




VR-UML: The Unified Modeling Language in Virtual Reality – An Immersive Modeling Experience

Roy Oberhauser^(✉) 

Department of Computer Science, Aalen University, Aalen, Germany
roy.oberhauser@hs-aalen.de

Abstract. Software models in the Unified Modeling Language (UML) can be created or automatically reverse-engineered and used for quickly gaining structural insights into larger, legacy, or unfamiliar software. But as the size, structural complexity, and interdependencies between software components in larger systems grows, two-dimensional viewing and modeling has limitations, and new ways of visualizing larger models and numerous associated diagrams of different types are needed to intuitively convey structural and relational insights. To investigate the feasibility of using Virtual Reality (VR) to create an immersive UML-based software modeling experience, this paper contributes a VR solution concept for visualizing, navigating, modeling, and interacting with software models using UML notation. An implementation shows its feasibility while an empirical evaluation highlights its potential.

Keywords: Virtual Reality · Unified Modeling Language · Software modeling · UML tools · Visualization

1 Introduction

Aristotle once stated “thought is impossible without an image,” and F. P. Brooks, Jr. asserted that the invisibility of software remains an essential difficulty of software construction - because the reality of software is not embedded in space [1]. Text-based program comprehension remains the norm in our day, despite the obvious limitations for this form of software comprehension, as evidenced in the low code review reading rates of around 200 lines of code per hour [2].

In general, modeling provides an abstracted or simplified representation of a system that can assist with understanding relationships between elements or concepts of interest. Typically, views are used to address stakeholder concerns and portray relevant aspects of a model. For visualizing the structural design of a software system, UML [3] has provided a unified and standard modeling notation. UML tools can support software developers via visualization, diagramming, model-based code generation, reverse engineering (from code to models), round-trip engineering, model transformation, and support for XML Metadata Interchange (XMI) [4] for transferring models between tools.

Commonly available 2D modeling depictions in standard modeling tools have limitations, and one can lose insight into the interrelationships across views, diagrams, and relevant model elements as the size of the model and views grows. Evidence includes [5], who concluded a network graph in VR was three times as good as a 2D diagram. For 3D UML, X3D-UML [6] determined a clear and measurable benefit in 3D UML software visualization, while a 3D UML tool case study [7] showed that a 3D perspective was intuitive and improved model comprehension. A VisAr3D experimental study with 18 participants [8] showed positive evidence for 3D for UML model understanding when many elements were present (and the third dimension's contribution), while showing that precision, efficacy, and time were not negatively affected.

VR could potentially assist with visualizing large and complex software models and their interrelationships simultaneously while also providing an immersive experience in the software models. VR is defined as a "real or simulated environment in which the perceiver experiences telepresence" [9], a mediated visual environment which is created and then experienced. VR has made inroads in various domains and become readily accessible as hardware prices have dropped and capabilities improved, increasing the accessibility and ubiquity of VR-based model visualization. VR-based visualization of *software* models for insights could rejuvenate the interest with software models in general and UML modeling in particular. In their study with 99 participants, [10] showed that VR resulted in better overall learning performance and higher engagement than textbook or video modes. A new approach via software model immersion could help rejuvenate the software modeling area and help transition from source-code only comprehension to more integrative use of visual models where it makes sense. VR offers a unique advantage in the unconstrained 3D space for visualizing, conveying, navigating, and analyzing complex and heterogeneous models simultaneously. As software models grow in complexity, an immersive environment could provide an additional visualization capability to comprehend the "big picture" for structurally and hierarchically complex and interconnected software diagrams, while providing an immersive experience for the UML models in a 3D space viewable from different perspectives. The sensory immersion of VR can support task focus during model comprehension while limiting the visual distractions that typical 2D display surroundings incur.

In prior work, [11] demonstrated the use of various metaphors for a VR immersion in software structures without the use of UML. VR-BPMN [12] described our solution concept for visualizing Business Process Model and Notation (BPMN) [13] models in VR. Next, VR-EA [14] presented a VR solution concept for visualizing, navigating, annotating, and interacting with ArchiMate [15] Enterprise Architecture (EA) models, while also describing our generalized VR modeling framework (VR-MF). Subsequently, VR-EAT [16] integrated EA tool visualizations into VR, in particular dynamically generated diagrams from the EA tool Atlas and its meta-model [17]. VR permits the extent of large models to be depicted and navigated visually, while overall interrelationships within and between heterogeneous elements, models, and diagrams can be indicated and considered. This paper extends our prior contributions with our solution concept VR-UML, which provides a way to visually depict and immersively navigate, model, and interact with UML-based software models in VR, enhancing these diagrams with

3D depth, color, and inter-diagram element followers, while supporting heterogeneous hypermodels in VR.

The remainder of this paper is structured as follows: Sect. 2 discusses related work. Section 3 presents our solution concept VR-UML. Section 4 then provides details on our prototype implementation that demonstrates its feasibility. In Sect. 5 VR-UML is empirically evaluated, and a conclusion follows in Sect. 6.

2 Related Work

Work on combining VR and UML includes Ozkaya & Erata [18], who propose their intent for a research framework of a conceptual modeling tool, Virtual Reality Unified Modeling Language (VRUML), but no VR realization details could be found. That VR features are not yet commonplace in UML tools is evidenced by Ozkaya [19], who systematically analyzed 58 different UML modeling tools without any mention of VR, and Ozkaya & Erata [20] who surveyed 109 practitioners to determine their UML preferences without any mention of VR. Related 3D (non-VR) UML visualization includes the aforementioned X3D-UML [6], VisAr3D [8], and the case study by Krolovitsch & Nilsson [7].

As to VR-based non-UML software model visualization, besides our own aforementioned prior software modeling in VR [11, 12, 14, 16], various metaphors in VR have been attempted. Schreiber & Misiak [21] and Nafeie & Schreiber [22] use an island metaphor in VR to represent components, packages, classes, and dependencies. Vincur et al. [23] applies a city metaphor to software analysis. Schreiber & Brüggemann [24] use a modular electrical component system metaphor in VR to visualize software components.

Regarding hypermodeling work, besides our own prior work, the survey by Bork et al. [25] comprehensively analyzed eleven visual modeling languages, including UML, ArchiMate, and BPMN, revealing heterogeneity in the specified modeling language concepts and techniques employed for concept specification. They found a lack of a common visual metamodel across various visual modeling languages, incompleteness, and thus difficulties in providing an overarching metamodel that could be used to simplify the specification and interrelations between various model types.

In contrast, the VR-UML solution concept realizes a VR-centric visualization of and immersive experience in UML models, providing automatic layout of views as stacked 3D hyperplanes, visualizing the reality of inter-view relations and recurrence of elements, and enabling interactive modeling in VR. Its support for hypermodeling, e.g., such that UML, ArchiMate, BPMN, and EA tool (Atlas) models can be visualized simultaneously in the same virtual space supports deeper cross-model analysis across various diagram types and stakeholder concerns. This capability may grow in importance with increasing digitalization as (automatically extracted) UML-based software models become more relevant to the business and EA and text-based code analysis (by non-developers) is no longer efficient or viable.

3 Solution Concept

With the upcoming challenges that increasing digitalization and IT infrastructure will bring to enterprise architecture, rather than viewing models in isolation and in separate tools, we envision the future of (software) modeling as integrative and holistic, utilizing and accessing various available models concomitantly. VR provides a unique medium of unlimited space and an immersive environment to support this modeling vision. Thus, the foundation for our VR-UML solution (shown in blue in Fig. 1) is our generalized VR Modeling Framework (VR-MF) [14]. It provides a VR-based domain-independent *hypermodeling* framework supporting multiple heterogeneous models while addressing three primary aspects of modeling in VR: visualization, navigation, interaction, and data retrieval. Relationships between elements can be shown in 3D space, and related elements can be grouped in 3D layers or views as appropriate. The capability to simultaneously visualize multiple heterogeneous models in VR is a key principle of our solution concept as realized via VR-MF. As depicted in Fig. 1, prior work based on VR-MF addressed enterprise architecture (EA) modeling with Archimate in VR called VR-EA [14], business process modeling in VR called VR-BPMN [12], and integrated EA tool data and visualizations demonstrated with VR-EAT [16] using the EA tool Atlas. ArchiMate models use a graphical notation consisting of a collection of concepts (approximately 50) to portray a wide scope of EA elements and relationships. On the other hand, BPMN models focus on business processes and consist of Business Process Diagrams (BPDs) composed of graphical elements consisting of flow objects, connecting objects, swim lanes, and artifacts. To meet commercial EA needs, Atlas, as a representative EA tool, provides access to diverse EA-related data in a coherent repository and meta-model and is not restricted to certain standards or notations. Thus, while UML is focused on modeling software structural aspects, ArchiMate, BPMN, and other EA models and views can convey other non-software aspects that may also be of importance to various stakeholders depending on their context and concern, especially as software becomes an integral part of the overall digital organizations and their processes. Thus, our *hypermodeling* principle as detailed in our prior work plays a fundamental role towards supporting heterogeneous VR model visualization with regard to our VR-UML solution concept.

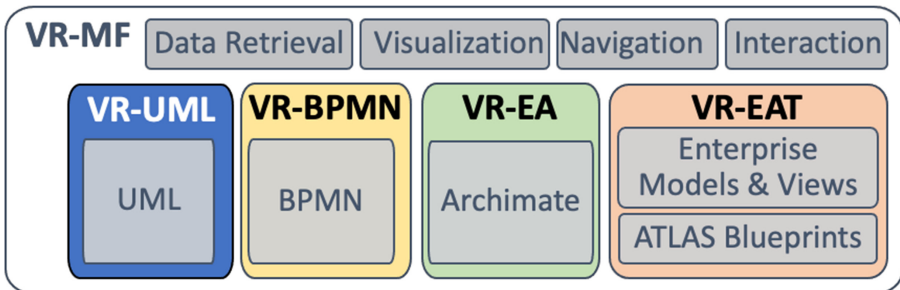


Fig. 1. Solution concept showing our new VR-UML solution concept within our VR-MF modeling framework, with VR-EAT, VR-EA, and VR-BPMN support.

Visualization. UML models use a graphical notation consisting of a collection of concepts to portray a wide scope of software elements and relationships. The diagram types can be categorized as structural diagrams (Class, Component, Composite structure, Deployment, Object, Package, and Profile) and behavioral diagrams (Activity, Communication, Interaction overview, Sequence, State, Timing, and Use case). These diagrams can participate in views used to convey information addressing concerns of specific stakeholders. While many visual options and metaphors can be considered for VR, diverging too far from the 2D diagrams and UML notations familiar to UML tool users would reduce diagram comprehension. Yet placing 2D UML images like flat screens in front of users would provide little added value in the 3D VR space. For visualizing and differentiating diagrams, planes are used to take advantage of the 3D space, with each plane representing a diagram. *Stacked hyperplanes* support viewing multiple diagrams at once, while allowing the user to quickly see an overview of how many diagrams of what type are available. Furthermore, stacked hyperplanes allow us to utilize the concept of a common transparent or invisible backplane to indicate common elements across diagrams via multi-colored inter-diagram *followers*. Stacked diagrams are a scalable approach for larger projects (compared to side-by-side) since the distance to the VR camera is shorter, and multiple stacks can be used to group diagrams or delineate heterogeneous models. Diagrams of interest can be viewed side-by-side by moving them from the stack via an affordance on a diagram corner we call *anchor spheres*, which can also hide or collapse diagrams to reduce visual clutter.

To distinguish UML elements types, generic (customizable) UML icons are placed on upper right and lower left of the top of the element. Rather than graphically modeling each element type separately, this enables us to quickly support many different element types using a common shallow box approach.

Due to the current lack of a common metamodel and/or inter-model specification language that can be used when visualizing heterogeneous models (cf. [25] in Sect. 2), we resort to a pragmatic approach of providing a basic inter-model annotation capability in VR.

Navigation. The immersion afforded by VR requires addressing how to intuitively navigate the space while reducing the likelihood of potential VR sickness symptoms. Two navigation modes are included in the solution concept: the default uses gliding controls, enabling users to *fly through* the VR space and get an overview of the entire model from any angle they wish. Alternatively, teleporting permits a user to select a destination and be instantly placed there (i.e., by instantly moving the camera to that position); this can be disconcerting but may reduce the likelihood of VR sickness that can occur when moving through a virtual space for those prone to it.

Interaction. VR interaction with VR elements has not yet become standardized. In our VR concept, user-element interaction is done primarily via the VR controllers and a virtual tablet. The virtual tablet provides detailed element information with CRUD (Create, Retrieve, Update, Delete) capabilities specific to each element as well as a virtual keyboard for text entry via laser pointer key selection. The aforementioned corner anchor sphere affordance supports moving/hiding/displaying diagrams. Inter-diagram element *followers* can be displayed, hidden, or selected (emphasized).

4 Realization

The VR-UML implementation architecture for our prototype is shown in Fig. 2. Due to its multi-platform support, direct VR integration, popularity, and cost, the Unity game engine 2020.2.0b4 is used with the SteamVR plugin v2.6.0b4. As shown, Unity uses various assets such as Models, Scenes, and Scripts, which in turn access external model files via our plugin adapter interface that parses and converts various model file formats (e.g., UML, BPMN, ArchiMate) to our internal generic object representation.

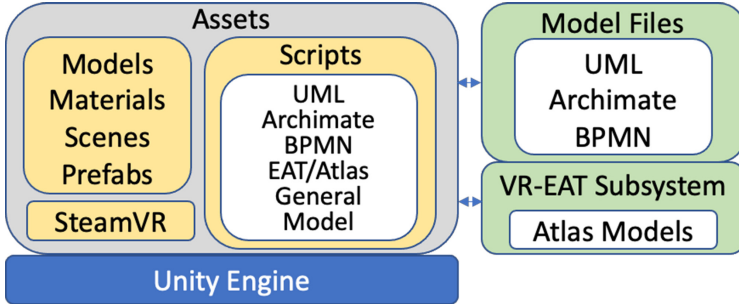


Fig. 2. VR-UML implementation architecture.

For text readability, an aspect that is irrelevant for 2D but which VR needs to consider is that the viewing angle from the user to the element (camera angle) can be dynamic based on the VR camera position in space (which is what is actually moved to “navigate”), thus the recognition and readability of elements must be considered from various angles. Thus, in VR-UML the diagrams and any elements they contain are raised slightly for a 3D effect and these visible side edges utilized for text placement to permit the text to be read from all sides in addition to the top. To support element delineation in space, rather than using clear elements with border outlines - as is typically done in 2D UML representations, in VR-UML a texture/color/material is used on all sides of an element to give it substance. However, in 3D space if the elements are opaque, then another element or relation could become hidden (and the user unaware of this), so a certain degree of transparency for diagram planes and for certain elements is used to ensure that relations and elements do not completely “disappear” within or behind other elements. Furthermore, a customizable color scheme, e.g., Coad et al. [26] or the colored layers used in the ArchiMate specification can be used to help distinguish UML diagrams and elements as models grow, since, in contrast to 2D, many elements can be depicted visually in VR.

One unexpected challenge in the UML visualization area is support for a common UML diagram interchange format between UML tools that contains positioning and layout data. While a mechanism for UML model exchange had been specified for UML 1.x using XMI, it only provides information on the model elements while lacking support for exchanging diagram and element positioning and layout information. This limitation is due to the UML metamodel lacking a standard way of representing diagram definitions. While UML Diagram Interchange (UMLDI) [27] was published in 2006, few UML tool

vendors appear to implement and support it. At the time UML 2.0 was published in 2005, UMLDI was unavailable for another year, so vendors may have ignored it and continued with their own proprietary format for maintaining diagram layout information. Most web-based UML tools and various desktop tools we tried support exporting only common image file formats, while some support exporting the model in XMI but lack any positional information. As XML can readily be converted into a JSON format, rather than relying on the older common XML format in UML, we wanted to investigate utilizing the newer and more efficient JSON format for UML model files. Various popular UML tools were analyzed to determine if they already used or supported a JSON format for UML. As StarUML uses JSON in their MDJ model files, VR-UML uses its UML JSON file format. In Unity, the JsonDotNet package was used in combination with quicktype to parse the JSON model file.

As shown in Fig. 3, at the highest level, an MDJ model file contains a single object of the type Project that includes the project name and ID as well as an array of the next level of objects. This array contains all saved Model Objects inside the project. These Model Objects contain another array of objects of different types, of which we focus on three: Diagram Objects (objects in a diagram and their positional data), Model Data Objects (objects and their model data including relations and Child Objects), and Collaboration Objects (all diagram data for sequence diagrams). While further objects for diagram types such as activity or flowchart would expand the types, they are similar to the sequence diagram in another object tree branch.

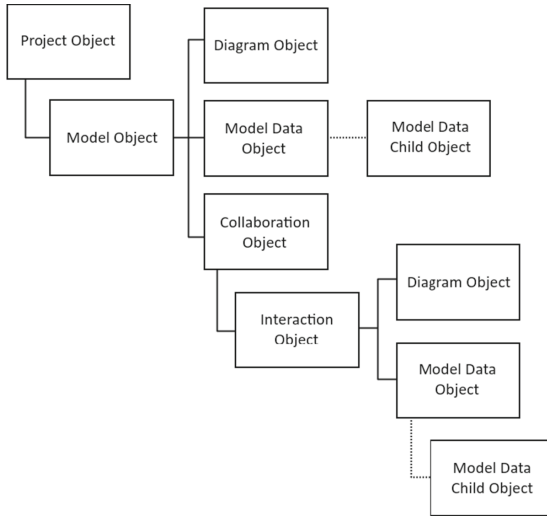


Fig. 3. Model file structure in JSON format.

To evaluate the practicality of the VR-UML solution concept and implementation prototype, a case study was used. Support is initially limited to the common UML diagram types: use case, class, sequence, and deployment. The travel agency example project provided by UML Designer [28] was used as a UML model basis and then imported to StarUML in order to get an MDJ model in JSON format. The model provides the basic UML diagram types known in the 4+1 view model [29]: a use case diagram depicting requirements in the scenario view (Fig. 4 left), a sequence diagram (Fig. 4 right) depicting runtime behavior in the dynamic or process view, a class diagram for depicting the internal structure in the logical view (Fig. 5), and a deployment diagram (Fig. 6) for depicting the physical view.

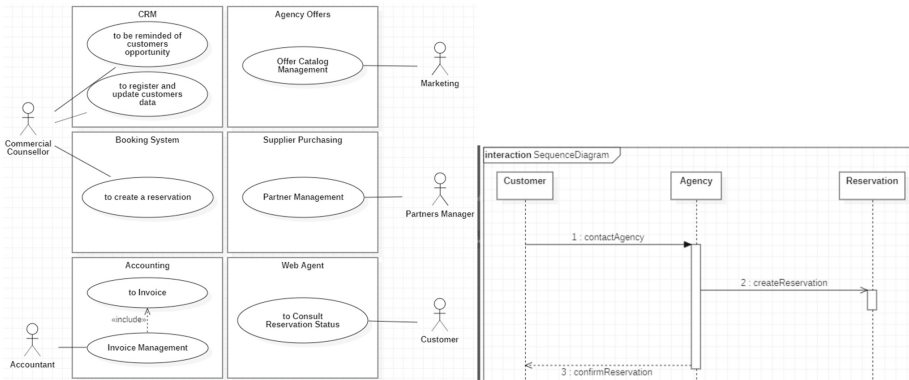


Fig. 4. Travel agency use case (left) and sequence (right) diagrams in StarUML.

The VR_UML visualization of the travel agency model is shown in Fig. 7, depicting *stacked hyperplanes* for this model. Colors help differentiate diagram types. Here the top grey plane shows a sequence diagram, the second purple plan the use case diagram, the third plane the deployment diagram in green, and the bottom a class diagram in red. Random colored *followers* along the invisible backplane (currently closest to the camera) are automatically generated between recurring elements across diagrams to follow participating elements across views (e.g., Customer (purple), Reservation (aqua) and Customer (light green), which recur in the class and sequence diagrams with details shown later), and can be used to quickly recognize recurring elements in other diagrams. *Anchor spheres* on the corner of each diagram act as affordances that supports expanding, collapsing, and moving a diagram.

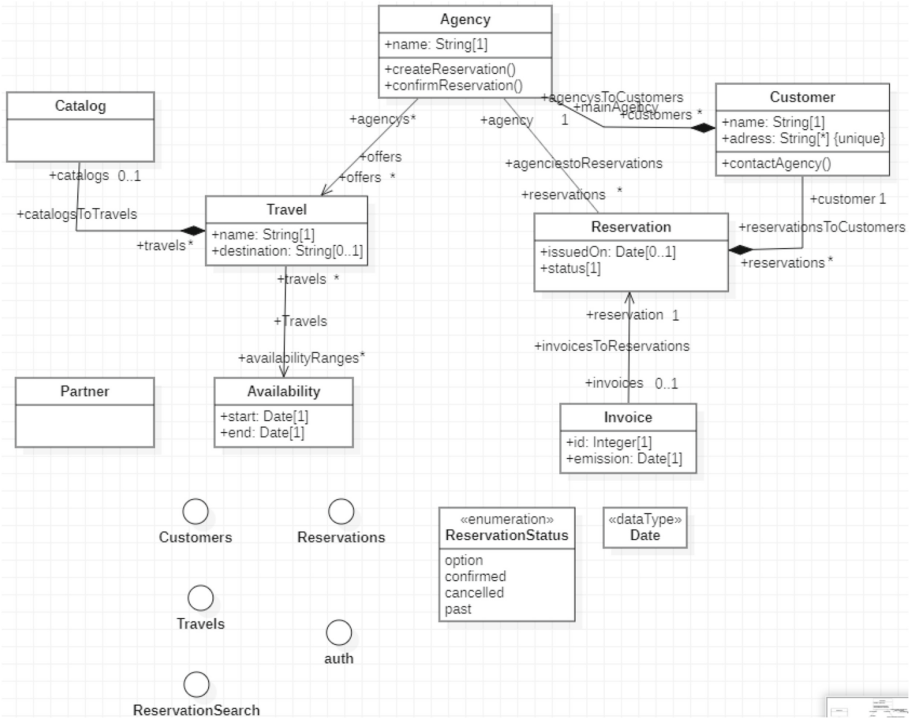


Fig. 5. Travel agency class diagram in StarUML.

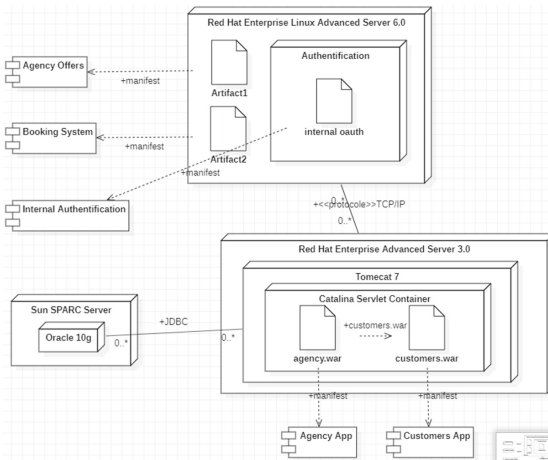


Fig. 6. Travel agency deployment diagram in StarUML.

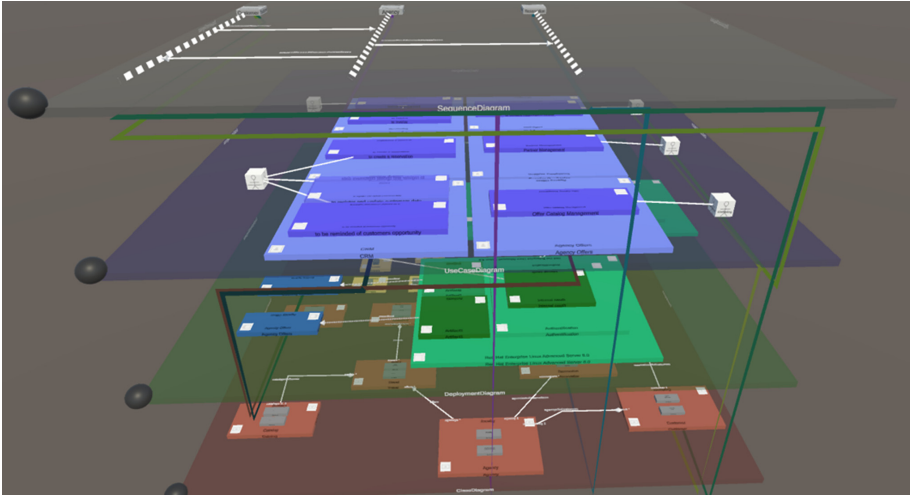


Fig. 7. VR-UML stacked hyperplane visualization of travel agency model.

A virtual tablet is provided in VR-UML to support interaction and modeling and to provide detailed information about an element. We chose this method since tablet usage is common and intuitive (less VR training needed), and other VR-based affordances are not yet standardized for providing detailed context-specific information for an element. Figure 8 shows the ability to add a new class to a diagram including a keyboard where each key is picked via a virtual laser pointer. Figure 9 shows the interface for creating a new relation and indicating the type of relation (e.g., association, aggregation, etc.) and its multiplicity. Figure 10 shows the ability to edit class attributes, e.g., the type, multiplicity, and visibility.



Fig. 8. VR-UML create class modeling support with virtual tablet and virtual keyboard.

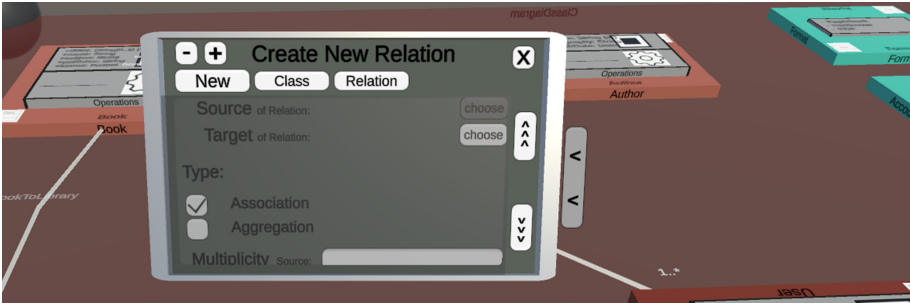


Fig. 9. VR-UML create relation modeling support with virtual tablet.



Fig. 10. VR-UML attribute modeling support via virtual tablet.

As exemplified in Fig. 11, visual clutter can be reduced via the anchor sphere affordance to collapse (hide) a diagram (which then displays the hidden diagram type).

Figure 12 shows side-by-side and offset diagram placement via anchor spheres.

Figure 13 shows the VR-MF hypermodeling capability for heterogeneous models in VR (e.g., here UML and ArchiMate); related elements can be annotated across models to support analysis.

5 Evaluation

To assess VR-UML empirically, a convenience sample of seven computer science students from sophomore through master students participated, despite the currently very restrictive COVID-19 pandemic situation and university contact policies. While the group is not large enough to be statistically significant, the results can provide insights to inform and guide future research. The subjects used an HTC Vive room scale VR set with a head-mounted display and two wireless handheld controllers tracked by two base stations. Each subject worked individually with a supervisor who provided instructions and timed the tasks. A Likert five-point scale was used for range-based responses. All had some familiarity with UML and had used Sparx Systems Enterprise Architect before; only two had used StarUML, and all but one had used VR.

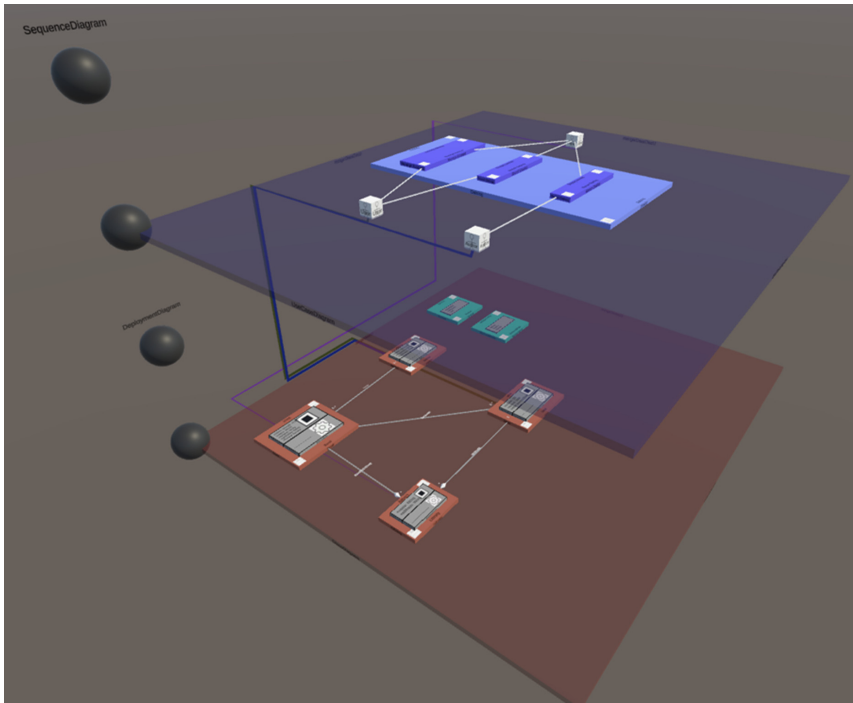


Fig. 11. VR-UML stacked plane view with two hidden/collapsed diagrams.

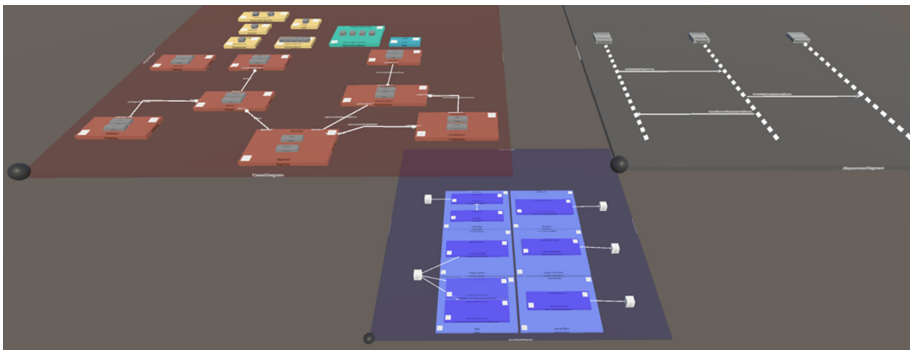


Fig. 12. VR-UML side-by-side and offset diagram placement.

The hypotheses that guided our tasks and questions were: while VR-UML will likely be less efficient than 2D modeling in general, (1) VR-UML is advantageous and efficient for more complex and multi-diagram models; and (2) users will subjectively enjoy the VR immersion experience in UML models more than the 2D models.

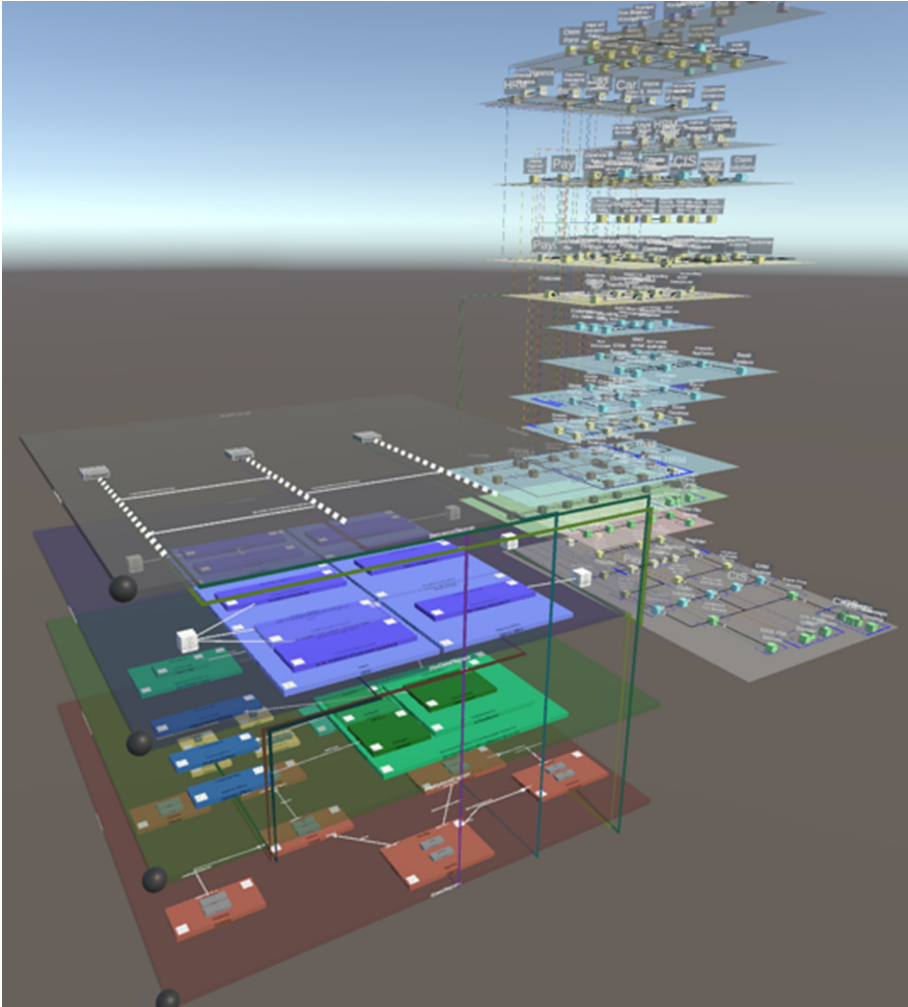


Fig. 13. Hypermodeling example showing a VR-UML and VR-EA ArchiMate model.

5.1 Quantitative Analysis

The subjects were timed for the following tasks in non-VR (using StarUML) and VR-UML:

1. Multi-diagram elements: which elements with the same name recur in multiple diagrams and how often?
2. Change the attribute “email” in the Customer class from public to private.
3. Change the relation multiplicity between Customer to Shopping Cart to 1-1.
4. Create the Model-View-Controller (MVC) pattern in the class diagram in non-VR (see Fig. 14) and VR-UML (Fig. 15). In non-VR a paper copy was accessible as a

reference, in VR they had to remember or verbally ask questions while wearing the headset.

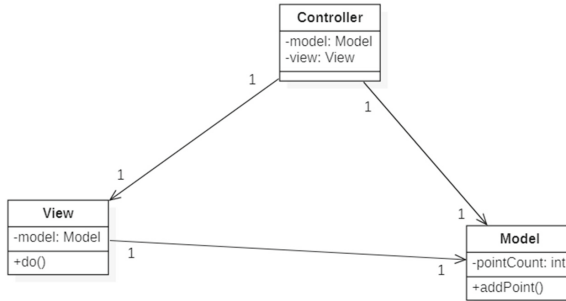


Fig. 14. MVC pattern task example in StarUML.

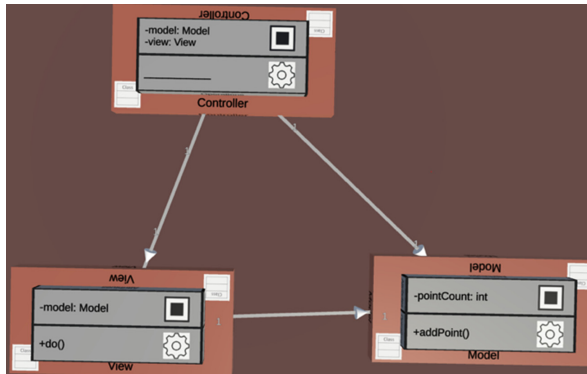


Fig. 15. MVC pattern task example in VR-UML.

Figure 16 shows the task duration results. On average, VR took 344% longer for Task 1, 141% longer for Task 2, and 43% longer for Task 4. For Task 3, when dealing with multi-diagram elements, VR was 14% more efficient on average - because VR-UML’s ability to visualize multiple diagrams and highlight inter-diagram elements. We see this result as providing support for hypothesis (1). Note that for Task 2, 3, and 4, the ranges show a large degree of overlap, which can be interpreted that VR can perform better than non-VR to depending on the user’s UML and VR competency.

Three possible reasons for the longer VR results are: 1) the VR interface is more cumbersome to control for modeling vs. a 2D mouse-based interface with which the subjects have been trained, 2) text entry via virtual laser pointer keypad selection (resembling one finger typing using a laser pointer) instead of the non-VR physical keyboard (enabling touch typing), and 3) time spent in VR navigating through 3D space to see or interact with the object of interest (vs. in 2D moving the mouse on a screen). As VR keyboards become commonplace, this could reduce this factor’s efficiency influence.

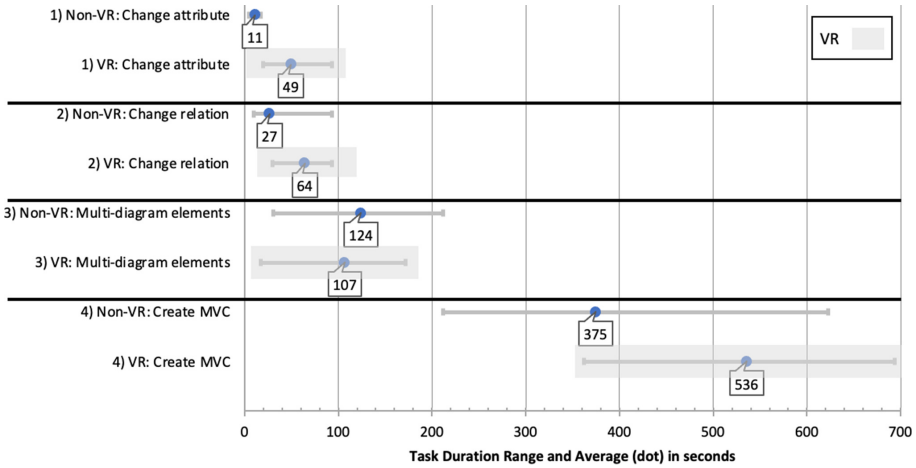


Fig. 16. VR and Non-VR task duration range and average (blue dot) in seconds. (Color figure online)

Table 1. UML familiarity (1 to 5, 5 = very familiar, 1 = unfamiliar) vs. error frequency.

UML familiarity	Non-VR errors	VR errors	Total errors
2	1	3	4
3	0	0	0
3	2	0	2
4	0	2	2
4	0	0	0
4	0	2	2
4	0	5	5

Table 1 shows the errors made. We note that in two cases no errors were made in either mode, in one case fewer errors were made in VR, while in four cases errors increased in VR. Since VR relied on subjects' memory of the pattern and subjects could not compare their model to a reference model on paper as they did in non-VR, we do not weight VR errors strongly. Due to the relatively minor error rate differences, we interpret the results to indicate that with additional training and familiarity with VR-UML, the error rate in VR could be equivalent to that of 2D and that it is not inherently more error prone for all cases and all subjects.

5.2 Qualitative Analysis

In the qualitative responses shown in Fig. 17, all agreed VR-UML to be intuitive and 43% more so than non-VR. 86% agreed that VR-UML provided a clear model structure.

As to changing an element, 71% found them equivalent while 29% found non-VR easier. For finding recurring elements across diagrams, 86% strongly agreed that it was easy in VR-UML compared to 43% for non-VR. In general, VR did not fare worse than non-VR on these qualitative aspects, and often even better.

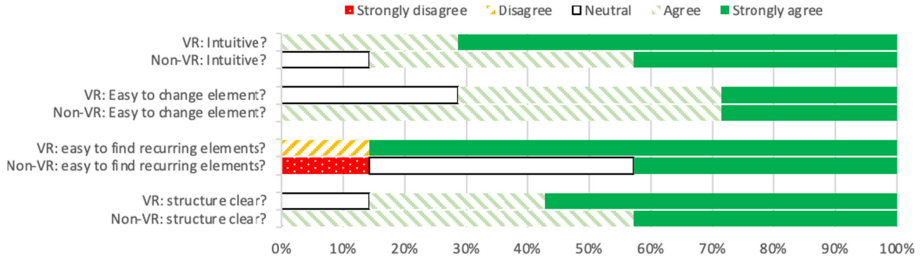


Fig. 17. Qualitative comparison of VR and Non-VR.

Additionally, 71% stated that they liked VR-UML better than non-VR. We interpret this as support for hypothesis (2). VR-UML advantages explicitly mentioned included: VR provided a better overview of diagrams and how they relate to each other, the layered 3D hyperplane stack makes comprehension of architecture easier, better visualization in general - and specifically for relations, VR is more intuitive, the VR user interface simpler than a menu system, and VR provides better focus due to immersion. Disadvantages mentioned included: efficiency to perform tasks, text input takes much longer via virtual keyboard with laser pointer, and the potential for VR sickness for sensitive users. Suggested improvements included: voice input or other text input alternative for VR.

5.3 Discussion

While our small sample size is not statistically significant, we believe there is still sufficient value from the results to infer trends and to inform future research. The study showed evidence that VR can indeed support modeling for certain scenarios. We hypothesized that VR would be advantageous relative to 2D for more complex structures or inter-diagram scenarios which VR can better depict simultaneously due to its 3D nature. As shown in Fig. 16, recurring elements across multiple diagrams were indeed found more quickly (16% on average), due primarily to our VR-based support for visually depicting these same elements, supporting hypothesis (1).

Factors that affected our study included: the COVID-19 policies to reduce interactions and interaction time, such that no preparation, training, or warm-up was given (no VR training nor VR-UML app training). In contrast, all participants had used 2D UML tools beforehand. Furthermore, VR app interaction and controls are not yet standardized and familiar, so subjects may not automatically know how to achieve some goal in VR – compared with professional 2D tools where common expectations exists as to where one will likely find menu items to achieve some task. Another aspect is cognitive stimulus: VR visualization takes up much more visual processing that is still relatively new and unfamiliar as yet to these subjects and can be disconcerting or initially affect efficiency (i.e., a new world to explore effect).

Threats to validity include: the small convenience sample size; the self-assessed UML competency (vs. a UML competency test); lack of experienced software developer UML competency or certification; VR tasks were performed directly rather than after a VR warm-up phase; users lacked a MVC reference image in VR (non-VR had a paper copy), thus subjects had to recall the MVC pattern from memory - which some may be better at than others - or verbally ask questions; lack of prior training with the VR-UML app, leading to inefficiencies and errors that may not actually depend on VR as a medium, but are caused by unfamiliarity with such an app and its interface (due to COVID-19 the evaluation time was minimized and training time cancelled).

As to counter-scenarios, VR-UML is likely not suitable or recommended for small and simple UML models or single-diagram models from an efficiency or effectiveness perspective. However, despite this, VR-UML could provide qualitative improvements which could possibly create (or rejuvenate) excitement for UML modeling.

In summary, we see various positive indicators from this study that VR-UML can show advantages where more complex and multi-diagram models are involved (and by inference hypermodeling); that the immersive experience of UML models in VR adds qualitative aspects that users prefer; and that any task inefficiencies in VR are probably tolerable (as shown by the task duration range overlap). VR-UML efficiency could be improved with explicit VR-UML training and text entry alternatives.

6 Conclusion

With our VR-UML contribution we have provided an immersive UML model experience for visually depicting and navigating UML diagrams of software models in VR. The solution concept and guiding principles were described, and its feasibility demonstrated with a VR prototype, with which we empirically evaluated our solution. Based on our VR hyperplane principle, it enhances UML diagrams with 3D depth, color, and automatically generated inter-diagram element followers based on our backplane concept. Modeling and interaction are supported via a virtual tablet and virtual keyboard. By leveraging the unlimited space in VR, the overall extent of multiple diagrams and large models can be depicted and navigated visually, while overall interrelationships within and between heterogeneous elements, diagrams, and models can be indicated and analyzed. Furthermore, our VR modeling framework VR-MF contributes a generalized hypermodeling approach for loading and visualizing different model types in VR whereby UML and EA-related models such as ArchiMate, BPMN, and Atlas can be visualized and analyzed simultaneously. The sensory immersion of VR can support task focus during model comprehension and increase modeling enjoyment, while limiting the visual distractions that typical 2D display surroundings incur. Most subjects preferred VR-UML overall.

Various UML tools support reverse-engineering models directly from the code (Ozkaya 2019). By leveraging today's processors and cloud computing, they can rapidly provide just-in-time reverse-engineered models to document and visually convey the real software model based on the actual codebase both efficiently and without model-to-code inconsistencies. In combination with VR-UML, visualization, analysis, and immersion in software models could rejuvenate UML-based software modeling in the face of

rapidly evolving codebases and in support of software maintenance of legacy systems. Future work includes adding support for additional UML diagram and elements types, enhancing the VR interface, adding additional inter-model annotation and informational capabilities, optimizing the model storage format, and a comprehensive empirical study.

Acknowledgements. The authors would like to thank Marie Baehre and Stefan Wehrenberg for their assistance with the implementation and evaluation.

References

1. Brooks Jr., F.P.: *The Mythical Man-Month*. Addison-Wesley Longman Publication Co., Inc., Boston (1995)
2. Kemerer, C.F., Paulk, M.C.: The impact of design and code reviews on software quality: an empirical study based on PSP data. *IEEE Trans. Softw Eng* **35**(4), 534–550 (2009). <https://doi.org/10.1109/TSE.2009.27>
3. OMG: Unified modeling language version 2.5.1 (2019)
4. OMG: XML Metadata Interchange (XMI) Specification Version 2.5.1 (2015)
5. Ware, C., Franck, G.: Viewing a graph in a virtual reality display is three times as good as a 2D diagram. In: *Proceedings of 1994 IEEE Symposium on Visual Languages*, pp. 182–183. IEEE (1994). <https://doi.org/10.1109/VL.1994.363621>
6. McIntosh, P.: X3D-UML: user-centered design, implementation and evaluation of 3D UML using X3D. Ph.D. dissertation, RMIT University (2009)
7. Krolovitsch, A., Nilsson, L.: 3D Visualization for Model Comprehension: A Case Study Conducted at Ericsson AB. University of Gothenburg, Sweden (2009)
8. Rodrigues, C.S.C., Werner, C.M., Landau, L.: VisAr3D: an innovative 3D visualization of UML models. In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 451–460. IEEE (2016)
9. Steuer, J.: Defining virtual reality: dimensions determining telepresence. *J. Commun.* **42**(4), 73–93 (1992). <https://doi.org/10.1111/j.1460-2466.1992.tb00812.x>
10. Allocoat, D., von Mühlenen, A.: Learning in virtual reality: effects on performance, emotion and engagement. *Res Learn Technol* **26** (2018). <https://doi.org/10.25304/rlt.v26.2140>
11. Oberhauser, R., Lecon, C.: Virtual reality flythrough of program code structures. In: *Proceedings of the Virtual Reality International Conference-Laval Virtual 2017*, pp. 1–4. ACM (2017). <https://doi.org/10.1145/3110292.3110303>
12. Oberhauser, R., Pogolski, C., Matic, A.: VR-BPMN: visualizing BPMN models in virtual reality. In: Shishkov, B. (ed.) *BMSD 2018. LNBIP*, vol. 319, pp. 83–97. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94214-8_6
13. OMG: Business Process Model and Notation (BPMN) Version 2.0.2 (2014)
14. Oberhauser, R., Pogolski, C.: VR-EA: virtual reality visualization of enterprise architecture models with ArchiMate and BPMN. In: Shishkov, B. (ed.) *BMSD 2019. LNBIP*, vol. 356, pp. 170–187. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24854-3_11
15. Open Group: ArchiMate 3.1 Specification. The Open Group (2019)
16. Oberhauser, R., Sousa, P., Michel, F.: VR-EAT: visualization of enterprise architecture tool diagrams in virtual reality. In: Shishkov, B. (ed.) *BMSD 2020. LNBIP*, vol. 391, pp. 221–239. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52306-0_14
17. Sousa, P., Leal, R., Sampaio, A.: Atlas: the enterprise cartography tool. In: *18th Enterprise Engineering Working Conference Forum*, vol. 2229. CEUR-WS.org (2018)

18. Zhang, B., Chen, Y.S.: Enhancing UML conceptual modeling through the use of virtual reality. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, p. 11b. IEEE (2005). <https://doi.org/10.1109/HICSS.2005.239>
19. Ozkaya, M.: Are the UML modelling tools powerful enough for practitioners? A literature review. *IET Softw.* **13**, 338–354 (2019). <https://doi.org/10.1049/iet-sen.2018.5409>
20. Ozkaya, M., Erata, F.: A survey on the practical use of UML for different software architecture viewpoints. *Inf. Softw. Technol.* **121**, 106275 (2020). <https://doi.org/10.1016/j.infsof.2020.106275>. ISSN 0950-5849
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. 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, pp. 1–2. ACM (2018). <https://doi.org/10.1145/3281505.3281602>
23. Vincur, J., Navrat, P., Polasek, I.: VR city: software analysis in virtual reality environment. In: 2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp. 509–516. IEEE (2017). <https://doi.org/10.1109/QRS-C.2017.88>
24. Schreiber, A., Brüggemann, M.: Interactive visualization of software components with virtual reality headsets. In: 2017 IEEE Working Conference on Software Visualization (VISSOFT), pp. 119–123. IEEE (2017). <https://doi.org/10.1109/VISSOFT.2017.20>
25. Bork, D., Karagiannis, D., Pittl, B.: A survey of modeling language specification techniques. *Inf. Syst.* **87**, 101425 (2020). <https://doi.org/10.1016/j.is.2019.101425>
26. Coad, P., Lefebvre, E., De Luca, J.: Java Modeling in Color with UML: Enterprise Components and Process. Prentice Hall (1999) ISBN 0-13-011510-X
27. OMG: UML Diagram Interchange (UMLDI) 1.0 (2006)
28. UML Designer (2021). <http://www.uml designer.org>
29. Kruchten, P.: Architectural blueprints - the “4+1” view model of software architecture. *IEEE Softw.* **12**(6), 42–50 (1995). <https://doi.ieeecomputersociety.org/10.1109/52.469759>