Collaborative software design and modeling in virtual reality

Martin Stancek a, Ivan Polasek b, a,*, Tibor Zalabai a, Juraj Vincur b, Rodi Jolak c, Michel Chaudron d

a Gratex International a.s., Galvanibo 17/C, Bratislava, 821 04, Slovakia
b Comenius University, Faculty of Mathematics, Physics and Informatics, Mlynská dolina F1, Bratislava, 842 48, Slovakia
c RISE Research Institutes of Sweden, Box 857, Bords, 501 15, Sweden
d Eindhoven University of Technology, O.L. Vrouwestraat 1, Eindhoven, 5612 AW, Netherlands

ARTICLE INFO

Keywords:
Software modeling
Software development
Immersion
Collaboration

ABSTRACT

Context: Software engineering is becoming more and more distributed. Developers and other stakeholders are often located in different locations, departments, and countries and operating within different time zones. Most online software design and modeling tools are not adequate for distributed collaboration since they do not support awareness and lack features for effective communication.

Objective: The aim of our research is to support distributed software design activities in Virtual Reality (VR).

Method: Using design science research methodology, we design and evaluate a tool for collaborative design in VR. We evaluate the collaboration efficiency and recall of design information when using the VR software design environment compared to a non-VR software design environment. Moreover, we collect the perceptions and preferences of users to explore the opportunities and challenges that were incurred by using the VR software design environment.

Results: We find that there is no significant difference in the efficiency and recall of design information when using the VR compared to the non-VR environment. Furthermore, we find that developers are more satisfied with collaboration in VR.

Conclusion: The results of our research and similar studies show that working in VR is not yet faster or more efficient than working on standard desktops. It is very important to improve the interface in VR (gestures with haptics, keyboard and voice input), as confirmed by the difference in results between the first and second evaluation.

1. Introduction

The engineering of software systems is a socio-technical process where software developers and other stakeholders having different expertise and experiences are involved [2]. One of the fundamental phases of software engineering is software design where important aspects of the problem and solution domain are explored, discussed, and often described in software models [3].

Whiteboards are widely used by developers to design and create software system models [4]. However, software engineering, including software design and modeling, is becoming more and more distributed [5]. Indeed, teams are often located in various departments, divisions, countries, and operate within different time zones [6].

Most online design and modeling tools do not provide immersive collaboration environments for collaborative developers [5]. Indeed, these tools lack features addressing collaboration awareness and effective communication, which could influence the effectiveness of the design process and thus the overall quality of the software.

Nowadays, Virtual Reality (VR) is an attractive and accessible technology that is used to build VR-based environments. These VR environments allow team members to work together in a shared virtual space, regardless of their physical location, which can increase flexibility [7]. In addition, VR environments provide an immersive telepresence experience and support awareness in remote collaboration.

At the beginning of this research there was a debate with the developers at the software company Gratex International, who had had to collaborate with their colleagues in Seoul and Australia on insurance software and data models for contracts, etc. They asked us if the decisions over model changes and the design of new modules would...
not be more intense and natural if we were also able to see what the others are looking at in VR and what they are doing (differentiation by a unique color for each person). Many colleagues felt a disconnect from the other team members in Seoul. We needed to somehow overcome the long distance between Europe, Australia and Seoul and get closer in VR. It was also about saving costs, so that other colleagues would not have to travel (flights and staying in Seoul and Australia are not cheap) and waste time on planes and in airports if they had another project here in Europe.

Later came the change in work due to the pandemic, as we needed to collaborate without endangering our health. Subsequently, after the initial waves of the pandemic and the restrictions, there was a greater preference for working from home among employees, and utilizing VR for designing or presenting models became an intriguing alternative.

We were also interested in VR as a means of changing our working style, in addition to the constant sitting behind a standard computer. We were curious about whether immersion in VR for collaborative modeling could improve the results or, at the very least, enhance the designers’ experience.

Several arguments for the use of VR have emerged:

• Overcoming the geographical distance of distributed teams and reducing travel and accommodation costs.
• Providing solutions for online work within a home office, especially during situations like illness or pandemics.
• Immersion in collaborative modeling by surrounding ourselves with models of the system.
• Changing the work style and breaking the monotony of sitting at the computer.

Brettschuh et al. in [8] stated that VR tools could be one of the promising implementations of Industry 4.0. They also reported the main domains of VR closely related to our area, which include Product design, Process design and visualization, Collaboration and Co-creation, Product Presentation, and Prototyping. The majority of studies they reviewed utilized the effect of immersion for their own purposes, leveraging VR to gain various advantages.

Berg and Vance argue in [9] that VR supports a sense of team engagement, which leads to better discussions and increased participation of team members in the decision-making process. They demonstrated in their survey [10] that VR has been accepted and actively used in numerous industries to support decision-making and enable innovation. VR can involve end-users in the design process better by providing different alternatives.

Due to the impact of the pandemic situation, VR has been considered a potential answer to online teaching for software engineering and learning enhancement [11]. In [11], they found that VR improves the experience in software engineering presentations, and the students of SE feel more presence and have better interactions during the presentations in VR. Borst et al. in [12] reported that VR could increase motivation and engagement. R. Oberhauser in [13] sees various positive indicators that VR-UML can show advantages where more complex and multi-diagram models are involved (and by inference hypermodeling). He anticipates that the immersive experience of UML models in VR adds qualitative aspects that users prefer.

Romano et al. in [14] observed that the use of immersive virtual reality leads to higher satisfaction among the participants. Their findings suggest that virtual reality might be a viable tool in software visualization, and their experiment seems to justify further research on its use in this area.

The aim of this research is to investigate the use of VR as a means to support as well as improve the efficiency of distributed software design activities. To achieve that, we employ the design science research methodology [15] to design and evaluate a VR software design environment that enhances the collaboration experience between remote or geographically distributed software developers.

In particular, we address the following research questions:

• RQ1. What is the impact of the created VR environment on the efficiency of collaborative, distributed software development?
We evaluate the collaboration efficiency when using the VR software design environment compared to a non-VR software design environment.

• RQ2. What is the impact of the created VR environment on the recall of design information during collaborative, distributed software development?
We evaluate the recall of design information when using the VR software design environment compared to a non-VR software design environment.

• RQ3. What are the perceptions and preferences of users when using the VR-based compared to the non-VR software design environment?
We collect the perceptions and preferences of users to explore the opportunities and challenges that were incurred by using VR software design environments. These perceptions will be used to further improve the created environment in order to support the design activities of distributed software developers better.

1.1. Structure of the paper

The background and related work are presented in Section 2. The approach that includes the design and creation of the VR environment is described in Section 3. The evaluation of the VR environment and experiments are presented and discussed in Section 4. Finally, concluding remarks and future work are presented in Section 5.

2. Related work

2.1. Software design, modeling and collaboration

There are several methods that aim to simplify complex and long-term software development processes. Several methods are known for projecting UML into three-dimensional space because large 2D diagrams might be hard to read and understand. Casey and Exton proposed a new method [16], where the elements in the diagrams are geons. Geon is basically a simple three-dimensional nonstandard UML element, e.g. square, pyramid or sphere.

Other approaches use the advantages of 3D space and layered 2.5D diagrams [17] or virtual reality [18]. The Ferenc prototype [19] supports these layered diagrams in web applications with real-time collaboration and visualization of the user’s editing history. The conversion of diagrams from 2D to 3D followed by visualization in augmented or virtual reality is researched in SAR process [20] or VisAr3D [21]. They are focusing on large-scale systems and experimenting with virtual and augmented reality in relation to education.

There are papers dealing with collaborative modeling drawing attention to the importance of the human element and its support [22, 23]. A different way of collaborative modeling is OctoUML [24] which uses sketchy diagrams and transforming them into formal UML notation. OctoUML supports a lot of inputs, e.g. interactive whiteboards. Interaction between a UML model and a human was researched by Anders Mikkelsen et al. [25]. They studied the interaction in augmented reality on Microsoft Hololens devices. They tried to make UML simpler by interaction, not by changing the UML itself.

Fukas et al. [26] adopted the 3C model (coordination, communication, and cooperation) and declared that awareness mediates and fosters all three aspects of collaboration. We can use this model as a base for analyzing and designing groupware.

Dourish and Bellotti [1] defined awareness as “an understanding of the activities of others, which provides context for one’s own activities”. Gutwin and Greenberg [27] claimed that every collaborator should be intuitively aware of current related aspects, such as who is in the workspace, where they are located, what they are working on, as well as past related aspects, such as how this artifact came to be in this state or who made this change and when. They confirmed the importance of
2.2. VR tools supporting software modeling

represents one instance of the GenMyModel modeling tool for studying UML modeling. It can be used as a platform for collaborative environment for UML modeling CoLeMo [35] designed for participants comparable to editing a model on a whiteboard together in one room. For 2D UML notation do not support a natural way of teamwork on a 2D user workspace. However, this does not solve the problem of readability of complex and large-scale UML models of software systems. Many research papers propose that modeling and visualizing a system in 3D space can eliminate this problem and introduce improvements.

The importance of collaboration between stakeholders is also confirmed by the observation of project failures in [30]. The six most common reasons include incomplete requirements, lack of user participation, lack of resources, unrealistic expectations, lack of executive support, and change in requirements and specifications. None of them deals with programming languages, technological environments, or hardware components. Five of the six mentioned reasons are due to the interaction. Collaboration and communication in the development of the software project are important for the execution of successful and effective projects.

Franzago et al. in [31] classified existing approaches of collaborative Model-Driven Software Engineering (MDSE) from 106 selected papers clustered into 48 studies. They summarized the most important characteristics of collaborative tools:

• modeling language (mostly UML),
• editor (for CRUD modifications and operations with model elements),
• application domain (mobile, web or business application, and orthogonally: user interface, business logic, data layer, etc.),
• multi-view support (using decomposition to reduce complexity),
• versioning support,
• conflict management (mainly conflict detection or locking elements to prevent conflicts), communication support,
• collaboration type (Synchronous in real-time or Asynchronous work on the same artifacts but offline or not at the same time).

Yigitbas et al. [32] claimed that standard modeling applications (Lucidchart [33], GenMyModel [34], etc.) containing remote collaboration on a 2D UML notation do not support a natural way of teamwork comparable to editing a model on a whiteboard together in one room. Lucidchart [33] supports UML and Business Process Model Notation (BPMN). GenMyModel [34] supports UML and users can see where their co-workers are editing the model but do not have voice chat to communicate with them. Another tool is the collaborative learning environment for UML modeling CoLeMo [35] designed for participants studying UML modeling. It can be used as a platform for collaborative software design. Teaching software engineering in Virtual reality was proposed by Herpich et al. [36] in their prototype Virtual Lab through using panels embedded into the 3D virtual environment. Each panel represents one instance of the GenMyModel modeling tool.

2.2. VR tools supporting software modeling

Collaborative modeling in virtual reality is proposed in [32]. The authors believe that the use of 3D cubes (Fig. 1) and tubes instead of 2D class rectangles and relationship lines could result in a more natural experience for users because the real world consists only of 3D objects. Every cube side shows the same 2D Class notation (name, attributes, and methods) similar to 2D UML. They built a prototype with these cubes on an unlimited planar grass field under a blue sky to depict a pleasant real-world environment for the user to walk. Visualization of the cubes is similar to visualizations of BPMN in 3D or VR [37].

It seems that trivial presentation of the boxes on a flat terrain does not contribute to reducing complexity nor promoting comprehensibility. It will also be probably difficult to get an overview of where other collaborators are located. Maybe the unused 3D space in the air would help with a very complex diagram to reduce the number of edge intersections and make it easier to layout diagrams or create the clusters. Authors could also use the top side of the cube for additional information (stereotype, state, module, etc.).

In the evaluation process in [32] the participants had to create one class diagram in VR and another one in the web application (Lucidchart). On average, participants needed more time in VR than in the Web application to complete their tasks (13:42 versus 07:35). Also, the number of the errors per minute was higher in VR. Better VR results were achieved in subjective evaluation (Interactivity, Co-Presence). “13% (3 out of 24) wanted to use the VR app over the web application and 37% (9 out of 24) vice versa. However, half of the participants indicated that they would like to use both applications, each for specific situations” [32].

Augmented reality with MS Hololens was tested in [38] for students to make UML more accessible. Zenner et al. [39] proposed a system that visualizes EPC (Event-Driven Process Chain) models in VR 3D. Their approach focused on model understandability and user experience by comparing the HMD VR experience with viewing a 2D process model. They observed that users understand the model faster in 2D (objective effectiveness and efficiency) while the users interest and immersion is higher in VR.

In augmented reality around the desktop, we can visualize other diagrams [40] behind the primary one for additional information (e.g. authors, element types, tags, metrics, bugs, design patterns, bad smells, etc.). The same approach is possible in VR behind or around our whiteboards.

In software engineering monitors with a mouse and keyboard are mostly used, which may limit natural perception and creativity of users [41]. Therefore, it might be very useful to use virtual reality as a tool for software development [42]. In the 3D space of VR, we can see the distance and scale of objects better than on standard monitors. VR technologies are now more affordable, which also creates new opportunities in the field of software design [43]. The following advantage of virtual reality is the indirect effect of movement on human creativity. Movement, for example, walking, has a positive effect on thinking, so it can increase the effectiveness in software development [44]. In real life, there is a possibility to determine indoor location of humans researched by Arjan J.H. Peddemors et al. [45]. They used Bluetooth technology of the mobile phone to determine where people are located in the building. They found out that the presence of people could be detected with room level granularity. We can use this approach in our prototype later, to determine the presence of the user’s location.
Information and Software Technology 166 (2024) 107369

2.3. Discussion

At the end of our effort to create a short and concise overview of similar modeling tools and approaches, we can state that in addition to standard industrial CASE systems such as IBM Rational Software Architect [46], SPARX Enterprise Architect [47] or Visual Paradigm [48], there are also academic experiments and concepts focused on modeling, collaboration, 3D space and VR utilization, etc. In the field of VR modeling tools, we found VmodlR as a comparable project, which also solves the enhancement of the immersion and satisfaction with collaboration using VR, but its approach is a little bit different: analysts and designers work on one model in VR 3D space, while we use 3D space for a dynamic and variable arrangement of whiteboards which developers are used to work on in their offices and laboratories. The advantage of whiteboards is also the possibility of distributing parts or modules into them as layers and their interconnections in this 3D space (see Sections 3.2 and 3.3). With 3D elements (cubes) on green grass under a blue sky in VmodlR [32], you cannot deal with this possibility of layering the model for reasonable and rational usage of 3D space.

At the end of the chapter, we created an overview (Table 1) for a short comparative analysis of particular tools, where we can find two methods for collaborative modeling in virtual reality.

<table>
<thead>
<tr>
<th>Tool</th>
<th>UML Modeling</th>
<th>VR/AR Support</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucid Chart [42]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>OctoUML [1]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>GenMyModel [41]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Fer 3D UML [6]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CoLoMe [43]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VRModlR [37]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DrawIO [32]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IBM [27]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Enterprise Architect [28]</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>AR UML [44]</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Visual Paradigm [29]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>VR Sketch (our prototype)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

3. Our approach of collaborative software design in virtual reality

We employ the design science research methodology (DSRM) [15] to create and evaluate a VR software design environment. DSRM is an iterative process that involves problem exploration, the creation of a solution, the evaluation of the solution, and adaptation.

The request for research and development from Gratex International was whether we are already able to achieve quality and speed in collaborative modeling in VR comparable to the classic desktop tool and whether it would speed up (or at least not slow down) the creation of models and achieve a better feeling of collaboration and encourage more frequent collaboration.

Another idea of our approach is to avoid the problems with conflicts and merging of the results from the work offline (Fig. 2) and support parallel collaborative model-driven development and collaboration of software development teams in virtual reality (Fig. 3) to reduce communication gap between distributed developers working on software models. The goal is to increase user and team productivity and reduce the costs of remote software development. We designed a new tool in VR that supports sketching of the software structure in the Unified Modeling Language (UML) [49] as standardized and widely used for the creation of software models that describe the architecture and functionality of the created system.

According to related work in Section 2 we rely on the following assumptions and hypotheses, which we would like to prove/test in experiments with our tool:

1. developers might not work worse or slower in virtual reality than on standard desktop systems,
2. developers might be more immersed in VR into their work and their models than on standard desktop systems and they might remember more of the model content from VR,
3. developers might have a better feeling of collaboration because they will see the presence and materialized statures, movement, and gestures of their remote colleagues through the avatars in VR.

3.1. Concept

From the main idea of the proposed method and from our assumptions from related work we can derive main features which have to support:

1. Parallel collaborative work on software models in modules or layers [13,17,19,50] and similar to draw.io [51] or GenMyModel [34] online in Virtual Reality, increasing productivity and productivity in work compared to standard or offline and uninformed noncollaborative work, visualizing trace of the particular sequence of work of co-workers to know who, where and what they are working on (Fig. 5). The layer could be defined as a set of diagram elements with particular commonality and collective operation [50]. A layered diagram is a partially ordered set of sequential, alternative, or orthogonal layers [50] and we could use a special object algebra of addition and subtraction [52,53].
2. Visualizing parallel whiteboards on which the other developers are working (collaborators can work on the same layer, too) supporting reduction of the complexity of large models with decomposing the extensive models to the real modules or components with particular structures, supporting other use cases, visualizing similar parts on the other component layers to not reinvent the wheel and reduce redundant and vague elements. Whiteboards provide space and comfort to developers and prepare the environment with the partitioning of the model into several worlds or interconnected layers containing: particular modules, author and time versions, particular type clusters (e.g. GUI, Business services, DB services). Developers can also highlight patterns or antipatterns in separate layers or converge to a lean architecture using the DCI approach with abstract domain classes, roles and scenarios [54] in separate tiers.
Table 2
Comparison of our prototype with similar systems.

<table>
<thead>
<tr>
<th>Feature</th>
<th>VR Sketching</th>
<th>OctoUML</th>
<th>draw.io</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported notations</td>
<td>UML</td>
<td>UML</td>
<td>UML, BPMN</td>
</tr>
<tr>
<td>Supported UML diagrams</td>
<td>Class diagram</td>
<td>Sequence diagram, Class diagram</td>
<td>Many</td>
</tr>
<tr>
<td>Environment</td>
<td>Virtual reality</td>
<td>Desktop, interactive board</td>
<td>Browser</td>
</tr>
<tr>
<td>Sketching</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Shape recognition</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Speech recognition</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Class diagram</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Adding classes</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>All connectors type</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Editing position and text of class</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Automatic layout on connectors</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Editing line layout</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Class delete</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Line delete</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Text on lines</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Multiple users</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Synchronization in real time</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Routing server</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Table of connected users</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>User visualization</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Unique colour per user</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Back button</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Class copy</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Connectors copy</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Diagrams in layers</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Unlimited space in one diagram</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Saving diagrams</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

3. Visualizing tiers with model and source code behind the diagram in various object-oriented languages (python, Java, C++, ...) to create a structural source skeleton of the project.

To reduce the complexity of our prototype VRsketch, we have decided to reuse open source modules for neural network and speech recognition (gray packages in Fig. 4, white packages are implemented as a part of this project).

Basic cognitive processes, such as focus and feelings (positive and negative), were indirectly investigated in post-task questionnaires using NASA TLX. Information processing, learning, and memory were investigated with RH2 and did not perform worse than the standard desktop, but neither did they perform better.

According to [55], we can derive the most important cognitive processes in collaborative modeling from their detailed classification. These processes include solution search, problem and solution design, problem analysis, synthesis, comparing, judging, reasoning, evaluating, and decision making.

Comparing creativity and decision making in VR versus standard desktop systems was not our goal, but it is certainly an appropriate topic for future work where we would engage in a broader perspective with colleagues from psychology and cognitive science. We were not motivated to increase creativity and decision making and did not pursue this. Instead, we have created a functional modeling tool in VR and tried to evaluate it against a similar desktop system. We did not want to end up like Yigitbas et al. [32], which was confirmed in the first phase of our research but after improving the user interface we got better results. Most importantly a good feeling prevailed in both phases of the experiments to motivate collaboration and satisfaction.

3.2. Main functionality of our prototype

The prototype provides a useful tool for experimentation and evaluation of software modeling in VR. It includes the standard functionality of a collaborative software modeling tool:

1. Diagram editor (currently UML class diagram as the most used);
2. Support for collaborative parallel networking for multiple developers, Parallel collaborative work on software models in modules or layers;
3. Viewing collaborators as avatars, their movement at the whiteboard, view, hands and activity status (looking, drawing, erasing, pointing, speaking, etc.);
4. Displaying the work history of individual model designers in a unique color for each, which naturally fades over time;
5. The ability to display multiple tables and rearrange them;
6. Saving the results and uploading them to the system;
7. Ability to edit the diagram using VR keyboard and voice to insert text in addition to, Updating Data in Class Diagram Using Speech Recognition;
8. Image Recognition, Automatic recognition of sketches as diagram elements for more convenient and faster creation in VR;
9. Ability to teleport in VR space to individual charts;
10. Possibility of communication, voice-call over tool;
11. Visualizing parallel whiteboards on which the other developers are working;
12. Possibility of a deep and shallow copy of classes or entire collections and bundles of classes.

All these features have been tested and tuned to standard to make it possible to work in VR comfortably and as fast as possible.
3.3. Sketching and editing with tracking of work history

We currently offer our prototype with the possibility to model in a UML Class Diagram. For recognition of UML shapes, we used and trained artificial neural network model from the machine learning (ML) library TensorFlow [56]. The external web service receives points and sends the possibility of each shape separately. Due to the delay in calling the web service and recognition, it is necessary to call it with a new thread and not block the smoothness of the application logic. We optimize the points by trying to send them within almost the same distance. If the user is drawing a line too fast, we split the line into more points. However, the hand of the user is not stable and shakes slightly while drawing. If the user is drawing too slow, virtual reality tracks the controller very specifically creating point clusters difficult to distinguish by neural network. We solved this problem by defining the minimum distance of the next collected point.

Due to the ML tool’s dependence on network resources and availability, we decided to implement a simplified and faster approach for shape recognition in the new version. This decision aimed to reduce noise in the experiment, as the initial prototype sometimes inaccurately evaluated shapes, or the ML service experienced unavailability issues. The new algorithm is specifically adapted to the class diagram and expects only lines and squares as input. It considers any drawing that starts and ends in the class elements as a line, while accepting all closed paths (start and end in the same place) as class shapes. All other drawings are not evaluated and are treated as informal sketches or drawings.

Each user is identified by a unique color in the space of virtual reality and diagram. This color is used every time, when a user does some change on the whiteboard. After the change, there is a fade function, which makes the borders fade until they lose color and stay black (Fig. 5) and thus leaves a colored trace after each user from which we can sense and recognize his intention and way of thinking and the process of creating a model.

We decided to use the open source project, Drawing-VR, which supports painting in virtual reality. However, it was necessary to modify many of its features. We changed the painting method to collecting points, which are easy to synchronize over the network to particular users.

3.4. Improvements of the prototype after the first phase of experiments

We were able to understand the importance of evaluation in this area [57] and the necessity of improving the proposed visualization method, its prototype and interface [58] from the surprising results we have obtained in the first phase of evaluation: rejection of the first and second hypotheses forced us to improve our prototype, enrich it with new features, and optimize the interface. We made the following list of new and redesigned features:

1. new approach for sketching, creating and removing classes,
2. added new and more detailed avatars (Fig. 6a, b),
3. changed interaction with the whiteboard (by deleting the panel and adding a menu into the table)
4. changed the option for choosing relationship with UI panel for selecting new symbol and cardinality (Fig. 6c),
5. upgraded speech recognition (using more precise version from Google Cloud Services),
6. changed the GUI of editing class properties (header, attributes and methods) with a new, straightforward window (Fig. 6c),
7. added the possibility of a deep & shallow copy (we ensured that the user can easily copy classes and set references),
8. added the possibility of creating packages for reducing complexity,
9. improved whiteboards settings, teleports (Figs. 8 and 6d), and upgraded voice-call over tool.

Fig. 6. New version of avatars in VRsketch (a,b), GUI for editing class properties (c), improved settings of whiteboards and teleports (d), new association setting (e).
4. Experiment and evaluation

4.1. Evaluation concept

The main goals of our evaluation are to validate our approach and its tool and to compare it with other standard desktop tool collaborations’ efficiency, immersion, and satisfaction. Our research hypotheses will also be derived from this intention.

4.2. Experiment design

For the evaluation of collaborative modeling in Virtual Reality with our tool VRsketch we used a controlled experiment. In this experiment, the developers try to complete a software model with VRsketch and with a similar desktop application for collaborative modeling OctoUML v.4.2. Our goal is to compare satisfaction, comfort, cognitive load, effectiveness, and functionality of our prototype to a similar solution existing in a standard desktop setup.

Our controlled experiment involves two versions of the interaction technique (standard desktop application versus VR) with different user interfaces (OctoUML v.4.2 versus VRsketch v.2 and VRsketch v.3 in the second phase of experiment) as the manipulated independent variable [59]. The response dependent variables include task completion time, the number of syntactic errors, and the answers in the questionnaires (testing memory and satisfaction with collaboration), which are observable, easily measurable, and quantifiable using post-task questionnaires [59].

We expose participants to two interfaces, VRsketch and OctoUML, and task them with solving prepared assignments. Subsequently, we compare the results obtained from each interface. We aim to reduce confounding variables by fixing the environment and computer devices for the experiment as control variables, maintaining a nominal setting [59]. The attributes of our experiment participants, such as gender, age, knowledge, profession, and VR experience, are considered random variables.

4.3. Research hypotheses

Based on our assumptions in Section 3 that developers might work faster, be more immersed, and have a better feeling of collaboration in VR, we propose the following three research hypotheses RH1, RH2, and RH3.

**RH1**
Collaborative Sketching in VR is not slower compared to standard tools (e.g. OctoUML)

We will compare execution time of fulfilling a task in both tools VRsketch and OctoUML (i.e., duration of experiment).

**RH2**
Collaborative modeling in VR, developers will remember more design information from the whole model compared to standard tools (e.g. OctoUML)

We strived to establish equal working conditions for both groups, similar tasks and equal preparation time. We created $2 \times 2$ pairs (8 people) that covered all combinations of tasks and sequences of tools (Fig. 7). Therefore, we tried to get as many eights as possible, created and divided randomly from diverse backgrounds and varying levels of VR experience, including researchers, students, developers from the software industry, and professionals from small and medium software houses (see Table 3).

We chose OctoUML as a classic desktop modeling tool for its ease of use and efficiency. We used it for comparison with our VRsketch in our experiments because we assume that the other classic desktop tools such as DrawIO or Lucid Chart would have similar average times of task execution, but OctoUML uses sketchy diagrams with transformation into formal UML notation as VRsketch does and OctoUML supports interactive whiteboards. At the beginning of the experiments in 2019/2020, we did not know VmodlR [32], which is a similar tool in VR.

Currently, we evaluated our prototype with SUS [60], NASA TLX [61] and as a comparison with similar applications (see Table 2). For the experiment we selected OctoUML2 [24] that is a research tool for collaboration and sketching in standard environments.

NasaTLX [61] is a method of evaluating subjective factors (overall workload, task difficulty, time pressure, exhaustion, etc.) on a user who performed some observed activity. This questionnaire focuses on mental, physical and temporal demand, performance, effort, frustration, and stress level.

We strived to establish equal working conditions for both groups, similar tasks and equal preparation time. We created $2 \times 2$ pairs (8 people) that covered all combinations of tasks and sequences of tools (Fig. 7). Therefore, we tried to get as many eights as possible, created and divided randomly from diverse backgrounds and varying levels of VR experience, including researchers, students, developers from the software industry, and professionals from small and medium software houses (see Table 3).

We chose OctoUML as a classic desktop modeling tool for its ease of use and efficiency. We used it for comparison with our VRsketch in our experiments because we assume that the other classic desktop tools such as DrawIO or Lucid Chart would have similar average times of task execution, but OctoUML uses sketchy diagrams with transformation into formal UML notation as VRsketch does and OctoUML supports interactive whiteboards. At the beginning of the experiments in 2019/2020, we did not know VmodlR [32], which is a similar tool in VR.

Currently, we evaluated our prototype with SUS [60], NASA TLX [61] and as a comparison with similar applications (see Table 2). For the experiment we selected OctoUML2 [24] that is a research tool for collaboration and sketching in standard environments.

NasaTLX [61] is a method of evaluating subjective factors (overall workload, task difficulty, time pressure, exhaustion, etc.) on a user who performed some observed activity. This questionnaire focuses on mental, physical and temporal demand, performance, effort, frustration, and stress level.

4.3. Research hypotheses

Based on our assumptions in Section 3 that developers might work faster, be more immersed, and have a better feeling of collaboration in VR, we propose the following three research hypotheses RH1, RH2, and RH3.

**RH1**
Collaborative Sketching in VR is not slower compared to standard tools (e.g. OctoUML)

We will compare execution time of fulfilling a task in both tools VRsketch and OctoUML (i.e., duration of experiment).

**RH2**
Collaborative modeling in VR, developers will remember more design information from the whole model compared to standard tools (e.g. OctoUML)

We strived to establish equal working conditions for both groups, similar tasks and equal preparation time. We created $2 \times 2$ pairs (8 people) that covered all combinations of tasks and sequences of tools (Fig. 7). Therefore, we tried to get as many eights as possible, created and divided randomly from diverse backgrounds and varying levels of VR experience, including researchers, students, developers from the software industry, and professionals from small and medium software houses (see Table 3).

We chose OctoUML as a classic desktop modeling tool for its ease of use and efficiency. We used it for comparison with our VRsketch in our experiments because we assume that the other classic desktop tools such as DrawIO or Lucid Chart would have similar average times of task execution, but OctoUML uses sketchy diagrams with transformation into formal UML notation as VRsketch does and OctoUML supports interactive whiteboards. At the beginning of the experiments in 2019/2020, we did not know VmodlR [32], which is a similar tool in VR.

Currently, we evaluated our prototype with SUS [60], NASA TLX [61] and as a comparison with similar applications (see Table 2). For the experiment we selected OctoUML2 [24] that is a research tool for collaboration and sketching in standard environments.

NasaTLX [61] is a method of evaluating subjective factors (overall workload, task difficulty, time pressure, exhaustion, etc.) on a user who performed some observed activity. This questionnaire focuses on mental, physical and temporal demand, performance, effort, frustration, and stress level.

4.3. Research hypotheses

Based on our assumptions in Section 3 that developers might work faster, be more immersed, and have a better feeling of collaboration in VR, we propose the following three research hypotheses RH1, RH2, and RH3.

**RH1**
Collaborative Sketching in VR is not slower compared to standard tools (e.g. OctoUML)

We will compare execution time of fulfilling a task in both tools VRsketch and OctoUML (i.e., duration of experiment).

**RH2**
Collaborative modeling in VR, developers will remember more design information from the whole model compared to standard tools (e.g. OctoUML)

We strived to establish equal working conditions for both groups, similar tasks and equal preparation time. We created $2 \times 2$ pairs (8 people) that covered all combinations of tasks and sequences of tools (Fig. 7). Therefore, we tried to get as many eights as possible, created and divided randomly from diverse backgrounds and varying levels of VR experience, including researchers, students, developers from the software industry, and professionals from small and medium software houses (see Table 3).

We chose OctoUML as a classic desktop modeling tool for its ease of use and efficiency. We used it for comparison with our VRsketch in our experiments because we assume that the other classic desktop tools such as DrawIO or Lucid Chart would have similar average times of task execution, but OctoUML uses sketchy diagrams with transformation into formal UML notation as VRsketch does and OctoUML supports interactive whiteboards. At the beginning of the experiments in 2019/2020, we did not know VmodlR [32], which is a similar tool in VR.

Currently, we evaluated our prototype with SUS [60], NASA TLX [61] and as a comparison with similar applications (see Table 2). For the experiment we selected OctoUML2 [24] that is a research tool for collaboration and sketching in standard environments.

NasaTLX [61] is a method of evaluating subjective factors (overall workload, task difficulty, time pressure, exhaustion, etc.) on a user who performed some observed activity. This questionnaire focuses on mental, physical and temporal demand, performance, effort, frustration, and stress level.

4.3. Research hypotheses

Based on our assumptions in Section 3 that developers might work faster, be more immersed, and have a better feeling of collaboration in VR, we propose the following three research hypotheses RH1, RH2, and RH3.

**RH1**
Collaborative Sketching in VR is not slower compared to standard tools (e.g. OctoUML)

We will compare execution time of fulfilling a task in both tools VRsketch and OctoUML (i.e., duration of experiment).

**RH2**
Collaborative modeling in VR, developers will remember more design information from the whole model compared to standard tools (e.g. OctoUML)
We will compare the results of the post-task questionnaires in both VRsketch and OctoUML tools.

**RH3p developers prefer collaboration in the VR environment**

We will compare results from post-task questionnaires in both the VRsketch and OctoUML tools.

---

### 4.4. Apparatus

For the experiment, we used regular standard computers with larger screens. Standard headsets with microphones were used for communication.

Windows and Linux were used as operating systems but they played no role as the participants did not come into contact with its features.

The only relevant software was OctoUML for desktop computers and VRsketch for virtual reality.

In the first phase of the experiment (v1) we used two HTC Vive devices for VR, then in preparation step (v2) we used HTC Vive and Oculus Quest 1, and in the final phase we used 2x Oculus/Meta Quest 2.

### 4.5. Tasks and models for experiment

We need to compare two collaborative tools (VRsketch, OctoUML) with two assignments in our evaluation process. Therefore, we need $2 \times 2$ couples in two steps for the following all combinations of couples (1, 2, 3, and 4), tools (X, Y) and assignments (A, B). For these four couples, we need eight developers.

- the first step: 1XA, 2XB, 3YA, 4YB,
- the second step: 4XA, 3XB, 2YA, 1YB.

---

Fig. 9. Class diagram *Cinema* describing the basic schema of a cinema club.

Fig. 10. Assignment for blue designer.

---

We created two models for two assignments: *Cinema* and *Library*. In both assignments, the pairs consist of blue designer and green designer. Diagram Cinema in Fig. 9 describes the basic model of the cinema club.

Diagram in Fig. 11 that participants received before the task is without blue, green and gray classes (without classes that have to be completed through experiment). Blue, green and gray parts have the same degree of difficulty: sum of classes, attributes, methods, and relationships. Assignment for blue designer is in Fig. 10.

### 4.6. Procedure

The process of the experiment was divided into four steps (Fig. 12), where the developers filled out pre-task questionnaires, analyzed the
model and system, worked on the common model (main task), and completed post-task questionnaires.

The main task was divided into two subtasks:

1. Designer1 has to create a new part of the system (new classes with methods and relationships) and Designer2 has to create another part. In this subtask, we are observing their work on a separate problem (effectivity of tool editor, effort, etc.).
2. Second subtask contains collaboration on a mutual problem — developers have to create another part of the system together (new classes with methods and relationships). We are observing their communication along with cooperation on a common assignment.

Detailed description of the evaluation process:

1. Explaining purposes of the evaluation to test the possibilities of Virtual Reality and not to test the knowledge and experience of the participants. We prepared a short presentation on the OctoUML and VRsketch functionality.
2. Creating random couples to achieve more independent statistical results. In real life situations, senior designers and newcomers do not know one another either, so they are not well coordinated. In January 2020, we created 21 random pairs from 42 participants, and each pair worked with both tools (VRsketch and OctoUML) on different models (Fig. 7). We have obtained 84 results for comparison (42 from the VRsketch experiment and 42 from the OctoUML experiment). Participants used the names of their favorite actresses, actors, and singers from our list as a unique and unrepeatable token to anonymize them not only for GDPR purposes, but also to feel comfortable and stress free.
3. Preparing tools and equipment. After introduction we divided two pairs into separate spaces or rooms and prepared the necessary tools and equipment (two desktops for OctoUML, four desktops for pre-task and post-task questionnaires and assignments, two VR stages with headset) for them to achieve the best performance (Fig. 13).
4. Pre-task questionnaires. Couples had to complete a questionnaire focused on their knowledge and experience in UML, UML tools, OOAD, and English language level (see Fig. 16).
5. Tutorial. Lack of experience with Virtual reality (especially with UML class diagram sketching in VR) and OctoUML tool has forced us to create a tutorial focused on handling and manipulation with both of these tools. Tutorials consisted of tasks, where the developers were introduced with functionality of tools (creating/deleting classes, editing methods or attributes, creating/editing/deleting relations, copying classes, speech recognition or sketching) and tried them. The tutorial was written in the form of 10–20 short commands.
6. Model and Assignment. Participants received assignments and a model with missing parts which they had to complete and a short description of the model. They needed to analyze and understand the model, think about the missing classes, identify them and their position in the model.
7. Collaboration and modeling. Participants are working on the first task in the first tool after proper analysis of model and assignment. In the case of VRsketch, participants are working in separate rooms/stages with their own VR device. If they are working on OctoUML, they have two PCs with Skype installed for online voice communication.
8. Post task questionnaires. After finishing the task (and recording their duration time), the participants complete three post-task questionnaires. The first questionnaire (see Fig. 17) is focused on their knowledge of the model they were working on (relations between classes, method/attributes in classes, inheritance, and whole structure of diagram). The second questionnaire (see Fig. 19) is focused on immersion, collaboration intensity, comfort, level of interaction, and general feeling about the tool and the modeling in it. The third questionnaire is the SUS (see Fig. 18) and NASA TLX questionnaire on mental, physical, temporal demand, performance, effort, and level of frustration. Post-tasks are described in Section 4.9.
9. Swapping tools and assignments. Steps 5,6,7,8 are repeated, but with different tools and models to create the same sets of tool and task combinations: (Octo:Task1 & VR:Task2, VR:Task1 & Octo:Taks2, Octo:Task2 & VR:Task1, VR:Taks2 & Octo:Task1).
4.7. Participants in the first phase of experiments

In the first phase we had 42 MSc students for 21 couples (more than $5 \times 8$ developers) and we achieved 84 outcomes (2 steps $\times$ 42 participants) from 42 experiments (2 steps $\times$ 21 couples). They were master’s students from the Department of Computer Science at the Faculty of Informatics and Information Technologies (University of Technology). These students already had completed courses focused on analysis and design of software systems with UML notation. They were capable of creating or completing existing models because of their knowledge and circa three years experience. They represented newcomers and junior developers in the projects and software companies. The 42 participants were between 21 and 24 years old (12 participants female and 30 male). The graph in Fig. 14 presents the answers to the selected questions on the Likert scale 1–5. We will try to improve the set of participants and add senior developers to the experiments in the final phase of the evaluation of our second version of the prototype.

4.8. Participants in the second phase of experiments

In the second phase of experiments with the new version of the prototype, we had 80 participants and we achieved 160 outcomes (2 steps $\times$ 80 participants) from 80 experiments (2 steps $\times$ 40 couples). We used not only students from the departments of computer science at the Faculty of Mathematics, physics, and informatics, but also developers and senior developers from software companies. In addition to students, experiments were also supported by large software houses (Gratex, Softec, Ditec) and progressive small new startup companies (see Table 3). All 80 participants were between 21 and 54 years old. 50% of them were programmers, the remaining 50% were combined (analyst, tester, programmer, IT student, etc.). Altogether 17 participants were females and 63 were males.

Graphs in Fig. 15 present demographic structure of participants in age and experience, and the answers to selected questions in likert scale 1..5.

With experience from the first phase of experiments, we can use the advantage of a larger number of participants and wider result set from the second stage of experiments to analyze correlations between age, experience, knowledge of UML or VR, etc. on one hand; and number of errors, satisfaction with collaboration in VR, quality of responses from the content of the model, on the other hand. Although in some cases (communicativeness, experience with UML or VR) the answers or satisfaction with VR were slightly better, we did not find a statistically significant correlation on which we could establish serious deductions and recommendations, according to the structure of the participants or developers in software houses.

Our repeated experiments took 2 months — several hours per day. After analyzing the results and close observation of the participants’ activity it can be concluded that it is necessary to improve the training with/for VR tools to achieve better satisfaction and efficiency.

Controlled training lasts 15 min. It has to be extended in lege artis to several days creating a small real-world project for team members.

Table 3

<table>
<thead>
<tr>
<th>Source</th>
<th>Number of participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gratex International, developers</td>
<td>30</td>
</tr>
<tr>
<td>Computer Science Dep., students</td>
<td>30</td>
</tr>
<tr>
<td>Computer Science Dep., R&amp;D</td>
<td>4</td>
</tr>
<tr>
<td>Kinit Institute, R&amp;D</td>
<td>4</td>
</tr>
<tr>
<td>Me-Inspection, developers</td>
<td>2</td>
</tr>
<tr>
<td>Ditec, developers</td>
<td>2</td>
</tr>
<tr>
<td>Softec, developers</td>
<td>4</td>
</tr>
<tr>
<td>Luigi’s Box, R&amp;D</td>
<td>2</td>
</tr>
<tr>
<td>Evolveum, developers</td>
<td>2</td>
</tr>
</tbody>
</table>

Fig. 15. Demographic structure of participants in age (a) and experience (b), and the answers to selected questions (c).
who want or have to work and communicate with others in VR tools. Of course, it is also necessary to put emphasis on improving communication when modeling in VR, but we could apply this to any social human activity. Well-developed communication skills from IT experts is a requirement which is not often met. However, there is hope that by using VR and a VR avatars, their approach to others might change, but this is not the scope of this paper.

![Fig. 16. Sample of questions from pre-task questionnaires.](image)

### 4.9. Pre and post questionnaires

Before any introduction or task, we were interested in their level of knowledge in software modeling, UML notation, understanding diagrams, and English language level. In this chapter we use Likert scale of answers from 1 (poor, not at all) to 5 (very good, very familiar). Every participant of the experiment was asked the following questions:

- After completion of the task (working in VRsketching or OctoUML), participants were asked to fill out a post-task questionnaire divided into 3 parts.

In the first part, we need to analyze and compare their level of understanding of the models they had been working on (relations between classes, method and attributes in the classes) to find out the participant’s immersion into the task in virtual reality and with the standard desktop collaborative modeling application. In this set of questions, the answers were single-choice or multiple choice (from 6 to 10 possible options). Every designer (green and blue) had his own set of questions for every particular model (Library and Cinema). For example, the questions for the blue designer in the Cinema model were as in Fig. 17.

The questions are divided into the first blue part (new classes from the task of the blue developer), the second gray part (new classes from the common task with the green developer), and the last white part (already existing classes from the model).

![Fig. 17. Sample of questions for RH2 testing from post-task questionnaires.](image)

### 4.10. Results

In the first phase of experiments we obtained shorter completion time with the tasks in OctoUML (RH1 was not accepted). It seems, that newcomers in VR need more time to train with relatively new and unknown technology (interface in virtual reality). Only three pairs from the set of 21 couples performed better in VR than in OctoUML on standard PCs: they have (extensive) experience in VR from playing games in VR.

We found that developers are more satisfied with collaboration in Virtual Reality and feel better than in a standard desktop environment. RH3 was accepted. This result was promising and we were motivated to improve our VR prototype and prepared a second phase of experiments.

The evaluation of the results from the new series of experiments with the second version of the VRsketch prototype is satisfying. Despite the Covid-19 pandemic and the lockdown with restrictions (home office and online teaching) we found more participants for the experiment with the second version of the prototype. 42 IT students took part in the first phase of the experiment and 80 students and developers in the second phase.

We were able to perform extensive experiments to confirm our expectations for higher efficiency (time to complete the tasks in VR) and satisfaction. The mean of time in VRsketch is 8:54 min. and the median is 8:39 min. The mean time in OctoUML is 8:01 and the median is 7:27. The spread with respect to time is slightly smaller for the standard application (min = 3:00, max = 15:53, diff = 12:53) than for VR (min = 4:07, max = 18:14, diff = 14:07). Fig. 20 shows 21 results from the first experiment (labeled v1) with 42 people, 4 results of the preparatory experiment (labeled v2) with 8 people and 40 results from the second final experiment (labeled v3) with 80 people.

We can accept RH4 because the p-value is ~ 0.4296 on the significance level α = 0.05 and the p-value > α, therefore there is no statistically significant difference in favor of OctoUML between the times in the VRsketch and OctoUML.

The next Fig. 21 shows a comparison of the results in the first and second experiment, where it can be seen that the quality of the answers did not change and it was approximately the same for the desktop...
OctoUML and the VR environment VRsketch (first two graphs on the left). The results of runtime duration (last two graphs on the right) were improved in the second experiment for VRsketch and they were already statistically insignificant although they were still a little worse for VR than for desktop.

We cannot accept \( R_H^2 \): after collaborative modeling in VR, developers will remember more design information from the whole model/functionality in contrast to standard tools (e.g. OctoUML), because \( p\)-value is 0.741 on the significance level \( \alpha = 0.05 \) and the \( p\)-value > \( \alpha \), therefore there is statistically no significant difference between the answers in VRsketch and OctoUML in favor of OctoUML or VRsketch.

In the third step we can compare immersion in collaboration, intensity and positive feeling of collaboration in VRsketch and in standard desktop tool (e.g. OctoUML). Both rankings contain 42 sums of seven answers in normalized scales (1 to 10). From the graph in Fig. 22 it appears that developers are more satisfied with collaboration in VR and feel better than in standard desktop environments.

Specific questions in Fig. 22 are the same collections as in the first phase of experiment, described in 4.9. Figure shows that the satisfaction with VRsketch was still higher with VRsketch than with OctoUML in the second experiment. The difference between the values of satisfaction with VRsketch and OctoUML were larger.

We can accept \( R_H^3 \): developers prefer collaboration in a VR environment, because \( p\)-value is 1.197e−06 on significance level \( \alpha = 0.05 \) and \( p\)-value < \( \alpha \), therefore there is statistically significant difference between rankings of VRsketch and OctoUML in favor of VRsketch.

4.11. Resume

After the end of the second series of experiments in 2022, we analyzed the data obtained and found the following positive facts about the new version of VRsketch:

(A) The execution times in VRsketch2 (2021/22) are better than the times in VRsketch1 (2020).

(B) Execution times in OctoUML are not significantly better than the times in VRsketch2 (2021/22).

(C) Satisfaction with VRsketch2 is still high, developers are more satisfied with VRsketch2 (2021/22) than with OctoUML (RH3 is confirmed), and satisfaction with OctoUML has decreased in the second series of experiments.

(D) Ratio of task execution times in both experiments was better with our VR tool (VRsketch) versus the standard collaborative modeling tool (we used OctoUML) than in experiments with a comparable VR tool (VmodlR [32]) versus the standard collaborative modeling (Lucidchart): \( T_{VRsketch}/T_{OctoUML} < T_{VmodlR}/T_{Lucidchart} \) (8 : 39/7 : 27 < 13 : 42/07 : 35).
We achieved better results in the number of errors when editing the diagram during the experiment (1.775 errors on average per task in VRsketch versus 1.850 errors per task in OctoUML). In VmodlR experiments [32] they observed a mean error rate of approximately 1.5 error per minute versus 0.66 error per minute in the web application.

Participants prefer VRsketch to OctoUML in two questions: 1. I think I would like to use this system frequently (3.1750 vs. 2.9250 from 5 points in Likert scale), and 2. Would you like to use such a system in your work in the future after some improvements? (3.4375 vs. 3.4000).

According to points D, E and F, our participants worked almost as fast in VR as in the compared standard desktop environment, they did not make more mistakes and did not refuse to use VRsketch in the future.

We used similar libraries and the Unity environment as in VmodlR [32] and we also found (as they did) that the work in VR is not faster compared to standard desktop tools. However, it is considered better because of the feeling of immersion and perception of collaboration with a co-worker.

4.12. Discussion

In this paper, we stated that collaborative modeling in VR can now converge in speed to that of a standard desktop version and that VR is preferred by most developers in our experiment.

In addition, VR could gain importance within distributed teams, working remotely also due to pandemic situations, lockdowns, and restrictions. VR immerses people in collaborative modeling, and the change in work style would bring refreshment and possibly better results, as we mention in points 1 to 5 below. VR also makes sense by distributing models across interconnected layers or tables in its extended space.

After the result analysis and evaluation of the particular research hypotheses in the previous Section 4.10, we can derive the answers to the research questions from the beginning of this study:

- RQ1. What is the impact of the created VR environment on the efficiency of collaborative, distributed software development?
  
  Answer to RQ1: We have to conclude that VR tools converge to the efficiency of standard desktop tools. The results may be better if developers work longer with VR and get used to it, and headsets are updated as well. In the software domain, we need to improve the keyboard, voice input, OCR, etc.

- RQ2. What is the impact of the created VR environment on the recall of design information during collaborative, distributed software development?

  Answer to RQ2: The results of both experiments do not support our expectation that more intensive immersion in the virtual space with the model would improve the perception of its content and the retention of information.

- RQ3. What are the perceptions and preferences of users when using the VR-based compared to the non-VR software design environment?

  Answer to RQ3: Most developers preferred to work in VR after the experiment ended. In future work, it would be interesting to let them work with the VR tool for a few months on a real project and evaluate afterwards their feelings, preferences and proposals about the environment settings, size and sensitivity of objects, operations of creating, editing, copy/paste, delete, zoom, hide, filter, and headsets calibration, etc.

  It would be useful to take advantage of the developers’ openness to such a change of interface as VR where they appreciate the change of working style and the novelty of the environment, but it would be necessary to very sensitively and carefully evaluate the emerging comments and suggestions and incorporate them into the next improved versions of the environment.

  From the comparative analysis of the results, different properties of both versions, and conditions of both series of experiments, we can conclude as follows:

  - in the second version of VRsketch we have significantly refactored interface in Virtual Reality (e.g. improvement of VR keyboard, interaction with the whiteboards, selections of relationship attributes, speech recognition, teleporting, voice-call over tool, better avatars, see Section 3.4) and therefore we got better times,
  
  - in the second version of our experiments we divided participants into separate rooms (laboratories) and they were not satisfied with the collaboration in the desktop system (OctoUML) as much as with VRsketch.

  Although the task completion times in VR are still worse than in standard desktop software with a real keyboard and a mouse, there are obvious benefits from our experiences in experiments in 2020 and 2021/22:

1. VR modeling offers positive changes in work style, replacing all-day sitting at the computer with movement and changing sitting position to help creativity and increase effectiveness and positive effect on thinking [41,44],

![Fig. 22. Aggregated results over the sum of specific questions from the first and the second phase of the experiments.](image-url)
2. the collaboration with other human beings and co-workers is a more immersive experience in VR because we can see the movement of collaborator avatars, their gestures, where they are, and what they are observing or preparing to do,

3. VR can enlarge the space for modeling to be more comprehensible for reading and more comfortable for editing,

4. 3D space in VR is a better alternative to present the layers (using whiteboards, for example) for presenting particular components or interconnected distributed clusters,

5. 3D space in VR clearly visualizes the model layer and source code layers behind it.

Also, we obtained a neither positive nor a negative result: developers do not remember more information from work in VRsketch than they do from work in OctoUML, which means that the second research hypothesis was not confirmed.

On average, the times in OctoUML are still slightly better than in VRsketch, and we can improve the VRsketch times with a more in-depth training in VR, lasting several days focusing on a specific task. We are able to achieve even better times using alternative speech recognition without a face mask (mandatory due to the covid pandemic in the second series of experiments, as shown in Fig. 23).

Fig. 23. Second phase of experiments with the face masks.

We attempted to reduce the errors in the experiment by dividing it into three groups.

1. **Threats to Construct Validity**: In our efforts to design a suitable experiment, we aimed to create a correct procedure that allowed participants to learn both the VR and conventional desktop tools and become familiar with the tasks. However, we encountered a limitation concerning the time allocated for training with VR. The relatively short duration, lasting only a few minutes, might not have been sufficient to achieve comparable expertise when compared to using a traditional desktop system. Unfortunately, developers would not participate in an experiment in which they had to learn a new VR system for at least a day or ideally a week in a real project to become comfortable working in it naturally.

2. **Threats to internal validity** (to ensure that the observed effect really does exist due to the test conditions [59]). We tried to map the focus and task content of the experiment to the use of such systems in the real world. We tried to avoid subjectivity in the design of the procedure and tasks, the content of the questionnaires and the assessment of individual variables to avoid errors and so on. We tried to reduce any bias of the experiment designers that a VR or desktop application would be better in specific tasks.

3. **Threats to external validity** (results are generalizable to other people and other situations [59]). We tried to recruit participants from different areas and assign them to the experiment randomly and without conscious selection, e.g. by experience, age, gender, etc. We created conditions of equal environment, time and material used for the experiment.

We indirectly addressed some problem areas in our research, cybersickness [62,63], and cognitive load [64]. To track satisfaction with the tool, we utilized post-task questionnaires with SUS and NASA TLX ratings. The satisfaction responses also reflected the impact of these undesirable effects. Some participants rated the tool with lower satisfaction levels due to discomfort and dizziness, which cannot be entirely eliminated, even with collaborative modeling. While our main goal was to enhance collaborative work efficiency and intensity through VRsketch, we acknowledge that cybersickness and cognitive load are important considerations. We recognize that some developers may prefer the desktop application, and we do not expect them to work in VR all day. Instead, VRsketch aims to facilitate more effective and intensive collaboration, allowing collaborators to work together in a VR space, designing or reorganizing models using whiteboards and collaborative tools.

Batmaz et al. in [65] identified that a change in target depth negatively affects virtual hand interaction in peri-personal space with single-focal displays. However, multifocal displays do not suffer from this issue. These results are consistent with the findings of the Barrera and Steuerzlinger [66] for 3D TVs and Batmaz et al. [67] for VR and AR headsets. Notably, they did not observe corresponding problems for lateral movements in [65]. Furthermore, none of their experiment participants complained about eye strain.

We assume that collaborative modeling sessions in real-life scenarios would typically last for about 1–2 h. Given the short-term use, the risk of VAC might be not significant: our experiments were not completed by just two people in v1 experiment from 152 participants (44 people in v1 experiment + 8 people in v2 experiment + 80 people in v3 experiment). In addition, our system prioritizes the lateral dimension over the depth dimension, utilizing flat whiteboards with UML models and excluding 3D geons and background.

This might not be a major problem even in future work in interconnected layers, as our experience with 3D UML suggests that we do not perceive all layers simultaneously. Instead, we tend to view them in a layer-interconnection-layer manner. Nevertheless, this aspect warrants more accurate monitoring, such as through the use of the Simulator Sickness Questionnaire [68], rather than relying solely on indirect NASA TLX measurements. We might have better time results with a multifocal headset in the future.

One limitation of our project is that we used standard quality avatars and texts in VR. Our primary focus was on optimizing the keyboard, editor, and environment interface. While there are already sophisticated avatars [69,70] available in games and VR applications, there are also simpler alternatives [71,72] for scenarios where the visual fidelity is not a critical advantage, and computing power needs to be prioritized for other essential functionalities. Our approach falls in the middle ground; we aimed for avatars resembling colleagues and providing insight into their actions, such as their gaze direction and current activities. This is complemented by tracking their work (VRsketch highlights the currently updated classes using a unique color assigned to specific developer). For future development, we envision the possibility of constructing avatars as point clouds [73] directly from photos or by capturing the developer's face through the computer's camera.

Why did our experiments yield better results in terms of efficiency, number of errors, and preferences than our colleagues at VmodIR [32]? It is not possible to rigidly compare particular results because we did not use the same tasks, participants, and base desktop standard 2D tools for comparison (we used OctoUML and they used Lucidchart). We
did not know their research in 2020 (they published their tool and experiments in Models 2021), and they did not know our approach and results either. But maybe there is one objective reason for different results: the distinctions or inequality in the graphical user interface. We used 2D whiteboards in 3D virtual reality and not 3D boxes on green grass, which may have been more difficult to work with, although the engine inside both of our tools was comparable in terms of the libraries and environments used.

So far, we had no time or opportunity to test the effects of the infinite extent of the surfaces (whiteboards) in VR working on a large-scale project. In the future we will rely on simplifying the cognitive load and complexity by using the distribution of the project content into multiple layers — connected whiteboards that divide the project according to modules, use cases, etc. (e.g. also into a model layer and a source code layer).

After long discussions about the advantages and disadvantages of VR tools, we proceeded to perform the third short experiment, where an experienced professional software designer in a standard tool (Object-oriented UML) confronted one of the main authors of VRsketch, who lost in the ratio of times $T_{VRsketch}$ vs. $T_{OctoUML}$: 3:33 vs. 2:52 for the first task and 3:39 vs. 2:54 for the second task.

5. Conclusion and future work

The aim of this research was to support as well as improve the efficiency of distributed software design activities. We designed and evaluated a VR software design environment (called VRsketch) that supports the collaboration experience between individual and geographically distributed software developers.

We conducted two experiments to evaluate the: (i) collaboration efficiency, (ii) recall of collaboration experience, and (iii) user preference as well as satisfaction when using the VR software design environment compared to a non-VR software design environment.

The results show that there is no significant difference in the efficiency and recall of design information when using the VR compared to the non-VR environment. After the first experiment, we had to improve the prototype of the VR environment to achieve better collaboration efficiency. To do so, we had to redesign the GUI of the prototype VRsketch to increase the convenience and efficiency to complete VR tasks in the second experiment. The results have improved. The difference between the execution times (i.e., efficiency) in VRsketch and the non-VR design collaboration environment was not statistically significant.

Based on our experience from the observations of the two experiments, we could improve voice input and hand gesture control with haptics and a possible addition of extended reality, where we could combine real keyboards and large comfortable visualizations, models or whiteboard in VR around a desktop computer or a laptop to improve collaboration efficiency.

Regarding user preference and satisfaction, we have found that developers are more satisfied with collaborating in a VR environment rather than in a standard desktop environment, thus it is important to further improve the prototype for a better support of collaborative software design in VR.

5.1. Future work

Our prototype is open for adding more extensions in the future: for example, user customized real-world avatars and collaborative modeling of the other UML diagrams (Use Case Diagram, Sequence, Activity, and State Diagrams, etc.). We can create interconnected whiteboard layers in virtual reality for particular modules, clusters or element types (GUI, Business services, DB services) or for lean architecture with Data, Context, and Interaction in tiers [54].

We are designing a clustered shallow and a deep copy which will include copying of all dependencies and relations in a cloud of classes. Copied classes will act like clones, which means that every change in the original class (such as deleting/adding attributes, methods, relations) will be visible on the clone (copied class). This strategy can support pair-training of software modeling and the use of design patterns in large models for software houses.

CRediT authorship contribution statement

Martin Stancek: Software (collaboration in VRsketch), Data curation, Writing – original draft, Validation, Writing – review & editing. Formal analysis. Ivan Polasek: Conceptualization (concept of VRsketch, tracking), Methodology, Writing – original draft, Investigation, Supervision, Validation, Writing – review & editing, Formal analysis. Tibor Zalabai: Software (editor features in VRsketch), Data curation, Visualization, Validation. Juraj Vincur: Conceptualization, Methodology, Software (VR features, sketching in VR). Rodi Jolak: Methodology, Software (OctoUML version for evaluation), Investigation (cooperation in evaluation process). Michel Chaudron: Methodology, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We would like to thank Gratex International for providing the space and their developers for the experiments. We would also like to thank Faculty of Mathematics, Physics and Informatics Comenius University Bratislava for providing the facilities and Dr. Pavel Petrovic for encouraging and supporting his students to participate in the experiments.

This work has been carried out in the framework of the TERAIS project, a Horizon-Widera-2021 program of the European Union, GA no. 101079338.

Ethical issues

In this study, we have considered all the main ethical issues according to [74]: informed consent, beneficence - no harm, respect for anonymity and confidentiality.

References


