A Role Based Model Template for Specifying Virtual Reality Software

Sai Anirudh Karre
Software Engineering Research Center
IIIT Hyderabad, India
saianirudh.karri@research.iiit.ac.in

Vivek Pareek
Software Engineering Research Center
IIIT Hyderabad, India
vivek.pareek@research.iiit.ac.in

Raghav Mittal
Software Engineering Research Center
IIIT Hyderabad, India
raghav.mittal@research.iiit.ac.in

Y. Raghu Reddy
Software Engineering Research Center
IIIT Hyderabad, India
raghu.reddy@iiit.ac.in

ABSTRACT

Research in hardware and software support for Virtual Reality (VR) has significantly increased over the last decade. Given the software platform fragmentation and hardware volatility, there is an apparent disconnect among practitioners while building applications in the VR domain. This paper proposes a role-based model template as a meta-model to specify the bare minimum VR software system. We conducted a grounded-theory-based qualitative study on prevailing and phased-out VR SDKs and standards to propose this meta-model. This model template can help VR practitioners build open-source tools to develop, design, and test VR software systems.

CCS CONCEPTS

• **Software and its engineering** → **Software system models**;

KEYWORDS

Virtual Reality; VR SDK; VR Model Template; Meta model; Grounded-theory

ACM Reference Format:
Sai Anirudh Karre, Vivek Pareek, Raghav Mittal, and Y. Raghu Reddy. 2022. A Role Based Model Template for Specifying Virtual Reality Software. In *37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022* . Rochester, MI, USA, 5 pages. https://doi.org/10.1145/3551349.3560514

1 MOTIVATION

A Virtual Reality (VR) software system strives to induce a targeted behavior in an organism (predominantly a human or an animal) using artificial sensory simulation [15]. VR hardware and software have evolved independently, causing disparity in the overall evolution of VR as a domain. Consequently, it has become difficult for VR Practitioners' to build portable and cross-platform VR products.

ASE '22, October 10–14,2022, Rochester,MI, USA

ACM ISBN 978-1-4503-9475-8/22/10…$15.00
https://doi.org/10.1145/3551349.3560514

Several attempts are made to standardize VR as a domain through standards like VRML (Virtual Reality Modeling Language) by W3C - 1997 [23], COLLADA (COLLAborative Design Activity) by Sony - 2004 [11], O3D by Google - 2010 [10] etc. These standards have failed due to a lack of interoperability support of VR software with VR Hardware [7]. Recently attempts were made to support interoperability between VR software and hardware using OpenXR standard APIs [8]. However, these new standards are still in draft and require more comprehensive usage/feedback within the VR practitioner community. Various commercial VR providers proposed and practiced variants of the VR conceptual model. These models are open-ended and do not permit interoperability to meet the necessities of a bare-minimum VR software system. At a minimum, a VR software system should depict the components like scene, scene-objects, camera, action-responses, and behavior outcomes. However, the existing conceptual models do not provide common control and data flow to facilitate constructing a bare-minimum VR software system. In this paper, we address the following research questions:

> ***RQ:*** *What constitutes a meta-model of a bare minimum VR software system?*

The paper presents the study setup, data analysis, and our theory on creating a role-based model template as a meta-model for VR software systems to avoid platform fragmentation.

2 METHODOLOGY

Conceptualizing a meta-model for VR systems requires a thorough understanding of VR as a domain in addition to understanding VR systems in application domains like health care, banking, etc. We made an unsuccessful attempt by conducting a systematic literature review on understanding what constitutes a bare-minimum VR software system. The results were obsolete and no longer significant to comprehend contemporary VR software systems. Most of the primary and secondary studies [16] [17] suggested superficial information on the working of a typical VR software system. We found studies that explain VR as a software system applied in various fields like education, tourism, simulation, healthcare, and design applications with no underlying information about using the constructs of a bare-minimum VR software system. Given the limited academic literature, we adopted the Socio-Technical

Grounded Theory (STGT) approach [9] to investigate components of bare-minimum VR software systems. It is an iterative and incremental research method using available resources with abductive reasoning for theory development.

2.1 Data Collection

We relied on the following resources to gather the required data to establish the theory for constructing a meta-model for VR software systems.

Informal Interviews We conducted informal interviews with developer communities of UNITY Technologies, Epic Games, Khronos Group (OpenXR), and the VR/AR Association (UK/APAC) for nearly four months (Oct-2021 to Jan-2022). The interviews aimed to understand practitioners' perspectives of a bare-minimum VR software system in practice. Participants with a minimum of five years of VR development experience were considered for interviews. In total, 39 VR practitioners participated. The following questions were the basis for the informal interviews.

- What do you consider elements of a meta-model for VR software system?
- Do current VR Development tools explain elements of a bare-minimum VR software system?
- Did you build any supporting tools for VR? If yes, How did they gather meta-information about a bare-minimum VR System?

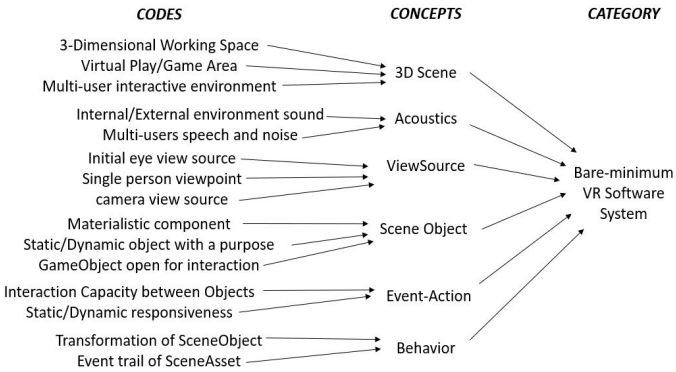
VR SDKs Selection: Our interactions with VR practitioners led us to review widely used VR SDKs like UNITY3D [5], Unreal Game Engine [6], CryEngine [2], AframeJS [3], and Amazon Lumberyard [4]. We examined SDK source code, underlying classes, and white papers. All other non-technical proprietary materials are excluded from examination.

VR Standards Selection: Interactions with VR practitioners led us to explore VR standards in detail. We considered prior standards - VRML[23], X3D [25], WebXR, O3D [10], and prevailing standards - OpenXR [8] and IEEE VR/AR Working Group for our study. Additionally, we examined peer-reviewed publications to understand the components of a meta-model for a bare-minimum VR software system.

2.2 Data Analysis

We used the Open Coding method [19] to annotate the interview transcripts, VR Standard documentation, and VR SDKs documentation with essential details. Our annotation criteria are to check for the presence of elements that explain the constructs of a bare-minimum VR Software system and depict the control and data flow among them. These annotations are linked with generalized ***codes***, which have the same meaning as our annotation criteria. These codes are further generalized into common labeled ***concepts***. The researcher has the flexibility to decide the concept labels. The concepts are ordered into a ***category*** goal for our study. Fig 1 explains a few examples of open codes linking overall concepts and categories to provide an overall understanding of a bare-minimum VR software system.

Figure 1: Common Codes from Interview Study



2.3 Observations

Based on the codes generated using open-coding, we used *Abduction Reasoning* [9] to theorize a meta-model for a bare minimum VR software system. Abductive reasoning helps researchers conduct data analysis through different means such as hunches, clues, metaphors or analogy, symptoms, patterns, and explanations. This approach opens various avenues for creative thinking and theory development. Following are the data points and observations captured to depict bare-minimum concepts that are found to constitute a VR Software System. These concepts are the building blocks of a meta-model understanding of a VR software system.

- ***Scene*** - A 3D environment or space with (un)limited dimensions in terms of length, breadth, and height with elements operating together as a whole in their respective part. It is referred to as "play area" in VR SDKs and "virtual operating space" in most VR standards.
- ***Article*** - A 3D object with specific dimensions and physical properties, including material type, texture, color. It has other properties like Pixel/Voxel type, CanCastShadow, IsRigitObject, IsCollidiable, CanRotate, Isluminous, etc. These objects are engaged as static objects or interactable with the character in a given scene.
- ***Action*** - Any engagement between two or more articles leads to interaction, causing a known/unknown outcome. The engagement may be internal and external to objects within the prescribed scene.
- ***Audio*** - This element is associated with the scene and article. A scene may or may not have background audio. An article may or may not give rise to audio internally or externally due to an action by another article within the prescribed scene.
- ***Behavior*** - The Action's outcome and the object's transformation within the prescribed scene.
- ***Viewsource*** - An initial viewpoint of a person trying to experience the scene. It is called a *camera* in VR SDKs and *viewpoint* in VR standards.

The culmination of these concepts varies between various VR standards and VR SDKs based on their levels of abstraction and nomenclature. However, the overall elements illustrated in Section 2.3 are the bare-minimum set to construct a VR software system. To understand how SDKs perceive the same underlying meta-model elements differently, we present examples from WebXR and AFrameJS-based VR Scenes. A WebXR scene[1] presents a no-audio 3D scene with a centered view-source, cube articles placed at a particular height with no action but a changes behavior in terms of color when clicked on the cube. On another end, an AframeJS scene [2] presents a no-audio 3D scene with a centered view-source, with unsized articles placed at a distance with no action but changes behavior in terms of their dimension with any external intervention. These two scenes are different and are built using different VR SDKs. These two SDKs work under different meta-model concepts and code templates. The WebXR-based scene is obsolete as browsers no longer support it. The AframeJS based scene is supported by javascript-based browsers only (chrome, firefox etc.). These two scenes are built for the web and are not compatible with high-end head-mounted-device consumption. One of the SDKs, WebXR, is now deprecated, thus limiting the portability of all WebXR scenes and causing platform fragmentation. Such gaps can be avoided if they are built using a shared underlying meta-model of VR. The following section presents a role-based model template representing a shared meta-model for a VR software system.

3 VR SOFTWARE SYSTEM META MODEL

We present the meta-model of a VR software system using a role-based meta-modeling language (RBML) approach proposed Dae-Kyoo et al. [13]. It is a UML-based language extension that supports rigorous specification of patterns that characterize a family of design models. Since RBML uses UML syntax, UML tools are used to create RBML specifications. Figure 2 presents the role-based model template class diagram of a bare-minimum VR software system. Template elements are marked with the "|" symbol. As shown in Figure 2, *|Scene*, *|Viewsource*, *|Time*, *|Behavior*, *|Physics*, *|Requestor*, *|Article*, *|Audio*, and *|Action* are called as class templates. Each class template consists of two sections, the attribute template and the association template. For example, the *|Audio* class template has *|sourcetype*, *|noise*, *|init*, *|inext* as part of attribute template section and *|runsound*, *|syncsound*, and *|asyncsound* are listed as part of association template section. Each association template is defined with a cardinality *[1..*]*, i.e., one to many with *|param** as unlimited parameters. Each role model template class is linked with UML based relationship specifications with association definitions like *|acesses*, *|Impartswith*, *|RendersInto*, *|Syncwith* and *|Validateswith*. The virtual software system is invoked by a *|Requestor* class template. *|Audio*, *|Action* and *|Scene* class templates are initiated synchronously to load underlying *|Article* (s) and their *|Behavior* (s) through a *|ViewSource* onto *|Scene*. *|ViewSource* initiates all other class templates asynchronously with *|Time*.

Implications of VR Meta-Model: Studies have shown that enterprise VR product development differs from traditional software development [12]. Contemporary VR practitioners are primarily from the gaming industry. They are adopting development methods that fit their needs to succeed in their business. Thus the VR practitioners are left with few monopolistic tools and frameworks. A VR Meta-model will provide a conceptual overview of VR as a domain and help practitioners create novel open-source tools to support and ease VR product development. The following are a few of many use-cases that can be formalized by ideating generic approaches to ease VR product development.

- *Requirements* - This area is a challenging aspect for VR. The meta-model will help practitioners build generic open-source tools to elicit, specify and track requirements in detail across the VR product development. These requirements can be programmatically validated across different stages of VR product development.
- *Code Analysis* - There is almost no tool support for VR source code analysis. Code patterns play a great role in addressing unstructured code in large-scale VR scenes. A meta-model can help build generic open-source tools to refactor, modularize the code-base, define code patterns, and suggest code-reusability.
- *Testing* - Unit testing and cognitive walkthroughs are currently practiced to test the VR scenes. Quality and Usability guidelines are manually evaluated through these studies. A meta-model can help build generic automated test-case generation, test strategies, guideline-based code-validation, and evaluation code-metrics specific to VR.
- *Release* - Traditional in-place release management is currently practiced in VR. A meta-model can aid dev-ops practitioners in automating the packaging of various components of VR with novel release strategies. Generic release management can address platform fragmentation to a certain extent.
- *Physics Engines* - Most widely used prevailing physics engines are either customized or proprietary. A meta-model can be used to develop a novel physics engine with and beyond the natural laws of physics. Different variants of VR engines can be developed with varying degrees of complexity.

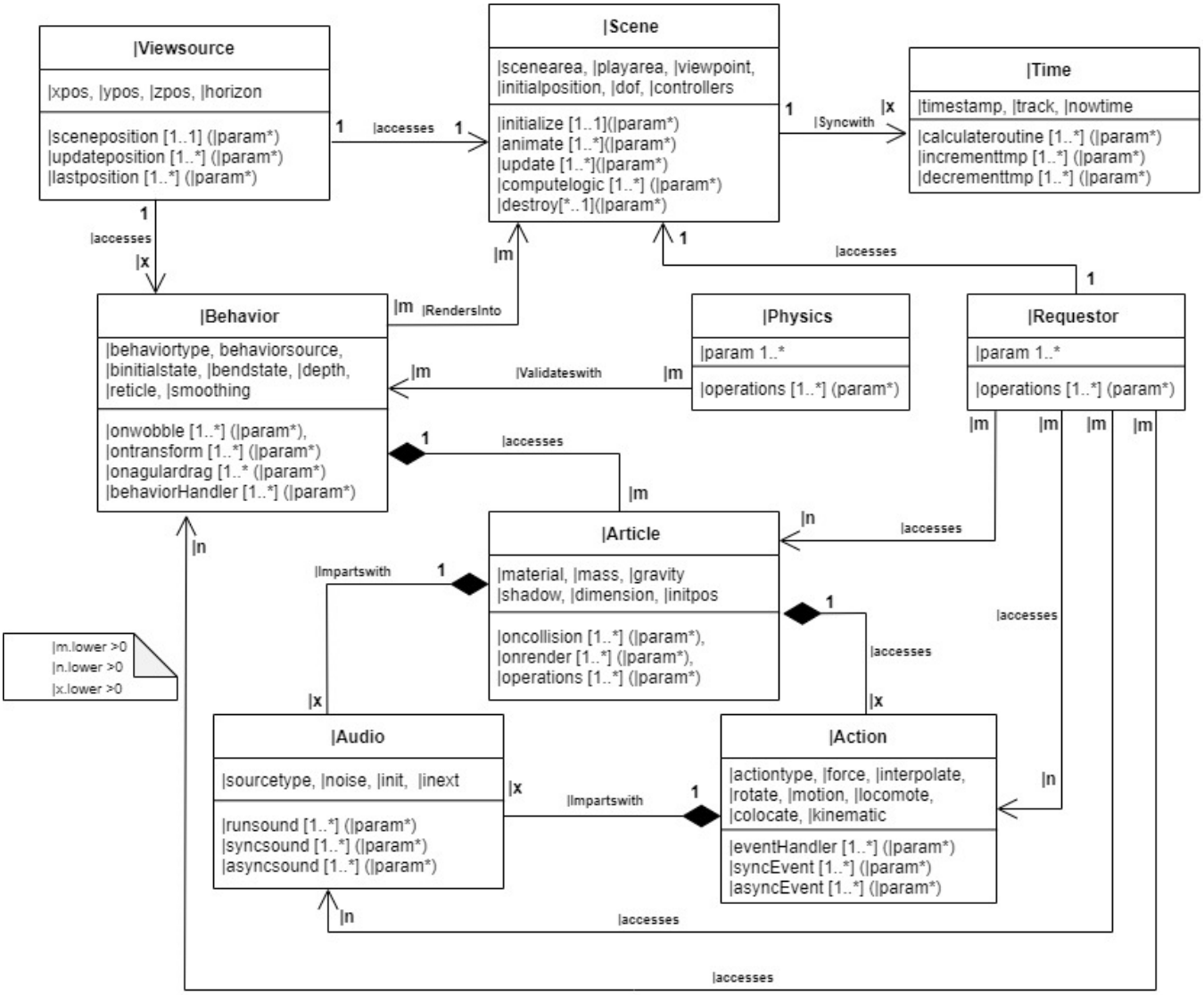
Meta-Model Instance - We present a meta-model instance example by providing example bindings for a sample VR football game[3]. Table 1 provides example bindings of a test case to be used for verifying the action:kick and response audio:splash associated with the article:ball. Example bindings can be used as part of a generic VR test-case generator tool. This table provides a correlation between meta-model parameters and test-case generator application-specific elements of a VR football game instance. Using the test-case application specification, a custom test-case generator tool can be developed to generate test cases for a game like VR applications.

4 RELATED WORK

Tanriverdi et al. presented an early conceptual model of Virtual Environments [20]. Different VR practitioners introduced various design-centered conceptual models to illustrate VR as a domain. Ossa [18], CLEVR [14], VR-WISE Virtools Dev [21], Marigold toolset [24], UsiXML [22] are few vital models. However, none

[1] https://immersive-web.github.io/webxr-samples/input-selection.html?usePolyfill=0
[2] https://aframe.io/aframe/examples/animation/warps/
[3] https://github.com/Utopiah/aframe-soccer

Figure 2: Role based Model Template of a bare minimum VR Software System



of these models illustrate the meta-model information, and they are either obsolete or no longer relevant for contemporary VR. Most of these models are design-centered and interface-centered in usage. Later, these models helped practitioners to extend them into standards like VRML [23], X3D [25], and AC3D [1] specifications for the Web. With the rise in hardware and software capabilities, VR offerings are dominated by head-mounted devices rather than traditional Web. Consequently, these models and standards failed to evade platform fragmentation, causing monopolies in VR offerings as their meta-model levels are dissimilar. OpenXR [8], the latest cross-platform specification for VR on the Web, created a new directive for VR offerings. However, they failed to present a matured conceptual model to illustrate a bare-minimum VR software system that could help VR practitioners build new tools to ease VR product development.

5 CONCLUSION

This paper presents a role-based model template as a meta-model for a bare-minimum VR software system. We constructed this meta-model theory using abductive reasoning of codes generated using open coding of informal interview transcripts, VR SDKs documentation, and VR standards documentation. The presented meta-model will help VR practitioners to develop generic open-source tools to ease VR product development and address platform fragmentation between VR hardware and software. As part of our future work, we planned to develop an overall state machine using the

Meta-model parameter	Application Specific element
|Action	Kick
|Audio	Splash
|Article	Ball
|Action::|kinematic	Kick::kinematic
|Action::|motion	Kick::motion
|Action::|syncEvent	Kick::syncEvent
|Audio::|init	Splash::init
|Audio::|syncsound	Splash::syncsound
|Impartswith	impartswith
|accesses	accesses
|x	*
1	1
|m	*

Table 1: Example bindings for a test-case of kicking the ball using template class

proposed meta-model to understand the states involved in a bare-minimum VR system comprehensively. We also planned to develop open-source tools for requirement specification and testing of VR applications.

ACKNOWLEDGMENTS

The authors thank the VR practitioners from SAP-XR, Deloitte Digital Labs, ThoughtWorks, Samsung Studios, UNITY Dev Group, and Khronos Dev Community for participating in the study and sharing their insights.

REFERENCES

[1] 1994. *AC3D.* Inivis Inc. http://www.inivis.com/features.html
[2] Dec, 2021. *CRYENGINE Programming Documentation.* Crytek Technologies. https://docs.cryengine.com/display/CEPROG/CRYENGINE+Programming
[3] Jan, 2022. *AframeJS Documentation.* https://aframe.io/docs/1.3.0/introduction/
[4] Jan, 2022. *Lumberyard Documentation.* Amazon Web Services. https://docs.aws.amazon.com/lumberyard/latest/userguide/lumberyard-intro.html
[5] Jan, 2022. *Unity3D Manual - Offline Documentation.* UNITY Technologies. https://docs.unity3d.com/Manual/OfflineDocumentation.html
[6] Jan, 2022. *UnRealEngine 5 Documentation.* EPIC Games Inc. https://docs.unrealengine.com/5.0/en-US/
[7] Matthew Brennesholtz. 2017. *VR/AR Standards – Are We Confused Yet?* https://www.displaydaily.com/article/display-daily/vr-ar-standards-are-we-confused-yet
[8] Khronos Group. 2019. *The OpenXR Specification 1.0.24.* https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html
[9] Rashina Hoda. 2021. Socio-Technical Grounded Theory for Software Engineering. *IEEE Transactions on Software Engineering* (2021), 1–1. https://doi.org/10.1109/TSE.2021.3106280
[10] Google Inc. 2010. *COLLADA - Digital Asset and FX Exchange Schema.* https://code.google.com/archive/p/o3d/
[11] Sony Computer Entertainment Inc. 2004. *COLLADA - Digital Asset and FX Exchange Schema.* Khronos Group. https://www.khronos.org/collada/
[12] Sai Anirudh Karre, Neeraj Mathur, and Y. Raghu Reddy. 2019. Is Virtual Reality Product Development different?: An Empirical Study on VR Product Development Practices. In *Proceedings of the 12th Innovations on Software Engineering Conference (formerly known as India Software Engineering Conference), ISEC 2019, Pune, India, February 14-16, 2019.* ACM, 3:1–3:11. https://doi.org/10.1145/3299771.3299772
[13] Dae-Kyoo Kim, Robert B. France, Sudipto Ghosh, and Eunjee Song. 2003. A Role-Based Metamodeling Approach to Specifying Design Patterns. In *27th International Computer Software and Applications Conference (COMPSAC 2003): Design and Assessment of Trustworthy Software-Based Systems, 3-6 November 2003, Dallas, TX, USA, Proceedings.* IEEE Computer Society, 452. https://doi.org/10.1109/CMPSAC.2003.1245379
[14] G. Jounghyun Kim, Kyo Chul Kang, Hyejung Kim, and Jiyoung Lee. 1998. Software Engineering of Virtual Worlds. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (Taipei, Taiwan) *(VRST '98).* Association for Computing Machinery, New York, NY, USA, 131–138. https://doi.org/10.1145/293701.293718
[15] Steven M. LaValle. 2020. *Virtual Reality.* Cambridge University Press. http://vr.cs.uiuc.edu/vrbookbig.pdf
[16] Joseph R. Levy and Harley Bjelland. 1994. *Create Your Own Virtual Reality System.* McGraw-Hill, Inc., USA.
[17] William R. Sherman and Alan B. Craig. 2003. Introduction to Virtual Reality Systems. In *Understanding Virtual Reality*, William R. Sherman and Alan B. Craig (Eds.). Morgan Kaufmann, San Francisco, 70–73. https://doi.org/10.1016/B978-155860353-0/50003-3
[18] Finnegan Southey and James G. Linders. 2001. Ossa - A Conceptual Modelling System for Virtual Realities. In *Conceptual Structures: Broadening the Base*, Harry S. Delugach and Gerd Stumme (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 333–345.
[19] Anselm Strauss and Juliet Corbin. 1967. Discovery of grounded theory.
[20] Vildan Tanriverdi and Robert J.K. Jacob. 2001. VRID: A Design Model and Methodology for Developing Virtual Reality Interfaces. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (Baniff, Alberta, Canada) *(VRST '01).* Association for Computing Machinery, New York, NY, USA, 175–182. https://doi.org/10.1145/505008.505042
[21] Olga Troyer, Frederic Kleinermann, Bram Pellens, and Wesley Bille. 2007. Conceptual Modeling for Virtual Reality., Vol. 83. 3–18.
[22] Jean Vanderdonckt, Quentin Limbourg, Benjamin Michotte, Laurent Bouillon, Daniela Trevisan, and Murielle Florins. 2004. USIXML: a User Interface Description Language for Specifying Multimodal User Interfaces. *W3C Workshop on Multimodal Interaction* (01 2004).
[23] W3C. 1997. *VRML Virtual Reality Modeling Language.* Web3D Consortium. https://www.w3.org/MarkUp/VRML/
[24] JAMES S. WILLANS and MICHAEL D. HARRISON. 2001. A toolset supported approach for designing and testing virtual environment interaction techniques. *International Journal of Human-Computer Studies* 55, 2 (2001), 145–165. https://doi.org/10.1006/ijhc.2001.0474
[25] X3D. [n.d.]. *Extensible 3D.* Web3D Consortium. https://www.web3d.org/x3d/what-x3d/