VReqST: A Requirement Specification Tool for Virtual Reality Software Products

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

Amogha A Halhalli
Software Engineering Research Center
IIIT Hyderabad, India
amogha.halhalli@students.iiit.ac.in

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

***Abstract*—Developing Virtual Reality (VR) software products with discrete and incremental requirements is a challenging task for VR practitioners. A domain expert's assistance plays a key role in VR product completeness, as most VR requirements are abstract or under-specified during the early stages. Slight changes to the requirements can significantly impact the overall flow of the VR software development process. In this paper, we introduce VReqST, a tool for VR requirement analysts to specify requirements for building effective VR software products. It is developed based on a Role-based model template for specifying virtual environments, custom behaviors, VR-specific algorithms, user-flows, action responses, timeline of events, etc. The tool has been developed after several interactions with VR practitioners from industry & academia. We share our insights on the effectiveness of this tool in practice.**

***Index Terms*—Software Requirement Specification, Requirement Elicitation, Virtual Reality, Industrial Practices**

I. Motivation

Virtual Reality (VR) software product development requires a certain set of practices that are different compared to traditional software product development [1]. This is primarily due to the involvement of various VR-specific stakeholders' like Virtual Environment (VE) designers, VR Developers, Acoustic Engineers, Interaction Animators, Usability Practitioners, VE Testers, Maintenance Engineers, etc. who are not used to traditional software development practices. Any miscommunication between stakeholders or the slightest deviation from the prescribed requirements can lead to significant rework in the VR software. Precise and complete requirement specifications help with the reduction in rework. While the observations hold for traditional software products also, it needs to be emphasized more for VR products. The following reasons motivated us to develop a tool to record precise requirement specifications to support VR product development.

- **Platform Fragmentation** - Proprietary SDKs do not support portability and cross-platform support. A VR code published for a particular head-mounted device (HMD) in a given SDK will not work when used in a different HMD. The developers must re-publish the same VR code for different HMDs [2].
- **Poor Tool Support** - VR Developer tools contain proprietary naming conventions. There is no common understanding of VR as a domain. Thus, most VR tools rely on their unique VR model information. VR community is fixating on a few tools to avoid confusion [3].
- **Multi-modality** - VR is multi-modal [4] and requires meticulous requirement gathering as they aim to achieve realism. There is a need for comprehensive requirements to achieve realism in VR.
- **Risky Assumptions** - Obscure and abstract prototype designs can spike the development cost of a VR product. A VR developer cannot presume the VR scene with certainty. Unclear and hypothetical use-case stories about the VR scenes, articles involved in the scene, action responses between the articles, specific behaviors, and events' timelines can lead to misleading outcomes.

Studies show that effective requirement engineering practices can yield better software quality [5]. In this paper, we introduce a novel model-template-based requirement specification tool called VReqST [6] that can help VR requirement analysts (RA) specify precise requirements. The rest of the paper is structured as follows: section II presents the model template that forms the basis for the tools, section III details the workflow and discusses the tool in action, section V and VI provides some related work and outlines future work respectively.

II. Background - VR Meta Model

In our previous work [7], we had specified the bare-minimum meta-model of a VR Software System using a role-based model template to illustrate the conceptual understanding of VR Technology as a domain. To specify requirements that are pertinent to VR software products, the RA is required to comprehensively understand the minimal set of elements of the VR software product. Thus we developed VReqST - a requirement specification tool for specifying VR software products on the basis of the role-based model template. The elements that form the building blocks of a minimal VR Software system are as follows:

- **Scene** - A 3D environment with (un)limited dimensions in terms of length, breadth, and height called a virtual play area with objects operating together as a whole in their respective parts. These are the dimensions of a virtual play area in a real-world physical space.
- **Article** - A 3D object with specific dimensions and physical properties, including material type, texture, color, etc.

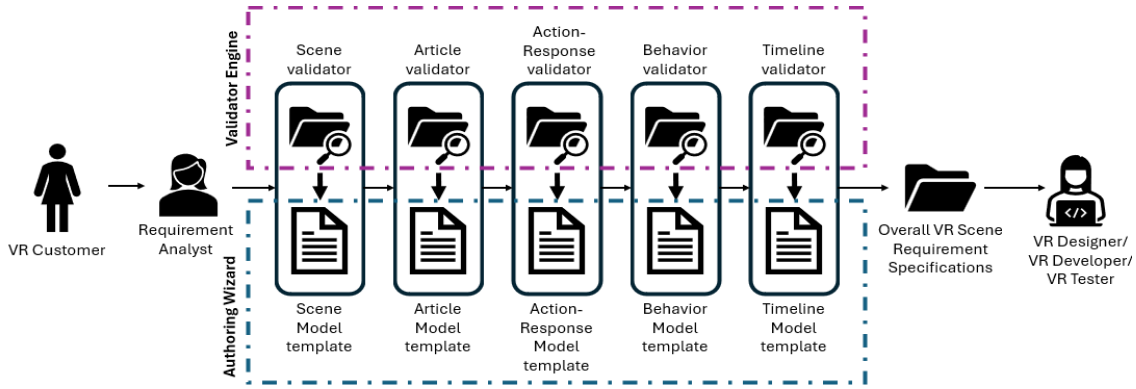

Fig. 1. VReqST - VR Requirement Specification pipeline

- **Action-Responses** - Any engagement between two or more articles leads to interaction, causing a known/unknown outcome.
- **Behavior** - It is about the outcome of an action over an article and its transformation within the given scene.
- **Audio** - This element is associated with a scene or an article.
- **ViewSource** - It is an initial viewpoint of a person trying to experience the VR scene.
- **Timeline** - Execute action responses between articles based on synchronous and asynchronous consequences within a prescribed time frame.

III. About the Tool[1]

The first iteration of VReqST was a formal-specification language with pre-defined VR domain-specific code constructs. After the feedback from VR requirement analysts (RAs) from the VR community, we transformed it into a tool with a fillable model template with underlying validators. Fig III illustrates the overall VReqST requirement specification pipeline for VR products. An RA captures requirements from the customer and instead of authoring them in plain English, uses VReqST model template files - Scene, Article, Action-Response, Behavior, and Timeline to publish the overall specification for VR designers, VR Developers and VR testing team to consume.

A. Model Template and its Validator

Model templates are extensions of model concepts defined as part of the role-based model template for VR Software Systems [7] in section II. These model concepts are scene, article, action-response, behavior, and timeline. Each model concept constitutes of model attributes that act as properties of the respective model concept that are represented in JSON (JavaScript Object Notation) format. For example, the model template file of the model concept called **"Article"** contains one of the model attributes called ***"_clippingplane"***, an imaginary plane that determines how much of a scene a camera can see in the viewport. Only objects between the camera's near and far clipping planes are rendered in the camera's view. The near clipping plane ***near_clip*** is the closest point to the camera that the camera can see, and the far clipping plane ***far_cp*** is the furthest point. The scope of these attributes *near_clip* & *far_cp* is in meters i.e. if ***near_clip*=0.01** and ***far_cp*=1000** then the clipping plane is set with 0.01m for near clipping plane and 1000m for far clipping plane. All such validation rules for model attributes for a given model template are defined as part of the underlying **Model Validator**. Each model template file (scene, article, action-response, and timeline) has a distinct model validator file to validate the datatype and scope of the attribute. It behaves like an underlying grammar file. The details are made available as part of our documentation [6].

Customizing Model Template-Validator: The model template files are open for customization, i.e. the RA can extend the existing model template by introducing new attributes without altering the bare-minimum attributes. However, the validation rules for the new attributes are to be included as part of its underlying validator file. For example, if a new attribute called **"IsMobile"** is added as part of **"Article"** model template, it is required to define the scope and datatype of the attribute as part of the article's model validator. Here IsMobile is defined as a physics property in the article that can execute motion when an external force is applied. The scope of this attribute is boolean, i.e., it can only hold either value **'1'** or **'0'** where **'1'** signifies that the respective article holds mobility physics property, whereas '0' signifies that the respective article does not hold mobility physics property. Steps to include such customizations are made available as part of our documentation [6].

B. VReqST Overview

VReqST is a web-based tool built using the MERN (Mongo-ExpressJS-ReactJS-NodeJS) technology stack, i.e., a ReactJS framework is for client-end web page rendering, ExpressJS framework and NodeJS for server-end setup, and MongoDB as a database to store the data. The source

[1] **VReqST Online:** https://vreqst.github.io/vreqst-deploy/

Fig. 2. VR Bowling Alley Game in a developer mode (UNITY Engine) - Point-of-View 1 & 2

code[2], tool demo, documentation, and sample requirement specification examples are available as part of our resources [6]. Authoring Wizard and Validator Engine are two parts of VReqST. The RAs are only exposed to the authoring wizard, whereas the validator engine is executed in the background. The authoring wizard contains five stages where a given stage is prerequisite of the next step, i.e. the RAs are required to fill the model template of Scene as the first part of the first step, followed by articles, action-responses along with state, their behavior(s) and place all these attributes on a timeline. The model attributes of each step are carried forward into the next step so that they can be reused for authoring required specifications.

Behavior Editor Validation: The behavior editor provides RAs to author conditional logic using inbuilt code constructs such as IF, IF-ELSE, SWITCH, FOR, WHILE, DO and DO-WHILE. The syntax of these code constructs does not follow any specific programming language, but follows a simplified version as prescribed by our documentation [6]. These behaviors are business rules with an expected output for the corresponding input. These business rules are based on use cases and vary from product to product. For example, to author **"Behavior"** - assume that we are building a tic-tac-toe game in VR with two articles, Circle (O) and Cross (X). If three of any one of these articles form a straight line (vertical, horizontal, diagonal) in a tic-tac-toe game, the player with the respective article is deemed the winner. The following is the sample specification for authoring this behavior.

1: **CASE #** *Circle* OR *Cross* = true **#**
2: **C1:** forms straight vertical line = *player*:*winner*;
3: **C2**: forms straight horizontal line = *player*:*winner*;
4: **C3:** forms straight diagonal line = *player*:*winner*;
5: **C4:** *player*: *next_step*;
6: **!**

The demo[3] is also made available as part of resources [6].

C. VReqST Features

Following are the features that VReqST provides RAs to embrace based on the sophistication of their VR software product:

- VReqST provides RA the ability to author on bare minimum model template attributes using the core model validator. It also provides an ability to customize the model templates by introducing new attributes and defining their validation rule under the custom model validator.
- RA can share the custom model templates and its corresponding custom model validator with other RAs. For example, if a new attribute called **"color"** (inspired by OpenXR standard specification [2] - **XrColor4f** structure) is included *scene model template*, whose validation rule for variables *(r, g, b, a)* with datatype as *(float, float, float, float)* and *non-nullable* should be included as part of *scene model validator*.
- Manage VR product specifications as a project and allow versioning of specifications over time to review, discuss, and revise with the respective VR designer, VR developer, or VR testing teams marking this into a *VR requirements management tool*.

IV. Validation

We conducted our validation in 4 stages. The details are illustrated below:

- **Stage 1:** We used VReqST to specify requirements for two pilot VR applications to understand the capabilities and shortcomings. First, we specified detailed requirements for a VR bowling alley game as shown in Fig 2 and later developed it as a full-end game. This helped us revise the model template to extend the bare minimum attributes. Second, a VR Virtual art gallery was developed that uses a limitless locomotion algorithm called PragPal [8]. This helped us understand the capabilities to author algorithms using VReqST and we extended our behavior model template from our learnings. We developed these VR Scenes using the UNITY Game Engine Version - 2022.2.16 works on the Oculus Meta Quest 2 head-mounted device. The resultant specifications along with the source code of these two VR scenes are available as part of our resources [6].
- **Stage 2:** We used VReqST as a teaching tool for Graduate students to specify requirements for VR Scenes [9]. We conducted this experiment as a tool-based collaborative assignment for assessing the correctness of software products using a Virtual Reality (VR) application as part of a CS300-level course at our University. The students authored VR specifications of more than 25 VR scenes as

[2]**Source Code:** https://github.com/Amogha027/VReqST-2
[3]**Tool Demo:** https://youtu.be/XW7C5luZCo4

novice requirement analysts. They shared their feedback on the usability of the tool and used it as our learning to improve the control flow of the VReqST.

- **Stage 3:** We elicit and specify requirements for multiple mood-based scenes (positive/negative/neutral) for the detection of depression using a gait analysis technique. VR is used as an intervention whereas the gait was captured using an external sensing mat [10]. The overall scenes were developed by VR practitioners using the VReqST-based specifications and conducted a detailed field study [10].
- **Stage 4:** We reached out to the VR Community to use VReqST in practice and share their feedback in 3 iterations. We reached them via Linkedin, UNITY Unite Conference, XR Discord Channels, and XR Community Connect. More than 50 RAs have taken part in our study and shared feedback. We revised and published the latest version of VReqST by incorporating their feedback. The details of our study are available as part of our resources [6].

V. Related Work

Early studies in requirement specification in VR are either human-centered or process-oriented. Kim et al. [11] was the first to explore the possibilities of creating a process-oriented approach for requirement engineering in VR. Early tools like STATEMENT [11] are backed by POSTECH, and Message Sequence Diagrams (MSD) paved the path for automation in VR requirement engineering. Pellens et al. proposed VR-WISE [12] - for modeling the static part of VR using the conceptional specification to code generation, Seo et al. proposed a CLEVR [13] model that considers the functions, behavior, hierarchical modeling, user task and interaction modeling, and composition reuse in early VR Systems. Alinne et al. proposed a scene-graph-based approach for requirement specification and testing automation of VR products [14]. These tools are not generalized for contemporary VR and are partially obsolete. We developed VReqST to address this gap using a model template-based approach that is extendable and customizable.

VI. Future Work

The current version of VReqST requires RA or a VR developer to author specifications by filling in the model templates. As part of future work, we are planning to take the following directions and extend this tool as a foundation for model-driven development for VR software products. We are extending the VReqST to consider plain English prompt text as input and use domain-specific Retrieval-Augmented generation-based LLM to populate the model template and validate ondemand to avoid overhead to RA. This approach will improve the current way of requirement specification and also bring in knowledge from various application domains like health care, aviation, etc. to build effective VR products. Further, we also started working on building a VReqST specification-based Design Authoring tool to generate design templates and support design versioning specific to VR Engines like UNITY and UnReal. This will provide an ability for VR designers to pre-populate mock scenes using a structured VR specification generated using VReqST. Additionally, tools like code generation, test-case generation, and release management can be developed as a model-driven development pipeline to pave new horizons for future VR product development.

Acknowledgements

The authors thank the VR practitioners from UNITY, Deloitte Digital, Khronos Group, Reality Lab, Microsoft for providing their constant feedback on improving VReqST.

References

[1] S. A. Karre, M. Neeraj, and Y. R. Reddy, "Is virtual reality product development different? an empirical study on vr product development practices," in *Proceedings of the 12th Innovations on Software Engineering Conference (ISEC)*. Pune, India: ACM, 2019.

[2] K. Group, *The OpenXR Specification 1.0.24*, 2019. [Online]. Available: https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html

[3] M. Nebeling and M. Speicher, "The trouble with augmented reality/virtual reality authoring tools," in *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, 2018, pp. 333–337.

[4] D. Martin, S. Malpica, D. Gutierrez, B. Masia, and A. Serrano, "Multimodality in vr: A survey," *ACM Comput. Surv.*, vol. 54, no. 10s, sep 2022.

[5] O. V. S. Filho and K. G. Kochan, "The importance of requirements engineering for software quality," in *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics: Information Systems Development-Volume I - Volume I*, ser. ISAS-SCI '01. IIIS, 2001, pp. 529–532.

[6] S. A. Karre, "Vreqst - resources and documentation," Feb. 2023. [Online]. Available: https://saianirudh-karri.gitbook.io/vreqst/

[7] S. A. Karre, V. Pareek, R. Mittal, and R. Reddy, "A role based model template for specifying virtual reality software," in *37th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE22. New York, NY, USA: Association for Computing Machinery, 2023.

[8] R. Mittal, S. A. Karre, Y. P. K. Gururaj, and Y. R. Reddy, "Enhancing configurable limitless paths in virtual reality environments," in *15th Innovations in Software Engineering Conference*, ser. ISEC 2022. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3511430.3511452

[9] S. A. Karre, K. Vaidhyanathan, and Y. Raghu Reddy, "A tool based experiment to teach elicitation and specification of virtual reality product requirements," in *Proceedings of the ACM Conference on Global Computing Education Vol 2*, ser. CompEd 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 195. [Online]. Available: https://doi.org/10.1145/3617650.3624936

[10] A. Souza, F. Nunes, and M. Delamaro, "An automated functional testing approach for virtual reality applications," *Software Testing, Verification and Reliability*, vol. 28, 10 2018.

[11] G. J. Kim, K. C. Kang, H. Kim, and J. Lee, "Software engineering of virtual worlds," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, ser. VRST '98. New York, NY, USA: Association for Computing Machinery, 1998, pp. 131–138.

[12] O. D. T. B. Pellens, W. Bille and F. Kleinermann, "Vr-wise: A conceptual modeling approach for virtual environments," in *Proceedings of the methods and Tools for Virtual Reality Workshop (MeTo-VR)*, 2005, pp. 1–10.

[13] J. Seo and G. J. Kim, "Design for presence: A structured approach to virtual reality system design," in *Teleoperators Virtual Environ.*, 2002, pp. 378–403.

[14] M. W. Wani, "Development of cost effective gait mat for assessment of depression," *Masters Thesis, IIIT Hyderabad, India*, vol. IIIT/TH/2024/44, p. 89, April 2024.