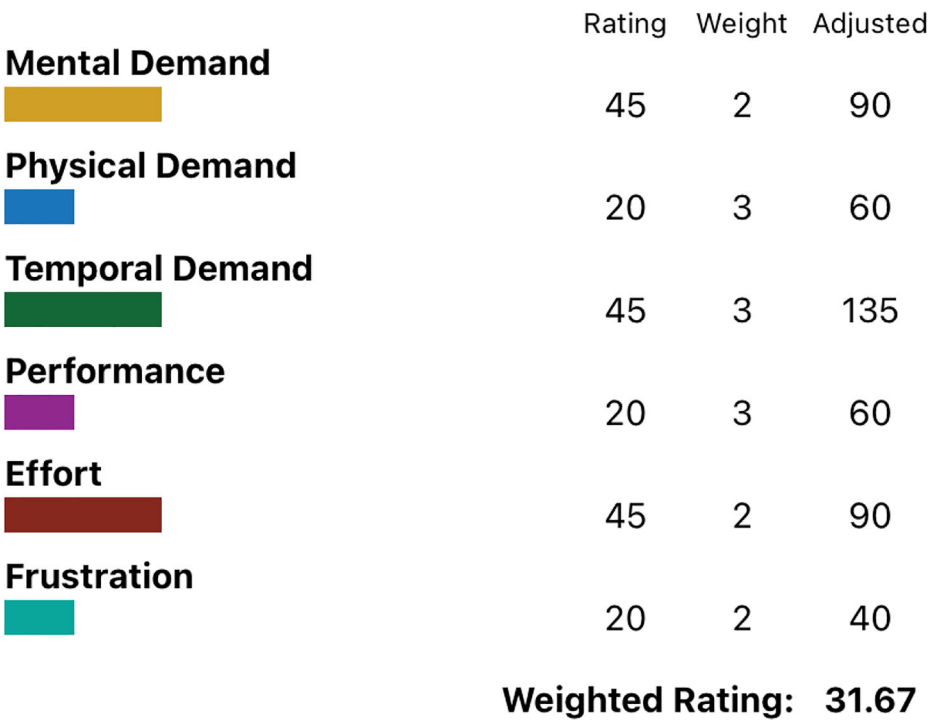

Fig. 3 Sample adjusted scored and overall task load score from the official NASA TLX application

4.5 Subject Application

The subject applications consist of eight Java projects. We selected these projects because they are common methods and functions used in programming paradigms. We selected methods that return the median from an array of integers, maximum and minimum values, compute the factorial of a number, return employee information using encapsulation, remove duplicates in a list, return the length of a word, demonstrate polymorphism, and determine if a string is a palindrome. The first two authors collaborated to collect the subject applications from various coding platforms such as Leetcode and W3Schools (The world's leading online programming learning platform n.d.; W3Schools free online web tutorials n.d.). These projects are unique in their size, domain, and architecture. Similar to previous studies and to accommodate for easy viewing in VR, we limited the code snippets to a size of 22 - 57 lines (Busjahn et al. 2015; Rodeghero et al. 2014b; Peitek et al. 2021). We kept six out of eight code snippets to under 30 lines of code. We provide the subject applications in our online appendix (see Section 4.10).

4.6 Participants

The participants in the program comprehension study were all computer science graduate students recruited through email. There were a total of 26 participants that participated. We recruited all participants from a graduate-level software engineering class at Clemson University. All these participants were fluent in English. In the desktop group, 10 participants reported intermediate Java programming experience. In the virtual reality group, eight participants reported intermediate Java programming experience, and two participants reported expert-level Java programming experience.

The desktop group consisted of four female participants and nine male participants with an average professional software development experience of 2.15 years. Eleven participants reported previous VR experience, with the majority reporting less than 1 hour of VR experience. The average age of this group was 24.76 years. The VR group had one female participant and 12 male participants with an average professional software development experience of 2.07 years. Eight of them reported previous VR experience, with the majority reporting less than five hours of VR experience. The average age of this group was 24.92 years. This data is provided in Table 1. We did not split up the participants based on their skills or experience.

4.7 Statistical Tests

The survey data obtained in this study are ordinal in nature: the NASA-TLX survey measures scores on a twenty-point ordinal scale. We recorded measured comprehension as the number

Table 1 Participants demographic information

Demographic	Desktop			VR		
	Mean	Median	Std Dev	Mean	Median	Std Dev
Professional Experience (yrs)	2.15	2	1.91	2.07	1	1.89
Age (yrs)	24.76	25	2.35	24.92	23	3.70

of source code correctly summarised by each participant during the study. During the post-experiment survey we recorded the perceived productivity, perceived understanding, and perceived complexity of the source code. The perceived productivity question provides ratings measured on a five-point ordinal scale (1 = no productivity; 5 = extremely productive). Participants reported their perceived understanding of the source code on a three-point ordinal scale (1 = definitely not; 3 = definitely yes). We recorded the perceived complexity on a three-point ordinal scale (1 = No; 3 = Yes). We used the Mann-Whitney U test to assess these data for differences among the VR and desktop groups (Hollander et al. 2013). This nonparametric test compares ranks among groups and is appropriate for non-numeric data. Continuity correction was applied to the U test statistic to enable computation of an approximate p-value in the presence of tied ranks. Performing multiple t-tests on subsets of the data obtained through the same observation could result in multiple comparison problem. A test's chances of returning false significance increases as we make multiple inferences from the same observation. The p-values were calculated and then adjusted using the Benjamini-Hochberg method to avoid this problem (Benjamini and Hochberg 1995; Yekutieli and Benjamini 1999). We used Cohen's *d* to calculate the effect size of data collected using NASA-TLX (Cohen 2013). The effect size quantifies the difference between the data collected from the two groups in this study. Finally, we performed a Spearman Correlation test between the combined measured performance and overall task load index of both groups (Binet 1904). This test helps us examine if measure performance is statistically correlated with the overall task load while performing code comprehension.

4.8 Equipment

We designed and developed the VirtualDesk system to conduct this experiment. The VirtualDesk is a hardware and software system that allows users to interact with a computer using a VR headset. We used HTC Vive Pro headsets (Vive Pro 2 - the best VR headset in the metaverse: United States n.d.). The headset had a Leap Motion controller attached to the front panel for hand tracking (Ultraleap: Tracking: Leap Motion controller n.d.). Two HTC Vive trackers were used to create the virtual keyboard and mouse (HTC Vive Tracker: Vive United States n.d.). One tracker was attached to a physical keyboard, and another was attached to the mouse. The virtual keyboard and mouse replicated the physical keyboard and mouse feedback when the user typed on the keyboard or clicked the mouse. The virtual keyboard provided this feedback by highlighting the key pressed and visually moving the key down and then back up. The virtual mouse would momentarily highlight the mouse button that the user pressed. Figure 4a shows a person using the VirtualDesk system with the HTC Vive Pro headset, Leap Motion controller, and the Vive trackers attached to the physical keyboard and mouse. Figure 4b shows the virtual keyboard and mouse where the 'z' button is pressed on the keyboard, shown in blue highlight, and the left mouse button is pressed, shown in white highlight. We used the VirtualDesk system in two previous studies to investigate programmer collaboration and code comprehension in VR (Dominic et al. 2020a; b). For more information on the equipment and setup, see the replication package in Section 4.10.

4.9 Threats to Validity

Similar to any evaluation, our study carries threats to validity. First, participants are susceptible to various symptoms of simulator sickness while using a VR headset. These symptoms include nausea,

(a) (b)

Fig. 4 Programmer using the VR setup. The virtual keyboard and mouse is shown in the second picture. **a**) A programmer performing code comprehension in virtual reality. **b**) Virtual keyboard and mouse used in the VR environment

headache, fatigue, and loss of balance. Factors like fatigue, stress, and error are all part of being human. We minimized this threat by allowing the participant to stay stationary and not require to move in VR. The virtual environment that the participant was in served as a stable horizon that reduces simulator sickness. We allowed the participants to stop any time they experienced simulator sickness symptoms or felt uncomfortable. We did not have anyone stop midway through the experiment. Second, the differences in programming experience, VR experience, and personal biases can affect our study. We cannot rule out the possibility that eliminating these factors would vary our results. However, we minimize this threat by recruiting 26 computer science graduate-level programmers rather than relying on a single graduate student or recruiting undergraduate students. These participants had an average of 2.11 years of professional programming experience. Due to COVID-19 we were unable to recruit additional professional programmers for this study. Another potential threat is the source code that we selected. There is potential that if we choose a different set of source code, programming paradigms, and a different programming language, our results might have been different. To mitigate this threat, we used source code that covered many different programming paradigms, object-oriented best practices, and common functions seen in programming. Our code snippets range in size from 22 lines to 57 lines. Thus we cannot claim that our results are generalizable to source code of arbitrary size. Finally, there is a threat that the technology used for this study might not generalize to other VR technologies. We used the state-of-the-art VR headsets and hand tracking equipment available at the time of the study. We also have made the software and all the data we collected available in Section 4.10. This will allow future researchers to conduct similar studies and build upon the system we have developed.

4.10 Reproducibility & Online Appendix

For reproducibility, we have made all data and software available via an online appendix: https://tinyurl.com/2p9au29a

5 Human Experience Results

In this section, we present our results to each research question, our rationale, and an interpretation of the answers.

5.1 RQ_1: Concentration

Participants in the VR group rated their overall task load significantly higher ($p = 0.03$, effect size = 1.20) than those in the desktop group. The participants in the VR group also found the source code to be more complex ($p = 0.04$, effect size = 1.10) compared to participants in the desktop group. We observed a large and significant difference in the overall task load and perceived complexity of source code between both groups. We present the adjusted p-values from the statistical analysis performed on the data collected during the post-experiment survey and the NASA TLX survey in Fig. 5 and Table 2.

> H_n The difference in concentration between code comprehension in VR and desktop is statistically-significant.

Existing research shows that software developers create more bugs under increased mental and physical stress (Furuyama et al. 1996; Kuo et al. 1998). We found that participants in the VR group experienced more overall task load than those in the desktop group. We argue that this increase in overall task load resulted in the increased perceived complexity of source code among participants in the VR group.

5.2 RQ_2: Mental Demand

We did not find statistically significant evidence that participants in the VR group experienced more mental demand ($p = 0.06$, effect size = 0.72) or temporal demand ($p = 0.24$, effect size = 0.50) than those in the desktop group. These results do not show any differences in the mental demand or temporal demand experienced by both groups. We present the statistical test results on data collected using the NASA TLX survey in Fig. 5 and Table 2.

> H_n The difference in mental load between code comprehension in VR and desktop is not statistically-significant.

While designing the virtual environment, we provided all features that participants would have access to on a desktop computer. This was to make the interactions in VR as natural as possible. The participants in the VR group had access to the virtual keyboard and mouse and the virtual desktop, which mirrored the screen of the computer they were working on. It is well known that the introduction of new technology could distract the users, temporarily reduce productivity and have unintended consequences (Blok et al. 2009; Beland and Murphy 2016). We believe that having minimal deviations while introducing a new technology helped to keep mental demand and temporal demand similar between both groups.

5.3 RQ_3: Frustration

We found no statistically significant evidence that participants doing virtual comprehension were more frustrated ($p = 0.12$, effect size = 0.47) than partici- pants doing desktop comprehension. Our results did not show any statistically significant difference in frustration experienced between the groups. We present the results from our statistical analysis on data collected using NASA TLX survey in Fig. 5 and Table 2.

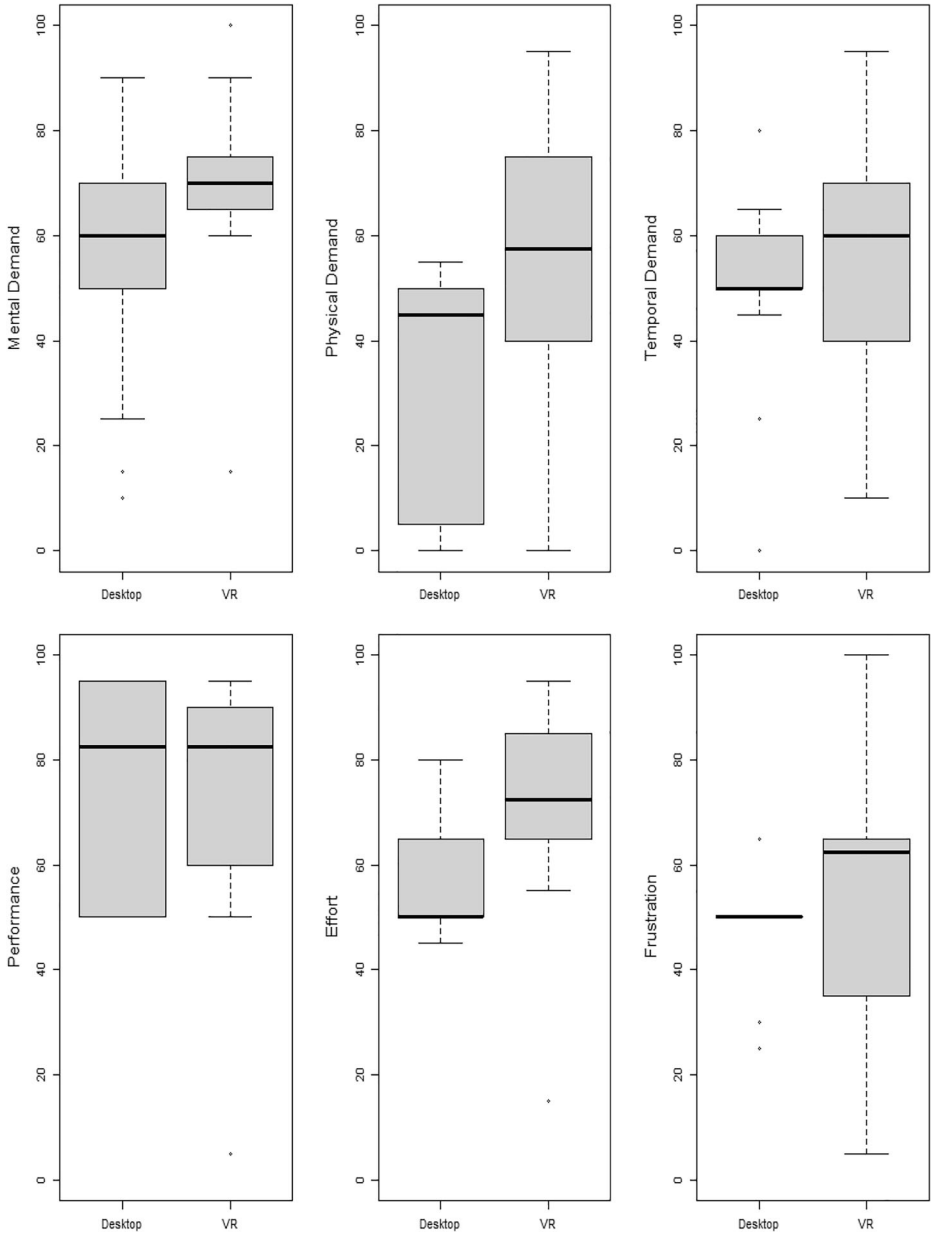

Fig. 5 NASA TLX scores reported by participants while comprehending source code in VR and desktop

H_n The difference in frustration between code comprehension in VR and desktop is not statistically-significant.

The VirtualDesk system required participants to use a VR headset and interact with the computer using the virtual keyboard and mouse. Unfortunately, the Leap Motion controller introduced jitter and drift artifacts while tracking the participant's hands, resulting in sub-

Table 2 Benjamini-Hochberg adjusted p-values and Cohen's *d* on self reported NASA TLX scores, perceived complexity, perceived productivity, and measured productivity

Measure	Desktop mean	VR mean	p-value	Cohen's *d*
Mental Demand	53.84	69.61	0.06	0.72
Physical Demand	30.76	55.38	0.04	0.94
Temporal Demand	48.84	59.61	0.24	0.50
Effort	58.46	71.53	0.04	0.76
Performance Rating	77.69	73.84	0.71	0.17
Frustration	46.92	56.15	0.12	0.47
Overall Task Load	55.66	70.46	0.03	1.20
Perceived Complexity	2.93	2.46	0.04	1.10
Perceived Productivity	4.07	3.30	0.16	0.74
Measured Comprehension	5.38	4.23	0.32	0.52

optimal hand tracking, making it more challenging for the participants to interact with the computer. The results do not show any statistically significant difference in frustration between partici- pants in either setting. It is important to note that the virtual keyboard and mouse were developed exclusively for the VirtualDesk system. However, we cannot rule out the possibility that participants with previous VR experience might have performed better with a shorter learning curve.

5.4 RQ_4: Physical Demand

We found statistically significant evidence that physical demand ($p = 0.04$, effect size = 0.94) and effort ($p = 0.04$, effect size = 0.76) were higher among participants in the VR group than those in the desktop group. Our results show a large and significant difference in the physical demand and a medium but significant difference in effort experienced by both groups. We present the results of the statistical analysis in Fig. 5 and Table 2.

> H_n The difference in physical demand between code comprehension in VR and desktop is statistically-significant.

Participants in the VR group found the headset to be heavy and the keyboard alignment to be subpar. Three participants from the VR group reported that the VR screen was blurry and was hard to read. In the VirtualDesk system, we provided a stable horizon, allowed participants to remain seated, and avoided locomotion within the virtual environment to reduce simulator sickness. However, four of the participants from the VR group reported either eye strain or headache during code comprehension. It is possible that a lighter, newer VR headset with higher display resolution could help alleviate some of the difficulties experienced by the participants.

5.5 RQ_5: Perceived productivity

We found no statistically significant evidence that suggests any difference in perceived productivity ($p = 0.16$, effect size = 0.74) or the NASA TLX per- formance rating ($p = 0.71$, effect size = 0.17) between participants in the VR group and the desktop group. Furthermore, 11 out of 13 participants in the VR group reported *"definitely yes"* when asked,

"Did you understand the source code?" during the post-experiment survey. Interestingly, 11 out of 13 participants in the desktop group also reported *"definitely yes"* when asked the same question. The remaining two participants in both groups reported *"somewhat"* in response to the question. Our results do not show any difference between both groups' perceived productivity or performance rating. We present the results of the statistical analysis performed on the post-experiment survey and the NASA TLX scores in Fig. 5 and Table 2 and the self-reported productivity score for each group in Fig. 6.

> H_n The difference in perceived productivity between code comprehension in VR and desktop is not statistically-significant.

Even though the use of VR for code comprehension is a novel approach, participants did not report significant differences in perceived productivity. Interestingly, there was no difference in the responses to the perceived understanding of source code between both groups. These observations reflect the performance ratings obtained using the NASA TLX survey, where we did not observe any statistically significant differences between the groups.

5.6 RQ_6: Measured Comprehension

We found evidence that participants in the desktop group correctly comprehended a greater percentage of code snippets compared to those in the VR group. Participants in the desktop group comprehended 93 summaries and 70 (75%) of those were correctly comprehended. Participants in the VR group comprehended 84 summaries, and 55 (65%) of those were correctly comprehended. On average, each participant in the desktop group comprehended 5.38 summaries correctly, whereas participants in the VR group only comprehended an

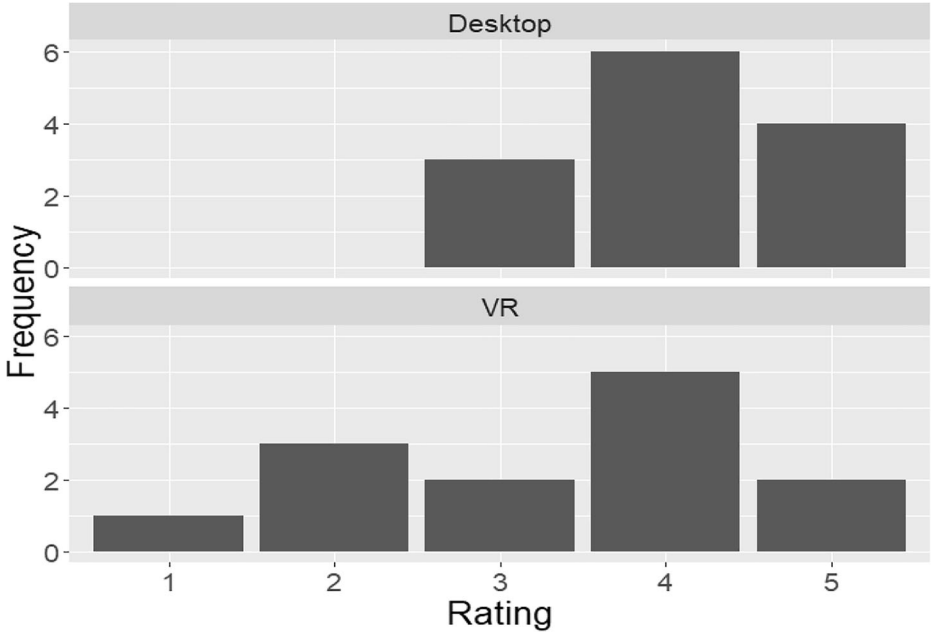

Fig. 6 Distribution of perceived productivity ratings by group (1 = no productivity, 5 = extremely productive)

average of 4.23 summaries correctly. However, we did not find this difference in measured comprehension statistically significant ($p = 0.328$).

> H_n The difference in measured productivity between code comprehension in VR and desktop is not statistically-significant.

Figure 7 shows the correlation between measured comprehension and overall task load. The Spearman Correlation test revealed a slight negative correlation ($r = -0.357$) between both groups' combined measured comprehension and overall task load index. This negative correlation means that the measured comprehension decreases as the overall task load increases. However, we did not find any statistical significance ($p = 0.072$) to this correlation. Since the alpha value is very close to 0.05, we cannot rule out the possibility that having a larger sample size would result a stronger correlation between overall task load

and measured comprehension. The above findings lead us to believe that there is no statistically significant difference in measured comprehension between the desktop and VR groups.

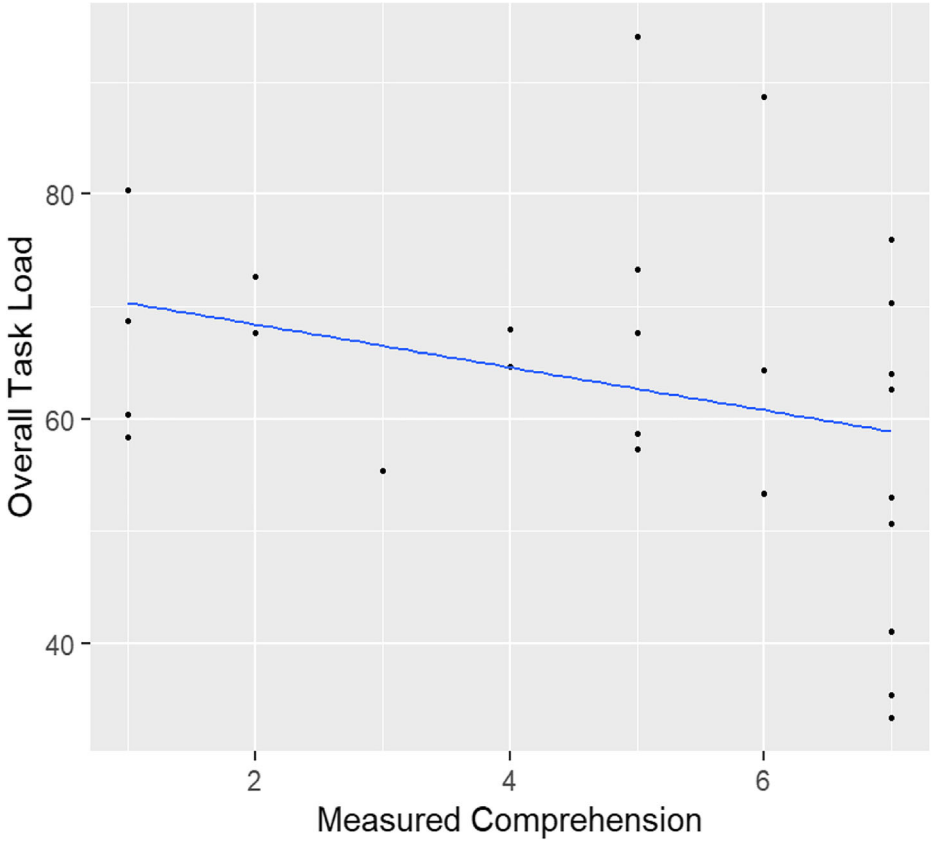

Fig. 7 Weak negative correlation exhibited between Measured Compre- hension and Overall Task Load

5.7 Summary of human experience results

We derive a few key observations from the study results. The participants experienced more challenges during code comprehension in VR over desktop. We base this observation on the statistical significance we found in the higher levels of physical demand, effort, and perceived complexity reported by the participants in the VR group on the NASA TLX survey during code comprehension. Participants in the VR group also reported a higher overall task load, reflected in the overall task load scores. Participants completed fewer code comprehension tasks and even fewer comprehended correctly in VR.

Even though code comprehension in VR placed a higher task load on the participant, we did not find statistically significant evidence suggesting that code comprehension in VR is less beneficial than code comprehension on a desktop computer. We found no statistically significant difference in perceived productivity, performance rating, or perceived understanding of source code between participants in both groups. We did not find any statistical significance for the differences in the actual number of source code snippets correctly comprehended in both groups. Our future work includes improving the hardware and software limitations that resulted in higher task load during code comprehension in VR, which could yield better results supporting the use of virtual environments for code comprehension in future studies.

6 Virtual Reality Study Lessons Learned

This section discusses our experiences and the lessons learned while conducting the code comprehension study in VR. We provided as little assistance as possible to the participants during the study. Unfortunately, once they were in the virtual environment, they were unsure of the study's instructions, often requiring us to repeat the instructions. We recommend letting participants get used to the VR headset and the virtual environment before providing the study instruction. We enlarged the Windows desktop and applications to compensate for the VR headset's limited resolution. As a result, the participants could only see about 20 lines of code at any given time and needed to scroll the Visual Studio Code window to see all source code at once. The magnified view also made it difficult for many participants to use the Visual Studio Code interface in VR. The participants spend much of their time scrolling to view all of the code and not successfully documenting the code summary.

During our study, other lessons learned were the limitations of the VR system and the computer vision technologies used. The Leap Motion hand tracking system was not accurate enough to properly align the participant's fingers with the virtual keyboard for typing in VR. Although each participant had the virtual keyboard aligned to their fingers before the study began, it went out of alignment if the participant made significant adjustments to the headset. This slowed down many participants while documenting code summaries. Many of the participants could not touch type, causing them to make many errors during the summary documentation process, which further slowed them down. We provided a keyboard alignment tool accessed through a space bar and "shift" key press on the keyboard to realign the keyboard as necessary. During the study, a few participants in both groups did not document the output for the given source code. However, all participants documented the source code summary. Hence we decided to consider both the output and the summary while determining the correctness of code comprehension.

7 Discussion

Our paper advances the state-of-the-art in two directions. We contribute to program comprehension literature by adding to the research investigating how programmers comprehend source code in VR. We do this by contributing empirical evidence of programmer behavior during source code summarization tasks in VR. We used the tool presented in Section 4 during these studies.

We recorded the source code summaries of 26 graduate-level software engineering students (13 in VR and 13 using desktop). They evaluated 8 different Java projects, each covering different domains, paradigms, and functionality. We released all of the survey data, summaries, evaluations, and source code via our online appendix (see Section 4.10) to promote independent research. At the same time, we have analyzed the data and found quantitative evidence that source code comprehension in VR is possible. We did not find any statistically significant differences in the measured productivity or the perceived productivity of the programmers between VR and desktop groups. Programmers in VR groups did not report more statistically significant mental demand or frustration than the desktop group. One of the participants from the VR group was excited about using VR for code comprehension and the prospect of using similar systems in the future.

> *"The experiment was quite fun to perform, and could open up new ways to how everyday activities such as programming can be performed."* (P318)

However, programmers experienced various challenges while performing code comprehension in VR. Participants reported higher levels of physical demand and effort in VR. We believe this was due to the difficulty of using the virtual keyboard and reading text in VR. We used an HTC Vive tracker to track the physical keyboard; and a Leap Motion controller for hand tracking. Unfortunately, this combination introduced jitter and drift artifacts, making typing very difficult. The new version of VR headsets are both lighter and comes with inbuilt hand tracking, significantly reducing the jitter and drift otherwise introduced by the two interacting systems, HTC Vive Pro and Leap Motion. Participants in the VR group commented about using the virtual keyboard during code comprehension.

> *"I found it challenging to touch my way around they keyboard when I had the headset on compared to how I normally use the keyboard."* (P304)
> *"The keyboard alignment needs some adjustment, which can take a while and can be frustrating as you are backspacing a lot."* (P300)

The HTC Vive headset has a resolution of 2160×1200 pixels per eye. Even though it is not a small pixel count in most normal circumstances, it has limited text rendering capabilities when considering the 120-degree horizontal field of

view. VR headsets such as the Pimax Vision 8K X offers 3840x2160 resolution per eye, which might offer a more pleasing experience reading text in VR (Pimax Vision 8K X 2022). We tried to mitigate this issue by enlarging the text and the windows interface, which made the text more readable. Even so, in the post-experiment survey, seven participants in the VR group mentioned having trouble reading the text in VR. Below we list the responses from two participants in the VR group.

> *"The hardest part was the loss of vision if that makes sense. It was less clear, had to focus more, maybe there's a setting to make it more crisp."* (P307

"After the first eight minutes or so my eyes started getting really irritated and burned for whatever reason. Made it really uncomfortable and I had to kinda close my eyes for a bit." (P321)

We contribute to measuring the human experience during code comprehension by using the NASA TLX survey. Our study, coupled with the NASA TLX results, produced statistical results for the overall task load, mental demand, physical demand, temporal demand, performance rating, frustration, perceived complexity, perceived productivity, and measured comprehension. Our evaluation found that programmers experience more physical demand and overall task load in VR than desktop code comprehension. Programmers also reported higher statistically significant perceived complexity and effort in VR. However, we did not observe any statistically significant differences in mental demand, temporal demand, performance rating, frustration, perceived productivity, or measured productivity between either group of participants. We provide the results for each of these variables in Table 2.

While conducting the experiments, we provided a very brief pre-training session to the participants using VR. The pre-training session involved finding a comfortable position for the VR headset on their face and introducing them to the keyboard alignment software. The participants aligned the virtual keyboard to their hands and started code comprehension. A participant in the VR group said the below about the lack of pre-training.

"The vision seemed hard while concentrating on the screen. Could have performed better with more exposure to the VR environment and less stress on eyes." (P314)

We cannot rule out the possibility that programmers with more pre-training in VR or more programming experience in virtual environments could perform better than those in our VR group.

8 Recommendations

Our paper offers a framework for future research in remote collaboration. As more programmers move towards remote work, a remote virtual environment can provide an office environment for programmers. Furthermore, VR opens many possibilities for remote collaboration. There are various examples of VR remote collaboration. The VirtualDesk environment already supports multiple users in the same virtual space (Dominic et al. 2020b). We found that programmers collaborated more efficiently using VirtualDesk versus using video conferencing tools. Based on our observations and results, we recommend the following:

Pre-training in VR We recommend that future researchers using VirtualDesk and similar VR systems for software engineering provide ample pre-training of the environment before starting the experiment. Some programmers wanted us to repeat the study instructions once they were in VR. We believe that having a pre-training session will help the programmer become familiar with the virtual environment and understand the instructions more clearly. During our study, the programmers could use the interface as intended once we repeated the instructions with them in VR.

Short Sessions in VR We recommend that programmers use VR headsets for short periods of time. We do not recommend that programmers spend all day working on a virtual headset. A short VR session can be highly productive as programmers can focus on the task without any

external distractions. Unfortunately, using a heavy VR headset and other necessary equipment can be fatiguing for longer sessions. As VR headsets become lighter, untethered, and get integrated hand tracking, programmers will be able to spend more time in VR without fatigue.

VR for Software Development Team Connection We recommend that VR is used for social connection within software development teams. As VR is used in many other domains as a social connection tool, we believe that there may be advantages for software development teams to build social connection and maintain it while teams work in a hybrid or remote model. We believe that further exploration is required in this area.

In addition, we will continue working in this area and hope to see others extend this work. Some of the extensions we have started working on include adding a whiteboard feature in VR. This feature allows programmers to take notes and discuss them with someone else in VR or outside VR. In this paper's work, we found that using VR increased the task load on programmers. We believe that adding a whiteboard will provide a more natural way of taking notes during source code comprehension tasks in VR.

9 Limitations

Limitations of our study include problems with the VR hardware used by programmers. The limited accuracy range of the Leap Motion controller introduced jitter and drift on the hand tracking used in this study. We developed a virtual keyboard calibration system for the user to recalibrate their virtual keyboard and hand calibration to mitigate this issue. The programmers could activate this function with the alt button on the keyboard and then use the arrow keys, which are easier to touch and recognize.

Another limitation of our study is that we used students and not professional programmers. We recruited computer science graduate students to participate in the study. Programmers in both VR and desktop environments reported an average of 2.5 years of professional programming experience. Unfortunately, due to the COVID-19 pandemic, we could not recruit additional professional programmers to participate in the study.

10 Conclusion

In this paper, we presented a study exploring the use of VR for program comprehension by software engineers. We compare the use of VR for program comprehension against desktop program comprehension. We have explored six research questions to understand how VR affects programmers' ability to com- prehend source code. We showed that programmers working in VR face more difficulties comprehending code versus desktop program comprehension. How- ever, we found no statistically significant evidence that suggests a difference in measured productivity or perceived productivity between comprehension in VR and comprehension in a desktop setup.

Acknowledgements We thank all the study participants. We thank Colton Smith for assisting with development of the VirtualDesk system, Jada Houser and Charles Ritter for helping organize and conduct the study.

Declarations

Conflict of interest The authors did not receive support from any organization for the submitted work. The authors have no relevant financial or non-financial interests to disclose.

References

Abbes M, Khomh F, Gueheneuc YG, Antoniol G (2011) An empirical study of the impact of two antipatterns, blob and spaghetti code, on program comprehension. In: 2011 15Th european conference on software maintenance and reengineering. IEEE, pp 181–190

Afridi AH, Mengash HA (2020) NASA-TLX–based workload assessment for academic resource recommender system. Personal and Ubiquitous Computing, pp 1–19

Akbulut A, Catal C, Yıldız B (2018) On the effectiveness of virtual real- ity in the education of software engineering. Comput Appl Eng Educ 26(4):918–927

Aleotti J, Caselli S, Reggiani M (2004) Leveraging on a virtual environment for robot programming by demonstration. Robot Auton Syst 47(2–3):153–161

Al-Saiyd NA (2017) Source code comprehension analysis in software maintenance. In: 2017 2nd International Conference on Computer and Communication Systems (ICCCS). IEEE, pp 1–5

Arisholm E, Gallis H, Dyba T, Sjoberg DI (2007) Evaluating pair pro- gramming with respect to system complexity and programmer expertise. IEEE Trans Softw Eng 33(2):65–86

Bacher I, Mac Namee B, Kelleher JD (2017) Scoped: visualising the scope chain within source code. In: EuroVis (Short Papers), pp 115–119

Baheti P, Gehringer E, Stotts D (2002) Exploring the efficacy of distributed pair programming. In: Conference on Extreme Programming and Agile Methods. Springer, Berlin, Heidelberg, pp 208–220

Beland L-P, Murphy R (2016) Ill communication: technology, distraction & student performance. Labour Econ 41:61–76

Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a prac- tical and powerful approach to multiple testing. J R Stat Soc Ser B Methodol 57(1):289–300

Bierbaum A, Just C, Hartling P, Meinert K, Baker A, Cruz-Neira C (2001) VR Juggler: A virtual platform for virtual reality application development. In: Proceedings IEEE Virtual Reality 2001. IEEE, pp 89–96

Binet A (1904) Spearman the proof and measurement of association between two things; general intelligence objectively determined and measured. Lánnèe Psychologique 11(1):623–624

Blok M, De Korte E, Groenesteijn L, Formanoy M, Vink P (2009) The effects of a task facilitating working environment on office space use, communication, concentration, collaboration, privacy and distraction. In: Proceedings of the 17th World Congress on Ergonomics (IEA 2009), 9–14 August 2009. International Ergonomics Association, Beijing

Bordegoni M, Ferrise F (2013) Designing interaction with consumer prod- ucts in a multisensory virtual reality environment. Virtual and Physical Prototyping 8(1):51–64. https://doi.org/10.1080/17452759.2012.762612

Boughzala I, de Vreede G-J, Limayem M (2012) Team collaboration in virtual worlds: editorial to the special issue. J Assoc Inf Syst 13(10):6

Bozgeyikli E, Raij A, Katkoori S, Dubey R (2016) Point & teleport locomotion technique for virtual reality. In: Proceedings of the 2016 annual symposium on computer-human interaction in play, pp 205–216

Brynjolfsson E, Horton JJ, Ozimek A, Rock D, Sharma G, TuYe HY (2020) COVID-19 and remote work: An early look at US data (No. w27344). National Bureau of Economic Research

Busjahn T, Bednarik R, Begel A, Crosby M, Paterson JH, Schulte C, Sharif B, Tamm S (2015) Eye movements in code reading: Relaxing the linear order. In: 2015 IEEE 23rd International Conference on Program Comprehension. IEEE, pp 255–265

Campbell GA (2017) Cognitive complexity-a new way of measuring under- standability. Technical Report. SonarSource SA, Switzerland

Castelhano J, Duarte IC, Ferreira C, Duraes J, Madeira H, Castelo-Branco M (2019) The role of the insula in intuitive expert bug detec- tion in computer code: an fmri study. Brain imaging and behavior 13(3):623–637

Cecil J, Kauffman S, Gupta A, McKinney V, Pirela-Cruz MM (2021) Design of a human centered computing (HCC) based virtual reality simulator to train first responders involved in the Covid-19 pandemic. In: 2021 IEEE International Systems Conference (SysCon). IEEE, pp 1–7

Cohen J (2013) Statistical power analysis for the behavioral sciences. Routledge, Milton Park

DeLine R, Czerwinski M, Robertson G (2005) Easing program comprehension by sharing navigation data. In: 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05). IEEE, pp 241–248

Dominic J, Tubre B, Houser J, Ritter C, Kunkel D, Rodeghero P (2020a) Program comprehension in virtual reality. In: Proceedings of the 28th International Conference on Program Comprehension, pp 391–395
Dominic J, Tubre B, Ritter C, Houser J, Smith C, Rodeghero P (2020b) Remote pair programming in virtual reality. In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, pp 406–417
Elliott A, Peiris B, Parnin C (2015a) Virtual reality in software engineer- ing: Affordances, applications, and challenges. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol 2, pp 547–550. https://doi.org/10.1109/ICSE.2015.191
Elliott A, Peiris B, Parnin C (2015b) Virtual reality in software engineering: Affordances, applications, and challenges. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. IEEE, vol 2, pp 547–550
Fittkau F, Krause A, Hasselbring W (2015) Exploring software cities in virtual reality. In: 2015 ieee 3rd working conference on software visualization (vissoft). IEEE, pp 130–134
Floyd B, Santander T, Weimer W (2017) Decoding the representation of code in the brain: An fMRI study of code review and expertise. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). IEEE, pp 175–186
Freeman D, Reeve S, Robinson A, Ehlers A, Clark D, Spanlang B, Slater M (2017) Virtual reality in the assessment, understanding, and treatment of mental health disorders. Psychol Med 47(14):2393–2400
Fritz T, Begel A, Müller SC, Yigit-Elliott S, Züger M (2014) Using psycho-physiological measures to assess task difficulty in software development. In: Proceedings of the 36th international conference on software engineering, pp 402–413
Furuyama T, Arai Y, Iio K (1996) Analysis of fault generation caused by stress during software development. In: Achieving Quality in Software. Springer, Boston, pp 14–28
Future of remote work? here's the reality of working in the metaverse (2022). https://www.euronews.com/next/2022/02/16/what-s-remote-work-like-in-the-metaverse-these-companies-are-building-the-tools-t
Good, J., Brna, P. Program comprehension and authentic measure- ment:: a scheme for analysing descriptions of programs. International Journal of Human-Computer Studies 61(2), 169–185 (2004)
Graham K, Fai S, Dhanda A, Smith L, Tousant K, Wang E, Weigert A (2018) The VR kiosk. In: Digital Cultural Heritage. Springer, Cham, pp 324–336
Grubert J, Ofek E, Pahud M, Kristensson PO, Steinicke F, San-dor C (2018) The office of the future: Virtual, portable, and global. IEEE Comput Graph Appl 38(6):125–133
Guerin K, Hager GD (2017) Robot control, training and collaboration in an immersive virtual reality environment. Google Patents US Patent 9:643–314
Gulec U, Yilmaz M, Isler V, O'Connor RV, Clarke P (2018) Adopting virtual reality as a medium for software development process education. In: Proceedings of the 2018 International Conference on Software and System Process, pp 71–75
Häfner P, Häfner V, Ovtcharova J (2013) Teaching methodology for vir- Tual reality practical course in engineering education. Procedia Computer Science 25:251–260
Hart SG (2006) NASA-task load index (NASA-TLX); 20 years later. In: Proceedings of the human factors and ergonomics society annual meeting. Sage CA: Los Angeles, CA: Sage publications, vol 50, no. 9, pp 904–908
Hart SG, Staveland LE (1988) Development of nasa-tlx (task load index): results of empirical and theoretical research. In: Advances in Psychology 52:139–183
Hayes A, Johnson K (2019) Cultural embodiment in virtual reality education and training: A reflection on representation of diversity. In: Foundations and Trends in Smart Learning, pp 93–96
Hayes B, Chang Y, Riley G (2018) Controlled unfair adaptive 360 vr video delivery over an MPTCP/QUIC architecture. In: 2018 IEEE International Conference on Communications (ICC). IEEE, pp 1–6
Herz M, Rahe V (2020) Virtual reality becoming part of our lives—assessing consumers' perceived applicability of virtual reality supported tasks and a critical reflection on the development. In: Augmented Reality and Virtual Reality. Springer, Cham, pp 113–122
Hodges LF, Kooper R, Meyer TC, Rothbaum BO, Opdyke D, Graaff JJD, Williford JS, North MM (1995) Virtual environments for treating the fear of heights. IEEE Comput 28(7):27–34
Hoffmann M, Meisen T, Jeschke S (2016) Shifting virtual reality education to the next level–experiencing remote laboratories through mixed reality. In: Engineering Education 4.0:235–249
Hollander M, Wolfe DA, Chicken E (2013 Nonparametric statistical methods. John Wiley & Sons, Hoboken
Hoppe AH, Westerkamp K, Maier S, Camp FVD, Stiefelhagen R (2018) Multi-user collaboration on complex data in virtual and augmented reality. In: International Conference on Human-Computer Interaction. Springer, Cham, pp 258–265
Hossain E, Babar MA, Verner J (2009) How can agile practices minimize global software development co-ordination risks?. In: European Conference on Software Process Improvement. Springer, Berlin, Heidelberg, pp 81–92
HTC Vive Tracker: Vive United States (n.d.) https://www.vive.com/us/ vive-tracker/
Jeon C (2015) The virtual flier: The link trainer, flight simulation, and pilot identity. Technol Cult 56:28–53

Johansson M, Roupè M, Viklund Tallgren M (2014) From BIM to VR-Integrating immersive visualizations in the current design process. In: Fusion-Proceedings of the 32nd eCAADe Conference-Volume 2 (eCAADe 2014), pp 261–269

Johnson PM, Kou H, Agustin J, Chan C, Moore C, Miglani J, Zhen S, Doane WE (2003) Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In: 25th International Conference on Software Engineering, 2003. Proceedings. IEEE, pp 641–646

Keating GD (2008) Task effectiveness and word learning in a second language: the involvement load hypothesis on trial. Lang Teach Res 12(3):365–386

Kennedy RS, Lane NE, Berbaum KS, Lilienthal MG (1993) Simulator sickness questionnaire: an enhanced method for quantifying simulator sickness. Int J Aviat Psychol 3(3):203–220

Khomokhoana PJ, Nel L (2019) Decoding source code comprehension: bottlenecks experienced by senior computer science students. In: Annual Conference of the Southern African Computer Lecturers' Association. Springer, Cham, pp 17–32

Kiger DM (1989) Effects of music information load on a reading comprehen- Sion task. Percept Mot Skills 69(2):531–534

Kircher M, Jain P, Corsaro A, Levine D (2001) Distributed extreme programming. Extreme Programming and Flexible Processes in Software Engineering, Italy, pp 66–71

Kreutzberg A (2015) Conveying architectural form and space with virtual reality. https://doi.org/10.52842/conf.ecaade.2015.1.117

Kuo W, Chien WTK, Kim T (1998) Reliability, yield, and stress burn-in: a unified approach for microelectronics systems manufacturing & software development. Springer Science & Business Media, Berlin

Kurumi MKY, Morikawa S (2016) Active and passive haptic training approaches in vr laparoscopic surgery training. Med Meets Virtual Reality 22: NextMed/MMVR22 220:215

LaToza TD, Garlan D, Herbsleb JD, Myers BA (2007) Program comprehension as fact finding. In: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, pp 361–370

Laver KE, Lange B, George S, Deutsch JE, Saposnik G, Crotty M (2017) Virtual reality for stroke rehabilitation. Cochrane Database Syst Rev (11). https://doi.org/10.1002/14651858.CD008349.pub4

Lohse KR, Hilderman CGE, Cheung KL, Tatla S, Van der Loos HFM (2014) Virtual reality therapy for adults post-stroke: a systematic review and meta-analysis exploring virtual environments and commer- cial games in therapy. PLoS One 9(3):1–13. https://doi.org/10.1371/journal.pone.0093318

Ma YF, Lu L, Zhang HJ, Li M (2002) A user attention model for video summarization. In: Proceedings of the tenth ACM international conference on Multimedia, pp 533–542

Martin-Gutierrez J, Martìn-Gutìerrez J, Mora CE, Añorbe-Diaz B, González-Marrero A (2017) Learning strategies in engineering education using virtual and augmented reality technologies. Eurasia J Math Sci Technol Educ 13(2):297–300

Matsas E, Vosniakos G-C (2017) Design of a virtual reality training sys- tem for human–robot collaboration in manufacturing tasks. International Journal on Interactive Design and Manufacturing (IJIDeM) 11(2):139–153

Merino L, Bergel A, Nierstrasz O (2018) Overcoming issues of 3D software visualization through immersive augmented reality. In: 2018 IEEE Working Conference on Software Visualization (VISSOFT). IEEE, pp 54–64

Miller C, Rodeghero P, Storey MA, Ford D, Zimmermann T (2021) "How was your weekend?" software development teams working from home during covid-19. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, pp 624–636

Muñoz Barón M (2019) A validation of cognitive complexity as a measure of source code understandability. Bachelor's thesis, University of Stuttgart

Mystakidis S (2022) Metaverse. Encyclopedia 2(1):486–497

Nazligul MD, Yilmaz M, Gulec U, Gozcu MA, O'Connor RV, Clarke PM (2017) Overcoming public speaking anxiety of software engineers using virtual reality exposure therapy. In European Conference on Software Process Improvement. Springer, Cham, pp 191–202

Nguyen-Duc A, Cruzes DS, Conradi R (2015) The impact of global dispersion on coordination, team performance and software quality–a systematic literature review. Inf Softw Technol 57:277–294

O Connor M, Conboy K, Dennehy D (2022) COVID-19 affected remote workers: a temporal analysis of information system development during the pandemic. J Decis Syst 31(3):207–233

Oberhauser R, Lecon C (2017) Gamified Virtual Reality for Program Code Structure Comprehension. International Journal of Virtual Reality 17(2):79–88

Panas T, Berrigan R, Grundy J (2003) A 3d metaphor for software production visualization. In: Proceedings on Seventh International Conference on Information Visualization, 2003. IV 2003. IEEE, pp 314–319

Parsons S, Cobb S (2011) State-of-the-art of virtual reality technologies for children on the autism spectrum. Eur J Spec Needs Educ 26(3):355–366

Pearlman DM, Gates NA (2010) Hosting business meetings and special events in virtual worlds: a fad or the future? In: journal of Convention & Event Tourism. Taylor & Francis 11:247–265
Peitek N, Apel S, Parnin C, Brechmann A, Siegmund J (2021) Program comprehension and code complexity metrics: An fmri study. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, pp 524–536
Peitek N, Siegmund J, Apel S, Kästner C, Parnin C, Bethmann A, Leich T, Saake G, Brechmann A (2018) A look into programmers' heads. IEEE Transact Softw Eng 46(4):442–462
Pellas N, Dengel A, Christopoulos A (2020) A scoping review of immer- sive virtual reality in stem education. IEEE Trans Learn Technol 13(4):748–761
Petermann F (2011) Frankfurter aufmerksamkeits-inventar 2 (fair-2). Z Psychiatr Psychol Psychother 59(4):325–326
Pimax Vision 8K X (2022). https://pimax.com/product/vision-8k-x/
Pourchera G, Micheletb D, Recanzonec T, Stitic S, Jolivetc E, Barréb J (2018) Interest of virtual reality (vr) simulation for surgical learning: Vr single port sleeve gastrectomy. In: Obesity surgery, vol. 28, pp. 531–531. Springer 233 Spring St, New York, NY 10013 USA
Racz A, Zilizi G (2018) VR aided architecture and interior design. In: 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE). IEEE, pp 11–16
Ralph P, Baltes S, Adisaputri G et al (2020) Pandemic programming. Empir Software Eng 25:4927–4961. https://doi.org/10.1007/s10664-020-09875-y
Rastogi A, Thummalapenta S, Zimmermann T, Nagappan N, Czer-wonka J (2017) Ramp-up journey of new hires: Do strategic practices of software companies influence productivity?. In: Proceedings of the 10th Innovations in Software Engineering Conference, pp 107–111
Regenbrecht H, Schubert T (2002) Real and illusory interactions enhance presence in virtual environments. Presence: Teleoperators & Virtual Environments 11(4):425–434
Rodeghero, P., McMillan, C., McBurney, P.W., Bosch, N., D'Mello, S. (2014a) Improving automated source code summarization via an eye-tracking study of programmers. In: Proceedings of the 36th International Confer- ence on Software Engineering. ICSE 2014, pp. 390–401. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2568225.2568247
Rodeghero P, McMillan C, McBurney PW, Bosch N, D'Mello S (2014) Improving automated source code summarization via an eye-tracking study of programmers. In: Proceedings of the 36th international conference on Software engineering, pp 390–401
Romano S, Capece N, Erra U, Scanniello G, Lanza M (2019a) On the use of virtual reality in software visualization: the case of the city metaphor. Inf Softw Technol 114:92–106
Romano S, Capece N, Erra U, Scanniello G, Lanza M (2019b) The city metaphor in software visualization: feelings, emotions, and thinking. Multimed Tools Appl 78(23):33113–33149
Ruvimova A, Kim J, Fritz T, Hancock M, Shepherd DC (2020) "Transport Me Away": Fostering flow in open offices through virtual reality. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp 1–14
Said S, Gozdzik M, Roche TR, Braun J, Rössler J, Kaserer A, Spahn DR, Nöthiger CB, Tscholl DW et al (2020) Validation of the raw national aeronautics and space administration task load index (nasa-tlx) questionnaire to assess perceived workload in patient monitoring tasks: pooled analysis study using mixed models. J Med Int Res 22(9):19472
Sampaio AZ (2018) Enhancing BIM methodology with VR technology. State of the Art Virtual Reality and Augmented Reality Knowhow, pp 59–79
Sattar MU, Palaniappan S, Lokman A, Shah N, Khalid U, Hasan R (2020) Motivating medical students using virtual reality based education. International Journal of Emerging Technologies in Learning (iJET) 15(02):160–174
Schenk J (2018) Industrially usable distributed pair programming. Dissertation, Freie Universität Berlin
Schweizer K (Ed.) (2006) Leistung und Leistungsdiagnostik. Springer Berlin Heidelberg
Sharma VS, Mehra R, Kaulgud V, Podder S (2018) An immersive future for software engineering: avenues and approaches. In: Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results, pp 105–108
Siegmund J, Kästner C, Apel S, Parnin C, Bethmann A, Leich T, Saake G, Brechmann A (2014) Understanding understanding source code with functional magnetic resonance imaging. In: Proceedings of the 36th international conference on software engineering, pp 378–389

Siegmund J, Peitek N, Parnin C, Apel S, Hofmeister J, Kästner C, Begel A, Bethmann A, Brechmann A (2017) Measuring neural efficiency of program comprehension. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp 140–150
Sörqvist P, Dahlström Ö, Karlsson T, Rönnberg J (2016) Concentration: the neural underpinnings of how cognitive load shields against distraction. Front Hum Neurosci 10:221
Steinmacher I, Conte TU, Treude C, GerosaMA (2016) Overcoming open source project entry barriers with a portal for newcomers. In: Proceedings of the 38th International Conference on Software Engineerin, pp 273–284
Takala TM (2014) RUIS: A toolkit for developing virtual reality applications with spatial interaction. In: Proceedings of the 2nd ACM symposium on Spatial user interaction, pp 94-103
Taxén G, Naeve A (2002) A system for exploring open issues in vr-based education. Comput Graph 26(4):593–598
The world's leading online programming learning platform (n.d.) https://leetcode.com/
TLX @ NASA Ames - NASA TLX App. NASA (n.d.) https://humansystems. arc.nasa.gov/groups/tlx/tlxapp.php
Tvarozek J, Konopka M, Navrat P, Bielikova M (2016) Studying various source code comprehension strategies in programming education. Eye Movements in Programming: Models to Data 23:25–26
Ullrich S, Kuhlen T (2012) Haptic palpation for medical simulation in vir- Tual environments. IEEE Trans Vis Comput Graph 18(4):617–625
Ultraleap: Tracking: Leap Motion controller (n.d.) https://www.ultraleap. com/product/leap-motion-controller/
Upwork Study Finds 22% of American Workforce Will Be Remote by 2025: Upwork (2020) https://www.upwork.com/press/releases
van Berlo ZM, van Reijmersdal EA, Smit EG, van der Laan LN (2021) Brands in virtual reality games: affective processes within computer- mediated consumer experiences. J Bus Res 122:458–465
Vincur J, Konopka M, Tvarozek J, Hoang M, Navrat P (2017) Cubely: virtual reality block-based programming environment. In: Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, pp 1–2
Vive Pro 2 - the best VR headset in the metaverse: United States (n.d.) https://www.vive.com/us/product/vive-pro2/overview/
W3Schools free online web tutorials (n.d.) https://www.w3schools.com/
Wang P, Wu P, Wang J, Chi H-L, Wang X (2018) A critical review of the use of virtual reality in construction engineering education and training. Int J Environ Res Public Health 15(6):1204
Wiese ES, Rafferty AN, Fox A (2019) Linking code readability, structure, and comprehension among novices: it's complicated. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET). IEEE, pp 84–94
Williams L, Kessler RR, Cunningham W, Jeffries R (2000) Strengthening the case for pair programming. IEEE Softw 17(4):19–25
Wolfgramm C, Suter N, Göksel E (2016) Examining the role of concentra- tion, vocabulary and self-concept in listening and reading comprehen- sion. Int J Listen 30(1–2):25–46
Workrooms: VR for business meetings (n.d.) https://www.oculus.com/workrooms/
Yekutieli D, Benjamini Y (1999) Resampling-based false discovery rate con- trolling multiple test procedures for correlated test statistics. Journal of Statistical Planning and Inference 82(1–2):171–196
Zirkelbach C, Krause A, Hasselbring W (2019) Hands-on: experiencing software architecture in virtual reality

James Dominic is a Ph.D. student at Clemson University's School of Computing. He completed his Bachelor's of Computer Science and Engineering from Mahatma Gandhi University, India, and his Master's in Computer Science from Clemson University. Before joining Clemson, he worked as an Oracle Apps Technical Consultant. In the Fall of 2018, he joined the HFSE team and explored using Virtual Reality for software engineering. His current research focuses on using artificial intelligence to create bots for software engineers.

Brock Tubre has a Bachelor's of Computer Science from Louisiana Tech University and a Master's in Computer Science from Louisiana State University Shreveport. Brock is an experienced software engineer and architect working on large scale enterprise software solutions for the US government. Building software and teaching the world about cloud computing are his professional passions.

Deborah Kunkel is an assistant professor in the School of Mathematical and Statistical Sciences at Clemson University. She completed a PhD in Statistics in 2018 from The Ohio State University. She is an applied statistician with expertise in linear and mixed effects models, hierarchical models, clustering methods, latent variable models, and generalized additive models. She also does methodological research in the areas of Bayesian methodology and mixture models.

Paige Rodeghero is an Assistant Professor in the School of Computing at Clemson University, where she directs the Human Factors Software Engineering research team. She obtained her Ph.D. in Computer Science and Engineering from the University of Notre Dame. Her main research interest is software engineering, focused on productivity, remote work, onboarding, source code comprehension, computer science education, and software engineering for autism. Previously, she was a visiting researcher at Microsoft Research. She has won multiple best paper awards, including two ACM SIGSOFT Distinguished Paper Awards, and the NSF funds her research. Previously to her research career, she worked in the industry as a lead software engineer for a startup company and as a software engineer at multiple medium-sized companies.