Visualization of Component-Based Software Architectures: A Comparative Evaluation of the Usability in Virtual Reality and 2D

Meike Schaller[1,2] and Andreas Schreiber[2(✉)]

[1] German Aerospace Center (DLR), Intelligent and Distributed Systems, Linder Höhe, 51147 Köln, Germany
{meike.schaller,andreas.schreiber}@dlr.de
[2] Otto-Friedrich-Universität, 96047 Bamberg, Germany
meike-barbara-anne.schaller@stud.uni-bamberg.de

Abstract. Software visualization provides a good opportunity to explore complex software architectures. But to reach a high level of usability it is important to evaluate such visualizations properly. We present the results of an usability study that we conducted to compare the visualization of component-based software architectures in both 2D and Virtual Reality (VR), based on different representations. Study participants had to conduct five interactive tasks. The results of our study shows that users are more likely able to complete the tasks in 2D than in VR, that they are more likely able to faster complete the tasks in 2D than in VR, and that they experience more satisfaction while using 2D than VR. Because evaluations of software visualization approaches are still rare, our results might help to create further study designs and offers some advice for the development of future visualizations—especially in VR.

Keywords: Software visualization · Virtual reality · Evaluation

1 Introduction

Development of complex software architectures [13] involves—beside all the technological advances—also several challenges: Source code gets more and more extensive and thus, intangible. This hampers the comprehensibility of large software projects which costs money and time (e.g., when a new developer comes on board of a software project).

Splitting code in several components is a first step to make source code more accessible. To reach the maximum possible success on both technology and user side it is necessary to represent the software in a way which supports users in quickly comprehending the software, including all components and dependencies. *Software Visualization* exactly deals with these aspects: simplifying and accelerating the process of becoming acquainted with existing software projects.

S. Yamamoto and H. Mori (Eds.): HCII 2019, LNCS 11569, pp. 208–222, 2019.
https://doi.org/10.1007/978-3-030-22660-2_14

Several studies haven shown that humans comprehend pictures and figures more easily than plain text such as millions of lines of code [9]. Most work available focuses on evaluating 3D visualization systems in comparison to 2D ones. *Virtual Reality* (VR) as an extension of 3D is also quite well investigated [10], but the research in comparing 2D with VR and evaluating them is still rare.

Our goal was to determine if and how VR is a supportive possibility to quickly explore large software systems or if it rather overstrains users due to some well-known disadvantages of VR, such as motion sickness or cognitive overload. Therefore, we measure the main criteria for good usability such as *efficiency*, *effectiveness*, and *satisfaction*; based on ISO-9241.

We present results of a planned usability study of two different interactive visualization approaches for software architectures: VR visualization using an island metaphor [21] and Web-based 2D visualization using graph layout algorithms [22].

After a short overview about software visualization as background information (Sect. 2), we present our contributions as follows:

- Brief description of two visualizations of component-based software architectures, which we choose to compare (Sect. 3).
- The design of our user study for measuring the usability of our software visualization in 2D and VR (Sect. 4).
- Some details on how we conducted the use study (Sect. 5).
- The results of our evaluation study with respect to effectiveness, efficiency, and satisfaction (Sect. 6).

2 Software Visualization

Current research has many innovative approaches for software visualizations with different representations. These approaches are based on *visual metaphors*, which adapts objects or situations from the real world for visualizations. Especially for 3D, popular visual metaphors include:

- *City metaphor*: One of the most frequently used real-world metaphors for software visualization. The foremost reason for its popularity would be the familiarity of the city concept [24]. Most users know that a city can be organized into districts, where each district can contain multiple buildings. These three hierarchical levels are the basis for most implementations of the city metaphor. Implementations exist for different media, such as VR [11,16], or programming languages, such as Java [1] or Go [7].
- *Solar system metaphor*: A metaphor to visualize Java-based projects [12]. Each package is mapped to a sun, which is being orbited by several planets at different orbits. While the planets represent classes, the orbits represent the inheritance level within a package. The size of each planet is mapped to the number of lines of code in its underlying class and the color helps to differentiate between classes and interfaces.

We experimented with other approaches for VR headsets. For example, a naive approach where we rendered 2D graphs with bundles and dependencies in 3D space by adding third dimension [22]. Another approach was based on the visual metaphor of electrical components, where bundles are rendered as cubes in 3D space and packages are stacked on top of these cubes [20]. In both version, user could fly through the 3D space by moving their heads. While moving head, users could point with a virtual cursor to any bundle or package, which then shows dependencies to other bundles. Despite the drawbacks of the visual representations, users reported motion sickness and dizziness.

3 Visualization of Component-Based Software Architectures

In our work, we focus on the visualization and exploration of *Software Architectures*, especially component-based architectures of software based on the OSGi framework for Java. OSGi modularizes and manages software projects and their services. OSGi projects includes *bundles*. Each bundle is a Java archive (`JAR`) with Java *packages* and a metadata file (`MANIFEST.MF`), which describes different information such as *dependencies* and *services*.

3.1 Island Metaphor for Virtual Reality

We use an *island metaphor* for the visualization of OSGi-based software systems in VR [21]. The entire software system is represented as an ocean with many islands on it (Fig. 1). Each island represents an OSGi bundle and is split into multiple regions. Each region represents a Java package and contains multiple buildings. The buildings are the representatives of the individual class types which reside inside of a package. Each region provides enough space to accommodate all of its buildings without overlapping, and hence the overall size of an island is proportional to the number of class types inside of a bundle (Fig. 2(a)).

The main entities of the OSGi service layer are *service interfaces* and *service components*. As these components are linked to Java class types, we visualize them as special building types. We visualize the relationships between the service entities with a *service connection node*. These nodes hover above the service interface and service component buildings at a certain height and act as connection points for them. Each node has a visual downward connection to its parent building in order for the user to quickly locate its associated service entity (Fig. 2(b)).

Our implementation of the island metaphor in VR, IslandViz [17,18], uses repository mining on the whole source repository and data mining on source code level to get all relevant data for the visualization. We store all data in a graph database for further analysis and visualization [19].

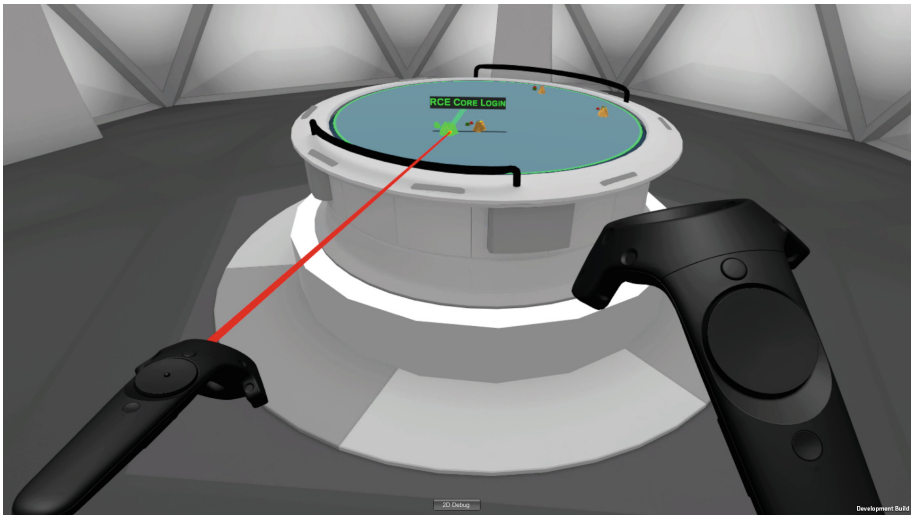
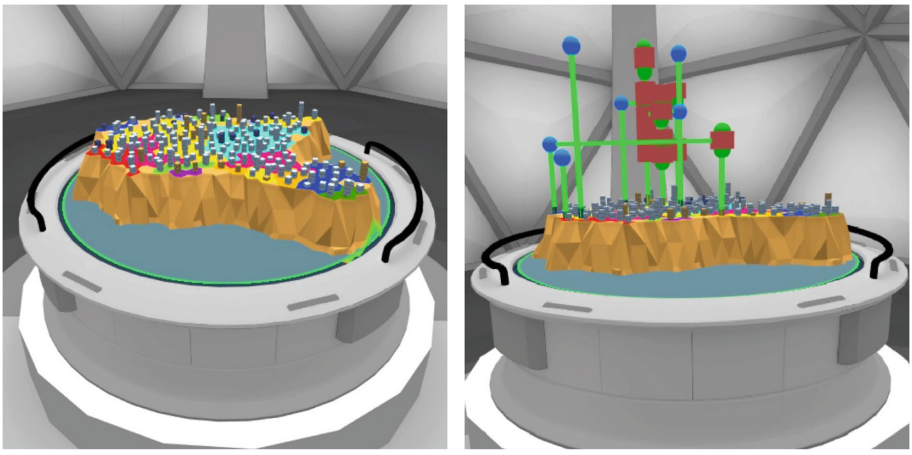


Fig. 1. IslandViz for VR: Bundles are rendered as islands on a virtual water level which is confined to extents of a virtual table. Elements of the visualization can be selected using hand controllers (HTC Vive).



(a) (b)

Fig. 2. Details of the IslandViz visualization: (a) a single bundle (island) with packages (colored regions) and classes (buildings); (b) services and service dependencies.

3.2 Graph Visualization in 2D

We use standard graph-layout algorithms to visualize the modularization of OSGi-based applications on different abstraction levels [22]. The visualization consists of different graph views: One bundle is represented as a node whose

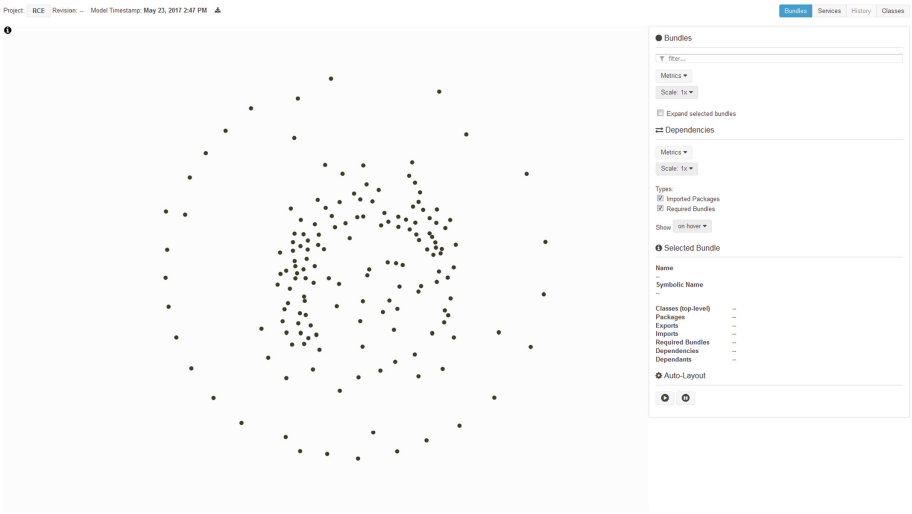
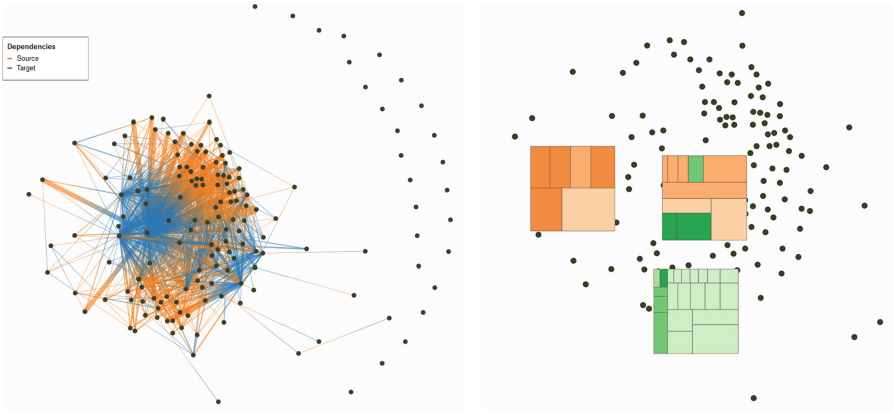


Fig. 3. Main view of interactive Web-based 2D visualization tool with visualization view (left side) and context information and configuration view (right side).

edges show dependencies to other bundles. Further-more one can change to different views (e.g., class and package view or treemap).

The visualization is embedded in a Website which consists of four areas (Fig. 3):



(a) (b)

Fig. 4. Examples of 2D visualization views: (a) Bundles and bundle dependencies rendered as force-directed graph; (b) bundles and package structure, rendered as treemaps where classes are assembled to packages by sharing the same color. The size of a class item is mapped to its lines of code.

1. *Project information*: static information about the OSGi-based application (e.g., name and time the model was generated)
2. *Navigation*: button-based bar for selecting the type of module that should be visualized
3. *Visualization view*: data visualization, depending on the selected module type
4. *Context information and configuration*: information about selected elements and configuration of the view (e.g., applying metrics)

Our implementation for 2D visualization uses the JavaScript library *Data-Driven Documents* (D3) [5]. It generates interactive data visualization using the Web standards HTML5, CSS, and SVG (Fig. 4).

4 Study Design

We designed our study to measure the three main aspects of usability, based on ISO-9241: *effectiveness* (accuracy and completeness with which users achieve specified goals), *efficiency* (resources expended in relation to the accuracy and completeness with which users achieve goals), and *satisfaction* (comfort and acceptability of use) [23]. Therefore, we measured both performance and self-reported metrics through two methods: *usability testing* and *questionnaires*.

4.1 Performance Metrics

In a first step of our study, users complete tasks while we measured the performance metrics *task success* and *time on task*. The overall goal of our study was to determine if the participants get a good overview of all important architectural elements (bundles, packages, classes, dependencies, and services) of the visualized OSGi-based software. Thus, the tasks needed to cover these components. Very challenging in this case was to find tasks which are feasible in both applications, because both applications had some functions which were not implemented in the other one. In the end—proved through some pretests—we determined the following five tasks:

Task 1: Select the bundle with the highest number of packages. Name the number of packages.
Task 2: Select an arbitrary bundle. Name the bundle which has the furthermost dependency and the number of its classes.
Task 3: Select the bundle `RCE Core Utils Scripting`. Name the number of its imports and exports.
Task 4: Show all existing dependencies between all bundles.
Task 5: Show all services and service components.

For each task, the success in completion was measured (i.e., giving the right answer without any help or supportive information). A task was not completed if a user gave the wrong answer or quitted the task. Time measurement started after the participant read the task description aloud.

4.2 Self-reported Metrics

In a second step, we measured self-reported metrics through two standardized questionnaires. Self-reported metrics are an effective way to gain insight into the users' subjective perception about the system and their interaction with it. The first questionnaire (*After Scenario Questionnaire, ASQ*) was developed by Lewis [14]; it measures the three main criteria effectiveness, efficiency, and satisfaction after completing a scenario or task. It consists of three items, phrased as statements which the user rates by a seven-point Likert scale from strongly agree to strongly disagree:

1. "*Overall, I am satisfied with the ease of completing the task in this scenario.*"
2. "*Overall, I am satisfied with the amount of time it took to complete the task in this scenario.*"
3. "*Overall, I am satisfied with the support information (online help, messages, documentation) when completing the task.*"

The ASQ is a good method to gain information about the three main aspects of usability: While the first statement answers the question for effectiveness, the second question focuses on efficiency. Satisfaction is measured through all three statements. Furthermore, it gives insight about which tasks the users found most difficult to find aspects in the system that still have to be improved. Additionally, the users can annotate each task.

The other standardized questionnaire that we used is the *System Usability Scale (SUS)* by Brooke [8]. It uses a five-point scale and consists of ten items which ask for the user's overall opinion about the application while half of the statements is phrased negatively and the other half positively (Fig. 5).

Please rate your level of agreement with the following statements:

	Strongly agree				Strongly disagree
I think that I would like to use the system.	○	○	○	○	○
I found the system unnecessarily complex.	○	○	○	○	○
I thought the system was easy to use.	○	○	○	○	○
I think that I would need the support of a technical person to be able to use this system.	○	○	○	○	○
I found the various functions in this system were well integrated.	○	○	○	○	○
I thought there was too much inconsistency in this system.	○	○	○	○	○
I would imagine that most people would learn to use this system very quickly.	○	○	○	○	○
I found the system very cumbersome to use.	○	○	○	○	○
I felt very confident using the system.	○	○	○	○	○
I needed to learn a lot of things before I could get going with this system.	○	○	○	○	○

Fig. 5. System Usability Scale (SUS) with ten items.

While the ASQ is evaluated through the rating's average, the evaluation of the SUS is a little bit different: Here, it is necessary to calculate a score. For this,

all items' score contributions have to be summed up while the items have different contribution, dependent on its either positive or negative statement (for more information see https://www.measuringux.com). After calculating the score, the interpretation of SUS scores based on further research says that results below 50 can be assessed as not acceptable, a result between 50–70 points is marginal, and results over 70 are acceptable [23].

4.3 Hypotheses

The gathered data supports proving our three hypotheses:

H1 Users are more likely able to complete the tasks in 2D than in VR.
H2 Users are more likely able to faster complete the tasks in 2D than in VR.
H3 Users are more satisfied while completing the tasks in Virtual Reality than in 2D.

We assume that concerning effectiveness and efficiency, users of the 2D visualization gain better results. Even though VR became more popular in the last years, using a "classical" computer screen is still the preferred way to get information quickly because users are familiar with it. Concerning the third hypothesis, we assume that users will experience more satisfaction while using VR. Reasons for that are the higher level of exploration and interaction. Moreover, it is still not that common to use VR in the working environment which could bring excitement and relief to the users.

5 Evaluation

We conducted the evaluation in January 2019. The participants were all Software Engineers at the German Aerospace Center. We divided them into two groups: Group 1 completed their tasks on the VR application, Group 2 used the 2D visualization. The division was necessary to avoid learning effects while testing which minimizes the risk of biased outcomes.

We tested both visualizations in the same surrounding—the VR Lab of the Department for Intelligent and Distributed Systems in Cologne, Germany. It offers a spacious room, which was especially important for evaluating the VR version with the HTC Vive. For evaluating the 2D application, we used a laptop with a 17-in.-screen.

First, the participants had to read a short written introduction, which explained the structure of the study (i.e., conduction of tasks and answering the questionnaires afterwards). Also the participants had to read an use case description, which introduced them into the situation (i.e., being a software developer who is new to a project). After reading the paper, the participants received an introduction to the evaluated application, including the explanation of all available functions and components.

The testing started after reading the task. For the 2D application, the participants read it by themselves, while for VR we read it for them (Task 3 as an

exception). A task ended with giving an answer to the question in the task. After each completed task the participants had to answer the ASQ—after completing all tasks they had to rate the whole system by with the SUS. We designed both questionnaires using the online tool https://www.soscisurvey.de.

6 Results

All in all, 14 participants (5 female, 9 male) took part in the study. 7 of them tested the VR application, 7 the 2D version while the attribution happened by coincidence.

6.1 Effectiveness

For the evaluation of the systems' effectiveness, the task success for each task has to be considered. We measured the task success with a score: 1 for success, 0 for fail. We calculated the score percentage for each task as average of 0's and 1's (Fig. 6). For both systems, the task success rates are quite high (i.e., that most of the tasks were completed successfully). Diving deeper into the diagram shows that for 2D, there is a higher success rate in two cases while for VR it is only in one case. For task 3 and task 5, the participants for both applications gained the same success rate.

With this result, the first hypothesis can be confirmed: **users are more likely able to complete the tasks in 2D than in VR.**

6.2 Efficiency

To determine how efficient an application is, measuring the time for each task is a reliable method. We measured the time with the timer function on the mobile phone—started after the participant read the respective task aloud. When the participant stated the answer of the task verbally, we stopped time measurement. To assess efficiency, we considered mean as well as median, which helps to identify possible outliers.

For both visualizations, we created a boxplot diagram to show an overview about the measured times for each task (in seconds). Comparing the boxplots for 2D (Fig. 7(a)) and VR (Fig. 7(b)), it becomes obvious that—except for task 4 and 5—the participants of the VR application need a lot more time to complete their tasks. For VR, there are two remarkable outliers, which are caused through some concentration issues by the respective participants. A remarkable difference can be found in task 3—which can be explained through the fact that the 2D version has a filtering option to find certain bundles. In VR, this function is not yet implemented. Overall, the results for efficiency also confirm the second hypothesis: **users are more likely able to faster complete the tasks in 2D than in VR**.

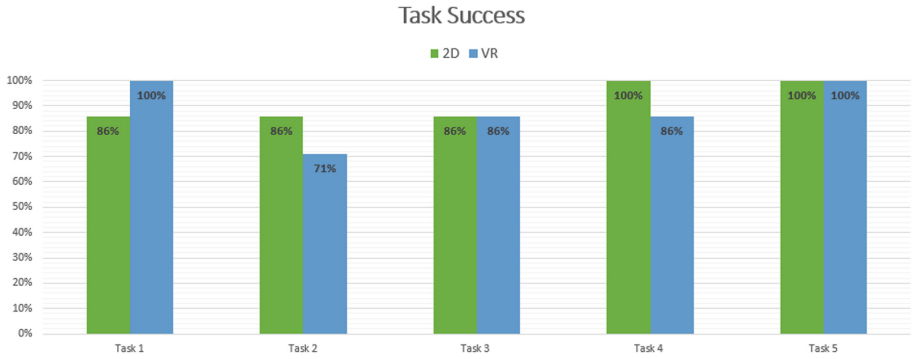


Fig. 6. Task success rates for both 2D and VR. It shows that for 2D, in two cases there is a slightly higher success percentage than in VR. In two cases the success rate is the same while VR gains only in one case better results.

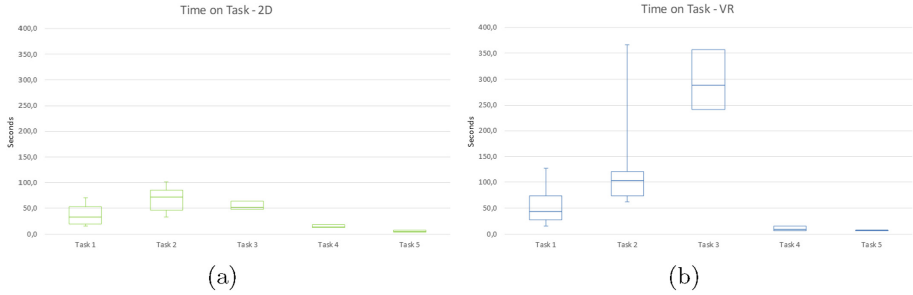


Fig. 7. Results for time measured on each task as boxplots: (a) Time on task results for 2D; (b) Results for VR with large outliers on task 1 and task 2 and remarkable more time needed to complete Task 3.

6.3 Satisfaction

We measured satisfaction through the questionnaires in two ways: satisfaction while completing a task with the ASQ and satisfaction concerning to the whole system, ranked by the SUS.

First, a closer look onto the results of the ASQ (Fig. 8). The left bar in each diagram indicates the rating about effectiveness ("Overall, I am satisfied with the ease of completing the task in this scenario"), middle one indicates efficiency ("Overall, I am satisfied with the amount of time it took to complete the scenario"). We measured satisfaction through all three statements, including the right bar ("Overall, I am satisfied with the support information when completing the task").

It turns out that concerning task completion and time on task, users seem quite satisfied for both visualizations. Remarkable high scores are on tasks 4 and 5, which are concerned with showing all dependencies and all services and service components. A large difference can be found on task 3, which required

finding a certain bundle. Participants of the VR application rated it quite worse because they did not have the option to search for it. In 2D, there is a filtering function already implemented.

For both visualizations, the rating about support information is not very high, except VR on the last two tasks. Possibly the participants perceived the virtual PDA–which must has to be used to complete task 4 and 5–as a support or help.

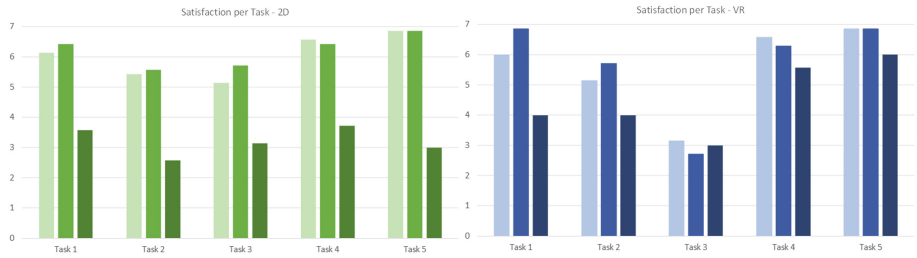


Fig. 8. Results for the three items about satisfaction on each task: (a) Satisfaction on each task for 2D; (b) Satisfaction on each task for VR. Remarkable differences on task 3 are explainable through the missing filtering function in VR.

For the SUS, it is important that the items have different weightings for the overall evaluation. While the items with odd numbers are phrased positively, the calculation is *participant's rating-1*. For items with even numbers, which are phrased negatively, it is *5-participant's rating*. The scores have to be summed up and then multiplied by 2.5 to obtain the overall SUS score (Fig. 9).

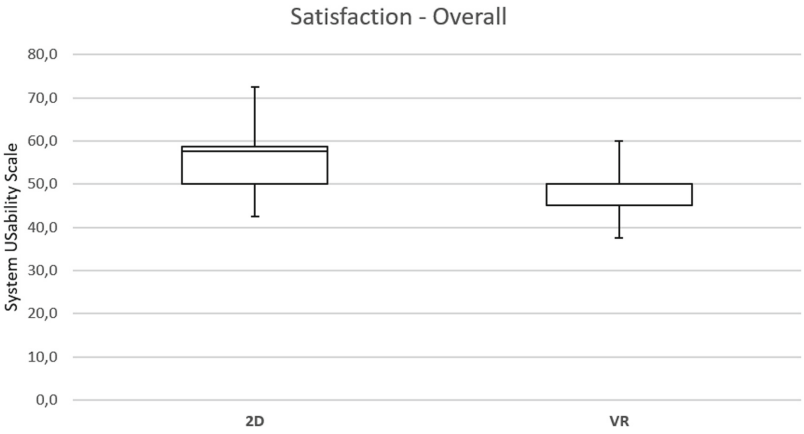


Fig. 9. Overall system satisfaction, measured by SUS. 2D is experienced slightly more satisfying than VR.

The results refute the third hypothesis: **participants experience more satisfaction while using 2D than in VR**.

For 2D, the mean value is 55.7 ($sd = 9.9$) while it is for VR 48.2 ($sd = 6.8$). Based on the interpretation of Bangor et al. [2], the usability for VR is not acceptable, for 2D it is marginal. But in both cases there are large outliers which explain the high number of standard deviations.

Taking a closer look at the ratings of the single items and the participants' comments, we found that indeed both systems seem interesting and beneficial concerning the use case (obtaining a first overview), but working with it in real life tend to be too time consuming and cumbersome, especially in VR.

7 Related Work

The research field of software visualization provides many approaches to explore software in different representations, but it is found that 62% of these approaches have never been evaluated or just offer weak evaluations (e.g., usage scenarios or anecdotal evidence). Another problem is that when there are any evaluations of a new software visualization approach, often they were conducted with participants which do not match with the target user group. Furthermore it is found that often the measured metrics are not well defined (e.g., *what does efficiency mean?*) [15].

If there are well implemented evaluations of software visualizations then they are based on different metaphors [25]. Comparing two visualizations in different representations (i.e., 2D and VR), however, is a new way for evaluating usability.

8 Conclusions and Future Work

The study's results provide a good first impression of how users perceive the two applications. Using a survey to rate each task is questionable in this case, because it seemed that it distracts the participants from their work flows. Especially in VR it turned out to be quite cumbersome, because the glasses had to be taken off and on again after each task. Furthermore, evaluating with an ASQ is not useful because concerning to the use case, the users do not have to complete tasks. The focus is more on exploring the application and gaining knowledge about the visualized project. To evaluate that, a survey with questions about the visualization would be more helpful (e.g., in VR, *for which component stands a region?*). Also, comparing a web-based 2D application with VR is not straightforward. In VR, physical constraints must be considered that do not appear while using 2D. Interpreting the results was also challenging because often the values of the standard deviation were quite high or the differences between 2D and VR were not significant enough. Even though the amount of 14 participants seems appropriate for discovering weaknesses, a higher number of participants would be helpful to get more precise results.

In summary, both applications have their advantages and disadvantages. Because VR is a relatively new field for most persons, it needs more time to

become familiar with it and to complete certain tasks. Especially in the work environment the additional benefit is questionable because it still needs too much effort to gain the desired goals. Future Work should focus on the implementation of supporting functions (e.g., a filtering option for finding certain bundles). Also conversational interfaces would be helpful which leads the user through the application or gives supportive information if needed. A good approach for that could be voice assistants based on *Natural Language Processing* (NLP) and *Natural Language Understanding* (NLU). Also, it seems that *Augmented Reality* (AR) or other future 3D output devices are good alternatives for such use cases [6].

Future work include evaluation of software visualization in AR supported by voice assistant and chatbots [3] and a deeper look into the usability of conversational interfaces for all kinds of visualizations (such as chat-based interfaces for 2D visualization [4]).

References

1. Balogh, G., Szabolics, A., Beszédes, Á.: Codemetropolis: eclipse over the city of source code. In: 2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 271–276, September 2015. https://doi.org/10.1109/SCAM.2015.7335425
2. Bangor, A., Kortum, P.T., Miller, J.T.: An empirical evaluation of the systemusability scale. Int. J. Hum. Comput. Interact. **24**(6), 574–594 (2008). https://doi.org/10.1080/10447310802205776
3. Baranowski, A., Seipel, P., Schreiber, A.: Visualizing and exploring OSGi-based software architectures in augmented reality. In: Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology, VRST 2018, pp. 62:1–62:2. ACM, New York (2018). https://doi.org/10.1145/3281505.3281564, https://doi.acm.org/10.1145/3281505.3281564
4. Bieliauskas, S., Schreiber, A.: A conversational user interface for software visualization. In: 2017 IEEE Working Conference on Software Visualization (VISSOFT), vol. 00, pp. 139–143, September 2017. https://doi.org/10.1109/VISSOFT.2017.21, doi.ieeecomputersociety.org/10.1109/VISSOFT.2017.21
5. Bostock, M., Ogievetsky, V., Heer, J.: D3: data-driven documents. IEEE Trans. Visual. Comput. Graphics (Proc. InfoVis) **17**(12), 2301–2309 (2011). https://doi.org/10.1109/TVCG.2011.185, http://vis.stanford.edu/papers/d3
6. Brath, R.: 3D InfoVis is here to stay: deal with it. In: 2014 IEEE VIS International Workshop on 3DVis (3DVis), pp. 25–31, November 2014. https://doi.org/10.1109/3DVis.2014.7160096
7. Brito, R., Brito, A., Brito, G., Valente, M.T.: GoCity: code city for go. In: 26th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). SANER 2019. IEEE, Hangzhou (2019)
8. Brooke, J.: SUS-a quick and dirty usability scale. Usability Eval. Ind. **189**(194), 4–7 (1996)
9. Diehl, S.: Software Visualization. Visualizing the Structure, Behaviour, and Evolution of Software, 1st edn. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-46505-8

10. Elliott, A., Peiris, B., Parnin, C.: Virtual reality in software engineering: affordances, applications, and challenges. In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, pp. 547–550, May 2015. https://doi.org/10.1109/ICSE.2015.191
11. Fittkau, F., Krause, A., Hasselbring, W.: Exploring software cities in virtual reality. In: 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT), pp. 130–134, September 2015. https://doi.org/10.1109/VISSOFT.2015.7332423
12. Graham, H., Yang, H.Y., Berrigan, R.: A solar system metaphor for 3D visualisation of object oriented software metrics. In: Proceedings of the 2004 Australasian Symposium on Information Visualisation, APVis 2004, vol. 35, pp. 53–59. Australian Computer Society Inc, Darlinghurst, Australia (2004). http://dl.acm.org/citation.cfm?id=1082101.1082108
13. Hasselbring, W.: Software architecture: past, present, future. The Essence of Software Engineering, pp. 169–184. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73897-0_10
14. Lewis, J.R.: Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the ASQ. SIGCHI Bull. **23**(1), 78–81 (1991). https://doi.org/10.1145/122672.122692. http://doi.acm.org/10.1145/122672.122692
15. Merino, L., Ghafari, M., Anslow, C., Nierstrasz, O.: A systematic literature review of software visualization evaluation. J. Syst. Softw. **144**, 165–180 (2018). https://doi.org/10.1016/j.jss.2018.06.027. http://www.sciencedirect.com/science/article/pii/S0164121218301237
16. Merino, L., Ghafari, M., Anslow, C., Nierstrasz, O.: CityVR: gameful software visualization. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 633–637. IEEE (2017). http://scg.unibe.ch/archive/papers/Meri17c.pdf
17. Misiak, M., Schreiber, A., Fuhrmann, A., Zur, S., Seider, D., Nafeie, L.: Islandviz: a tool for visualizing modular software systems in virtual reality. In: 2018 IEEE Working Conference on Software Visualization (VISSOFT), pp. 112–116, September 2018. https://doi.org/10.1109/VISSOFT.2018.00020
18. Misiak, M., Rawi85, Schreiber, A.: DLR-SC/island-viz: Islandviz 1.0, October 2018. https://doi.org/10.5281/zenodo.1464633
19. Nafeie, L., Schreiber, A.: Visualization of software components and dependency graphs in virtual reality. In: Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology, VRST 2018, pp. 133:1–133:2. ACM, New York (2018). https://doi.org/10.1145/3281505.3281602, http://doi.acm.org/10.1145/3281505.3281602
20. Schreiber, A., Brüggemann, M.: Interactive visualization of software components with virtual reality headsets. In: 2017 IEEE Working Conference on Software Visualization (VISSOFT) (2017)
21. Schreiber, A., Misiak, M.: Visualizing software architectures in virtual reality with an island metaphor. In: Chen, J.Y.C., Fragomeni, G. (eds.) VAMR 2018. LNCS, vol. 10909, pp. 168–182. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91581-4_13
22. Seider, D., Schreiber, A., Marquardt, T., Brüggemann, M.: Visualizing modules and dependencies of OSGi-based applications. In: 2016 IEEE Working Conference on Software Visualization (VISSOFT), pp. 96–100. IEEE (2016)
23. Tullis, T., Albert, B.: Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics, 2nd edn. Morgan Kaufmann, Bern (2013)

24. Wettel, R., Lanza, M.: Visualizing software systems as cities. In: 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, pp. 92–99, June 2007. https://doi.org/10.1109/VISSOF.2007.4290706
25. Wettel, R., Lanza, M., Robbes, R.: Software systems as cities: a controlled experiment. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, pp. 551–560. ACM, New York (2011). https://doi.org/10.1145/1985793.1985868, http://doi.acm.org/10.1145/1985793.1985868