Requirements Engineering Practices for Developing Enterprise Virtual Reality Software Products

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Engineering

by

Sai Anirudh Karre
20162153
saianirudh.karri@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500032, INDIA
June, 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this PhD thesis, titled ***"Requirements Engineering Practices for Developing Enterprise Virtual Reality Software Products"*** by *Sai Anirudh Karre*, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Advisor: Dr. Y. Raghu Reddy

To *my Parents*

Late Karre Srinivas Rao *M.A(PPM), PGDBM*

Karre Uma Devi *M.Com, PGDECE*

Acknowledgments

My journey at IIIT-Hyderabad has been a wonderful experience. It could not have been so without the support of many people. Firstly, I thank my advisor, Dr.Y. Raghu Reddy, for accepting me as a student and constantly guiding me during the MS Research program and eventually into the Ph.D. program. His guidance has helped me improve both as a researcher and a person. His help and support during difficult times strengthened and motivated me to move further. I also thank the faculty of the Software Engineering Research Center: Dr. Venkatesh Chopella, Dr. Raman Saxena, Dr. Karthik Vaidhyanathan and Prof. Viswanath Kasturi for creating a research-driven ecosystem at IIIT Hyderabad and providing great feedback on my research. I thank my fellow research mates at Software Engineering Research Center: Lalit Mohan Sanagavarapu, Mritunjay Kumar, Aman Banerjee, Abhinav Gupta, Raghav Mittal, Y. Pawan Kumar Gururaj, Mohammad Waqas Wani, Vivek Pareek, Vivek Iyer, Dhiraj Shanmukha Mitra V, Abhay Kumar and Amogha Halhalli.

In particular, I am grateful to Prof. Kamalakar Karlapalem and Late Prof. Kesav Nori for sowing seeds of my early research ability and creating enthusiasm and tremendous motivation to pursue research at IIIT Hyderabad. I sincerely thank my manager(s) from my workplace who supported me with a hassle-free research journey: Mr. Ryan Persichilli - Sr. Director, Accurate Background (USA); Mr. Nihil Kantameni - VP (Success), India, Mr. Khaled Itani - VP(Technology), USA , Mr. Craig Fearon - VP(Products), Canada from SumTotal Systems Inc. I sincerely thank my workplace mentor, Mr. Rajesh Kedarisetti, Product Owner at BlueCross BlueShield of Tennessee (USA), for motivating me throughout my journey.

I also thank my reviewers Prof. Sudipto Ghosh - Colorado State University, USA; Prof. Prasad Onkar - IIT Hyderabad and Prof. Suresh Purini - IIIT Hyderabad for providing valuable feedback and helping me shape the overall thesis better.

I could not have accomplished it without the support and understanding of my parents. I wish to thank my parents, Uma Devi Karre, Late Srinivas Rao Karre, and my sister Sharath Deepika Karre for being my constant support and motivation. I am also thankful to the IIIT Ph.D. Student Community, IIIT Academic Office, and fellow students for giving me an inspiring environment and many opportunities to grow.

Abstract

Virtual Reality (VR) is an emerging technology that is gaining traction across various domains in the past decade. Mainstream adoption is still work in progress, especially in enterprise domains like health care, manufacturing, construction, Defence, National Security, etc. Building VR technology has its own set of challenges. Challenges like volatile hardware, low computation capacity, trademark monopolies, hyper-enthusiastic user base, lack of evident domain expertise, and lack of structured processes within the developer ecosystem, have been encountered by the VR Enterprise and Consumer community over the years. These challenges require thorough investigation to understand the state-of-the-art approaches and formulate better solutions to optimally utilize VR technology.

To comprehend the development practices adopted while building VR software products, we conducted an year-long multi-level exploratory study with the VR Enterprise developer community worldwide. As part of this empirical study, we observed a wide range of strategies, methodologies, approaches, and models embraced by the VR practitioners while building new-age VR software products at various stages of VR product development (including requirements gathering, VR product design, software toolkits, VR product quality, VR product release, and reuse). The major observations from the empirical study being: (1) conventional software engineering practices are force-fitted and practiced to manage overall VR products (2) modified conventional approaches that deal with specific software problems are embraced as alternate solutions to VR problems (3) most of the VR Enterprise community have their antecedents in gaming industry (4) VR Enterprise suffers from hardware and software cross-platform support and portability issues (5) development cost of the overall VR product will spike if development practices are altered during VR product development and (6), requirements engineering, software quality (specifically usability), security, and lack of open-source software tools are the thrust areas for immediate advancement. Research support is needed to improve and introduce VR enterprise-centered approaches to produce high quality VR software products and to reduce development costs.

After studying the development practices of the VR Enterprise, we looked at relevant literature. We reviewed academic research on Requirements Engineering methods, Software Quality, and Software Usability methods for VR product development. Academic methods focus on conventional software development. They do not consider the realities of VR Enterprise practices. Literature shows poor requirements engineering practices negatively impact VR product development. Rigorous, enterprise-oriented requirement methods could help mitigate VR software quality issues.

The primary motive of this thesis is to discuss the need for rigorous requirements engineering for VR product development. We present a novel tool-based approach to specifying requirements for VR products using a conceptual understanding of VR as a domain. We formulate a "**Role-based model template**" for a VR software system that provides a meta-model understanding of VR as a domain. The model template of the VR software system illustrates the bare minimum concepts and the underlying attributes required to build a minimalistic VR software product. The model template can be customized by the VR development community without altering its bare minimum concepts and attributes. This allows mechanisms to ease overall VR development and overcome being tied to specific VR hardware and software development (s).

Further, we developed **VReqST** - *a requirement specification tool* for VR requirement analysts to formally elicit, analyze, and specify requirements based on the role-based model template. VReqST can be used by VR designers to eventually design VR scene templates with higher precision. VR developers can avoid imagination stress by relying on these design templates. VReqST-based requirement specifications enable traceability, reusability, and incorporate quality requirements in VR product development. To understand VReqST's practical influence, we had the VR Enterprise adopt and provide feedback on it. Their systematic, iterative feedback helped improve the tool with developer-friendly features and enhancements. As proof-of-concept, we also used VReqST to specify requirements for a real-time VR depression detection application.

As a future work, we envision expanding VReqST's capabilities to serve as an input for Generative AI tools, aiding in the various stages of development of VR software and facilitating model-driven development approaches for VR products.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The concept of an immersive, computer-generated environment was first explored in the 1960s through research in fields like computer graphics, human-computer interaction, and simulation. Science fiction movies, novels, and video games featuring virtual worlds and immersive experiences captured the public's imagination and fueled interest in VR technology in the early 1990s. VR practitioners from academia and industry have developed many applications to illustrate how VR can be used to complex tasks and ease sophistication. However, compared to web-based and mobile applications, VR as a technology is yet to gain broad based acceptance. Head-mounted devices (HMDs) seem to be still restricted to niche domains/applications. As shown in Figure 1.1, The Emerging Technologies Hype Cycle, published in the 2017 Gartner Report, outlines five distinct phases in the life-cycle of a technology: Technology Trigger, Peak of Inflated Expectations, Trough of Disillusionment, Slope of Enlightenment, and Plateau of Productivity. In the 2017 report, Virtual Reality (VR) technology was positioned on the Slope of Enlightenment, indicating that the technology was beginning to mature, and its potential benefits were becoming more evident as it moved towards mainstream adoption. Since 2018, VR has consistently been plotted on or above the Plateau of Productivity. In other words, the report suggests that VR technology has reached its full potential and has become an established part of the business landscape. During the Slope of Enlightenment phase, more practical applications and use cases of the technology for enterprises started to emerge and become better understood. Second- and third-generation VR products were released, and more realistic expectations were formed about the technology's capabilities, limitations, and potential applications. However, despite the reported maturity, widespread adoption of VR has proven more challenging than anticipated for various reasons. To comprehend the overall challenges hindering widespread adoption, there is a pressing need to examine the current state-of-the-art in VR technology, its capabilities, its developer community, and the practices employed by this community. This examination will provide insights into the underlying factors contributing to the adoption problems faced by the masses.

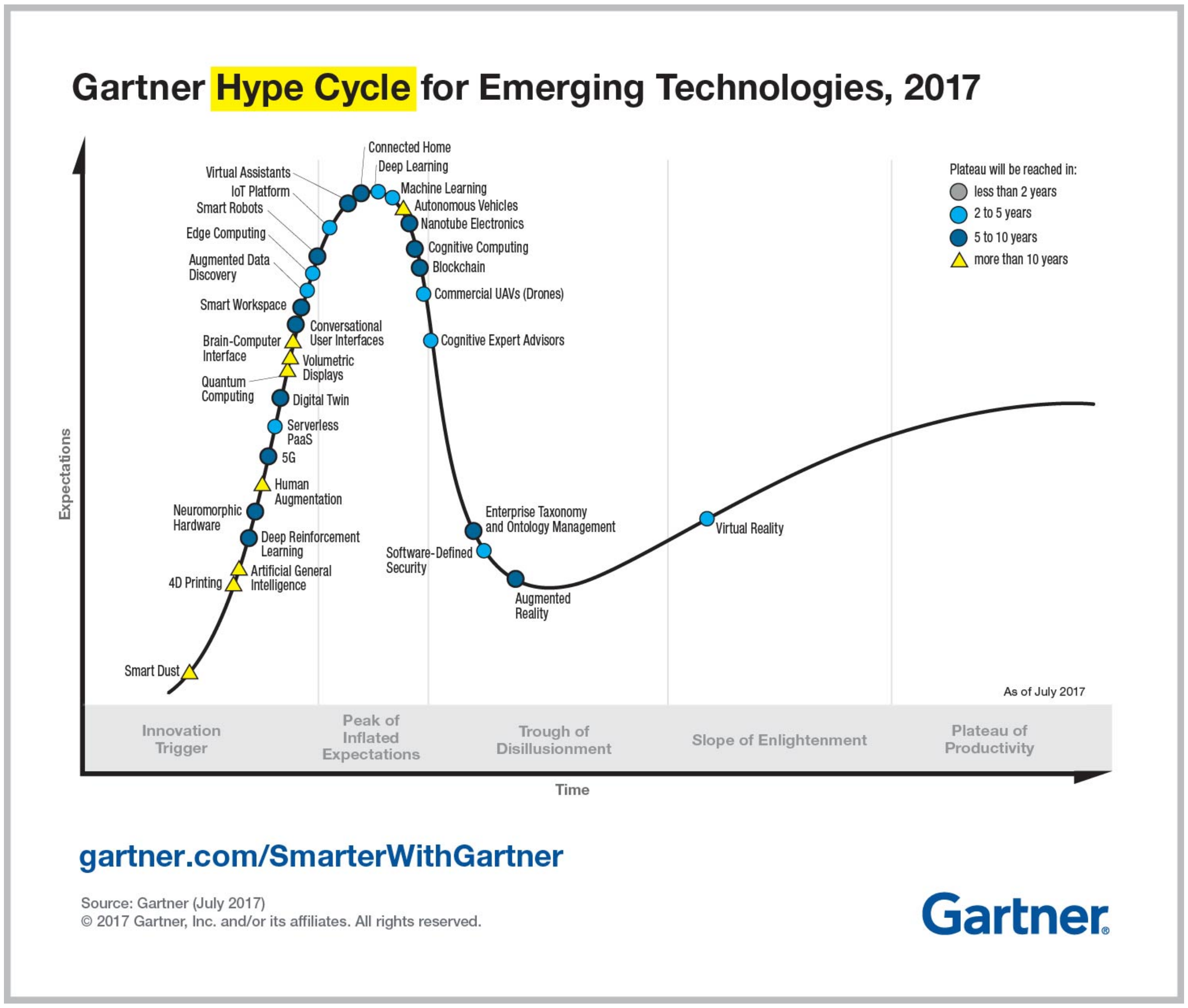


Figure 1.1 2017 Gartner Hype Cycle for Emerging Technologies

VR products can be categorized as (1) **Consumer VR** and (2) **Enterprise VR**. *"Consumer VR"* refers to virtual reality applications and experiences primarily intended for personal entertainment and leisure. These applications typically prioritize creating immersive gaming experiences, interactive storytelling, virtual tourism, and other forms of entertainment. The main focus in consumer VR is on developing visually stunning and highly interactive environments that can transport users to different virtual worlds. On the other hand, *"Enterprise VR"* refers to virtual reality applications specifically designed for professional and business purposes. These applications aim to enhance productivity, train employees, simulate real-world scenarios, and improve various industries such as healthcare, architecture, engineering, and manufacturing. Enterprise VR emphasizes providing realistic simulations, accurate representations of real-world environments, and practical training experiences. It is crucial to distinguish between Consumer VR and Enterprise VR because the objectives of these applications differ significantly. Consumer VR applications like gaming demand high-performance rendering on the Head-Mounted Display (HMD) to ensure smooth and immersive experiences. These applications require exceptionally high frame rates rendering to prevent motion sickness and provide a seamless virtual experience. On the other hand, VR applications designed for diagnosing Myopia or Glaucoma, for example, may not require the same level of performance on the HMD. These medical applications primarily focus on accurately representing visual stimuli and measuring specific eye conditions. While a high frame rate rendering may enhance the overall experience, it may not be as critical as in gaming applications. Recognizing the varying performance requirements in different VR applications is essential for optimizing resources and ensuring the best user experience. By understanding the specific needs of each application, developers can allocate resources accordingly.

Who are developing the VR products? - As we explore towards low adoption of enterprise VR products, we need to learn that conventional software practitioners have moved from ad-hoc processes to customized processes for developing software. Software practitioners have benefited from decades of empirical studies conducted by academic researchers, which have helped guide more reliable approaches to software product development. Empirical evidence suggests that standardized methods and tools strengthen the processes for Industrial practice [1]. Improvements to these methods and tools can advance product quality and help practitioners during product development. Given the rapid technological advances, clearly understanding the processes involved in executing disruptive and rapidly changing systems can impact product usage in the market [151]. Thus, **"Virtual Reality"** as technology can no longer be developed in conventional approaches and requires a VR-centered approach to improve product usage. Since 2005, VR-based Games have been considered VR software products due to their rich exposure among gamers [219]. However, this view does not apply to Enterprise VR products as they are more complex and domain-specific in execution. As traditional software products are now undergoing a refreshing change by transforming themselves into VR-based offerings, more software engineering practices are being adopted by VR practitioners. The primary goal of such a transformation in practices is to provide a cutting-edge experience to its consumers [30]. Enterprise Software providers are now investing in VR-based business solutions to reconstruct the traditional ideas for the new generation of consumers. As per Digi-Capital Market Research Report, [2], there has been $3 billion investment in the VR Enterprise product market for the next decade. It includes various applications in areas like Education, Health, Art/Design, HRTech, Services, Tourism, News, etc., to provide a personalized visual experience to respective end-users. Invariably, these products are being developed by software practitioners practicing traditional software engineering principles while building a routine software product [107] for the conventional market. Traditional Game developers and designers actively involved with VR-based games have created an ecosystem for software developers to enter into VR space [31]. Game developers are good at design and strategy planning. In the past two decades, they have adopted some software engineering principles and fine-tuned the practices based on their development needs [186]. They even adopted game-based development guidelines and a game-based development cycle [15] to improve their internal processes for an easy VR product release. With developers' amalgamation from gaming and software engineering into VR space, a new ecosystem has emerged [265]. By imparting traditional SE practices, core software developers are building enterprise products using game-based VR engines. Oculus, a leading VR technology firm, has opened developer support programs [3] to fill the talent gap for enterprise VR product development in this ecosystem.

These observations posits us further to explore the VR developer ecosystem, their practices, and approaches in detail. It may help us determine the gaps and instigate new proposals for building better enterprise VR products so that it can raise the adoption of VR to the masses. We constructed our problem statement around these early observations and motivation to investigate further. We illustrate the problem statement in detail in the following sub-section.

In contrast to VR Technology, other alternate reality-based technologies like Augmented Reality (AR), Mixed Reality (MR), and Extended Reality (XR) borrow concepts from VR in practice. Our focus is primarily on VR due to its unique properties and use cases. However, our work can still be extended or applied to other alternate realities in practice after thorough experimentation. We further detail the differences between these alternate realities along with the unique properties of VR in Chapter 2.

1.1.1 Problem Statement

In early 2017, we started exploring the area of VR through the lens of software engineering to understand the reasons behind the poor adoption of VR as a technology for developing enterprise VR applications to reach the masses. Over time, we synthesized the following problem statements to investigate the reasons in the following stages.

- **What is the journey of a typical VR product from development to delivery?** - At this preliminary stage, such an investigation will help us understand the overall processes, practices, tools, standards, and stakeholders involved during VR product development in an Enterprise setup. It requires closely examining VR practitioners who release VR products for a targeted customer base. It is worth investigating as it helps us identify the potential gaps in adopting VR as a technology for building enterprise products.

- **Is VR product development different from conventional software product development?** - In general, VR is yet another software. All aspects of conventional software engineering practices may apply to VR product development. However, we may not guarantee they will help to deliver a satisfactory, i.e., a quality VR product. Such an investigation will help understand how various areas of conventional product development, like requirement gathering, design, coding, testing, and release, are managed in practice. This review can provide us with a direction toward exploring each vertical to determine the areas of improvement and suggest a solution to ease VR product development.

- **What areas within the VR product development life-cycle require research support?** - Considering the outcome of examining the gaps in VR software development, exploring the areas of interest within VR product development requires additional research support. Here, the scope of research support is to determine new tools, methods, approaches, or metrics that can eventually ease VR product development.

These problem statements aim to understand the overall VR enterprise ecosystem to conduct more profound research into specific areas that can enable the VR enterprise to improve adoption issues in practice.

1.1.2 Research Journey

We started our research journey in January 2017, exploring areas of interest in researching Virtual Reality from a Software Engineering context. Various new articles and white papers by industry leaders plagued our early overview of virtual reality as a next-generation emerging technology with billions of worth in the marketplace. In the middle of 2017, we started contacting the VR Community (both VR Enterprises and VR Consumers) to understand the overall VR ecosystem. We observed VR got its facelift in early 2011 when Palmer Luckey, the founder of OculusVR (now merged into Meta Inc.), released a well-defined VR head-mounted device tethered to personal computers. Existing head-mounted displays in the market suffered from low contrast, high latency, low field-of-view, high cost, and extreme bulk and weight. OculusVR addressed all such gaps by introducing a variety of VR HMDs like Oculus Rift and Oculus Rift S with a focus on the gaming industry. It created a phase shift for VR in the consumer market. The Oculus HMDs started attracting significant interest from industry and professional spheres for productivity enhancement, visualization, and advertising. Various architecture, automotive, and manufacturing firms have shown great interest in visualizing their products in VR space. By end of 2017, the VR Enterprise products had risen significantly with a vast talent pool across the globe. Conversely, there is a severe crisis in the VR enterprise developer community as they began adopting approaches, methods, and various other engineering practices from conventional software engineering. Apart from the gaming industry, there was no widespread adoption across the masses due to a lack of enterprise products that suffice the intent of VR in practice. Thus, we started exploring the VR domain through the prism of Software Engineering and tried to understand their VR Community.

As part of our journey, we conducted a first-of-its-kind multi-year empirical study with the VR community to understand their strategies for developing VR software products. We recorded potential areas of research that require Software Engineering expertise to address the challenges faced by the VR community. Requirements, Design Versioning, Usability, Software Quality, Product Planning, and Release management are observed to be the areas within the VR product life cycle that require immediate research assistance and tool support. We developed a conceptual model for a VR system and a meta-model for building future tools for VR product development. We explored Requirements as a critical area of interest from a software engineering perspective, as effective requirement management can improve the overall quality of the VR product. We developed a requirement specification tool called VReqST to elicit, specify, analyze, and manage requirements for VR product development. We later evaluated the tool in practice and improvised it with the feedback from the VR community.

With the advent of the early large-language model in 2018 and their evolution towards generating detailed 3D models without extensive expertise in 3D modeling software like Blender, Maya, or 3ds Max. Recently, a 3D-LLM [115] was developed that considers 3D point clouds and their features as input to perform a diverse set of 3D-related tasks, including captioning, dense captioning, 3D question answering, task decomposition, 3D grounding, 3D-assisted dialog, navigation, and so on. Such could enhance such tools to consider VReqST specifications as input to generate instruction-driven 3D modeling and 3D design versioning. It can accelerate automated LLM-based 3D article designs using VReqST specifications as an alternative to human-driven designs. Integration of LLMs with model-driven development through VReqST may ease overall VR software development in the future. While VReqST-based requirement specifications create various design versions using LLMs, a VR developer can pick one among many design templates to transform it into code. LLM-based MDD using VReqST for VR Software requires a multi-year effort, and we plan to consider this as our future work.

1.1.3 Scope of our Research

The challenges illustrated in section 1.1 require a multi-disciplinary approach carving across Software engineering principles, Software Product Development methodologies, and Human-Computer Interaction for the Virtual Reality Domain. However, this thesis discusses a technological perspective of Virtual Reality while borrowing knowledge and concepts from core software engineering. This thesis aims to identify directions to ease the development of Virtual Reality products by delivering an approach for new evaluation methods and tools across the VR product development lifecycle.

- This thesis discusses our attempts to understand the VR practitioners' ecosystem through empirical studies and identify thrust areas for improvement and potential research.
- By identifying the thrust areas, It also discusses systematic literature reviews in these thrust areas on academic research databases to deduce currently available practices and review their impact on current VR product development.
- With the extent of improving the thrust areas in VR, this thesis illustrates VR as a domain through a role-based model template that acts as a meta-model for VR. This aids the VR community in creating a customizable semantic knowledge base to build new tools for VR product development.
- Using bare-minimum VR semantics, we formulated a requirement specification tool called VReqST that helps the VR community elicit and specify VR requirements effectively. It demonstrates the development of a tool for VR using a model template. Similar tools can be developed for VR to simplify other development tasks like VR scene design versioning, scene generation, 3D environment generation, article generation, open-source physics engines, test-case creation, VR code versioning and VR code-build management.

- This thesis validates the capabilities of VReqST by running a iterative validation study with VR community. We revised VReqST based on the feedback from Industry practitioners and have used it to specify requirements for a VR based depression detection application as a proof-of-concept. These are the significant contributions of this thesis.

1.2 Contributions

In this section, we illustrate our overall contributions to our this thesis.

Understanding VR Developer Community: We conducted detailed multi-year empirical study on understanding the day-to-day challenges faced by VR developer community in practice and published a detailed work at ISEC 2019.

- 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 in Software Engineering Conference (formerly known as India Software Engineering Conference) (ISEC'19). Association for Computing Machinery, New York, NY, USA, Article 3, 1–11. https://doi.org/10.1145/3299771.3299772

Examining the VR Developer Practices: Based on our insights from the detailed empirical study about challenges faced by VR developer community, we examined various methods that are practiced by VR developers during the Enterprise VR product development in the form of systematic literature reviews. Our primary thrust areas for reviews are Requirements Engineering, Software Quality and Usability. We published our observations at SAC 2019, SIGAPP 2019 Journal, ISEC 2021, ISEC 2023 and ACM Transactions of Software Engineering Methodology Journal 2024.

- Sai Anirudh Karre, Neeraj Mathur, and Y. Raghu Reddy. 2019. ***Usability evaluation of VR products in industry: a systematic literature review.*** In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19). Association for Computing Machinery, New York, NY, USA, 1845–1851. https://doi.org/10.1145/3297280.3297462

- Sai Anirudh Karre, Neeraj Mathur, and Y. Raghu Reddy. 2020. ***Understanding usability evaluation setup for VR products in industry: a review study.*** SIGAPP Applied Computing Review Journal, Volume 19, Issue 4 (December 2019), 17–27. https://doi.org/10.1145/3381307.3381309

- Mohit Kuri, Sai Anirudh Karre, and Y. Raghu Reddy. 2021. ***Understanding Software Quality Metrics for Virtual Reality Products - A Mapping Study***. In 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) (ISEC 2021). Association for Computing Machinery, New York, NY, USA, Article 8, 1–11. https://doi.org/10.1145/3452383.3452391

- Sai Anirudh Karre, Raghav Mittal, and Raghu Reddy. 2023. ***Requirements Elicitation for Virtual Reality Products - A Mapping Study.*** In Proceedings of the 16th Innovations in Software Engineering Conference (ISEC '23). Association for Computing Machinery, New York, NY, USA, Article 12, 1–11. https://doi.org/10.1145/3578527.3578536

- Sai Anirudh Karre, Raghav Mittal, and Raghu Reddy. 2024. ***Requirement Engineering Methods for Virtual Reality Software Product Development - A Mapping Study***, ACM Transactions on Software Engineering Methodology Journal, pp:1-30, https://doi.org/10.1145/3649595. (Accepted November 2023, Published February 2024).

Meta-model of Virtual Reality as a Technology Domain: After concluding our comprehensive systematic literature reviews, we conceptualized a meta-model for Virtual Reality technology domain. This is to ideate new tools for VR developers in the observed thrust areas and provide strong tool support to ease their overall product development. We published the Virtual Reality meta-model based on the concepts of role-based modeling approach in ASE 2022.

- Sai Anirudh Karre, Vivek Pareek, Raghav Mittal, and Raghu Reddy. 2023. ***A Role Based Model Template for Specifying Virtual Reality Software.*** In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22). Association for Computing Machinery, New York, NY, USA, Article 225, 1–5. https://doi.org/10.1145/3551349.3560514

VReqST - Tool for Requirement Specification: Using the Virtual Reality role based model template i.e. a meta-model of VR, we developed a requirement specification tool to specify VR requirements called VReqST. We worked with VR community from Industry and revised the tool incrementally to make it more usable and adaptable. We submitted our tool to RE 2024 and a detailed validation results to Frontiers Journal on Virtual Reality, 2024. We also published a short validation study in ACM CompEd 2024 to understanding capability of VReqST as a teaching aid to specify complete and correct specifications when authored by group of novice requirement analysts.

- Sai Anirudh Karre, Raghu Reddy. 2024. ***Model Based Requirement Specification for Virtual Reality Software Products***, Frontiers Journal on Virtual Reality, 2024 [*TO BE SUBMITTED*]

- Sai Anirudh Karre, Karthik Vaidhyanathan, and Y. Raghu Reddy. 2023. ***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 (CompEd 2023). Association for Computing Machinery, New York, NY, USA, 195. https://doi.org/10.1145/3617650.3624936

1.3 Outline of the Work

In this section, we discuss the organization of our thesis.

Chapter 1: So far, in the current chapter, we discuss our early observations about Virtual Reality Technology which motivated us towards formulating our problem statement for research. We discussed our research overview at length along with is scope. We also detailed our contribution in terms of publications and patents paved a direction to identify solutions to our motivation.

Chapter 2: In this chapter, we present detailed background about Virtual Reality, Definitions, Key concepts related to VR along with VR Technology Stack. We further detail the prevailing software engineering challenges while developing VR software products.

Chapter 3: In this chapter, we discussed the state-of-the-art methods and approaches practiced by the VR developer ecosystem collected from our multi-year empirical study. We present the methodology of our empirical study. We also present our results from the survey, focused interviews and online forum threads. We illustrate our key findings by presenting thrust areas for research in VR domain to ease overall VR product development. We also present various methods practiced by VR community in thrust areas like usability and software quality through systematic literature reviews.

Chapter 4: Considering requirement engineering being the key area in the life cycle of VR product development, we present detailed systematic mapping study on methods practiced by VR community to elicit, specify and analyze the requirements for build VR software products. As *requirements* is one the key thrust areas for our research interest, we explore in detail to understand the bare-minimum attributes that are required for building VR products to ease requirement gathering for VR.

Chapter 5: In this chapter, we illustrate a role-based model template to specify a VR Software product. These present our methodology, approach towards data collection, analysis and overall observations in regard to model template. We suggest the ways of relying on our model template for developing productivity tools for VR to ease VR software development.

Chapter 6: In this chapter, we introduce VReqST - a model based requirement specification tool for building VR software. We describe the tool overview, validation workflow, authoring behaviors and state in VReqST and its additional features. We also present few VR scene examples to illustrate sample specifications developed using VReqST.

Chapter 7: In this chapter, we discuss the validation strategies of VReqST in practice. We illustrate our endeavors toward reaching the VR Enterprise for adoption and feedback. We provide details about the feedback we received in iteration and our strategy to improve VReqST for better adoption. We also present our proof-of-concept on leveraging VReqST to specify requirements for a VR based depression detection application.

Chapter 8: In this chapter, we illustrate details on software and hardware requirements to configure and deploy VReqST. We also included required configuration settings that are required to be applied on underlying supporting software to implement VReqST as in-house or as a cloud offering.

Chapter 9: In this chapter, we present our future work on extending our role-based model template along with VReqST requirement specifications to transform the prevailing VR product development life cycle into a Model-driven VR product development in practice using 3D large learning models and Generative AI.

Chapter 2

Overview of Virtual Reality

2.1 Background

Virtual reality (VR) has a rich history from the 1950s. The concept of VR was initially explored by researchers who sought to create immersive environments that could deceive the human senses into perceiving a realistic alternate reality. Here are some key milestones in the history of VR:

- **1962:** The Sensorama was a machine that is one of the earliest known examples of immersive, multi-sensory (now known as multimodal) technology. This technology, which was introduced by Morton Heilig and is considered one of the earliest virtual reality (VR) systems.
- **1965:** Ivan Sutherland presented the concept of the Ultimate Display, envisioning a virtual world viewed through a Head-Mounted Display (HMD).
- **1968**: Ivan Sutherland and Bob Sproull created the first virtual reality head-mounted display (HMD) system, named The Sword of Damocles. It was suspended from the ceiling and required users to be strapped in although it was too heavy for comfortable use.
- **1969**: Myron Krueger developed ”artificial reality” experiences, leading to the development of VIDEOPLACE technology.
- **1972**: General Electric built a computerized flight simulator with a 180-degree field of vision.
- **1975**: Krueger’s VIDEOPLACE was the first interactive VR platform, using computer graphics and projectors
- **1977**: Developed at MIT, the Aspen Movie Map enabled users to virtually tour Aspen, Colorado, through video filmed from a moving car.
- **1982**: Finger-tracking gloves for VR called ”Sayre” gloves were invented, laying the foundation for data gloves in early VR

- **1984**: Jaron Lanier founded VPL Research, one of the first companies to develop and sell VR products.
- **1989**: Crystal River Engineering developed real-time binaural 3D audio processing for VR applications
- **1990**: Jonathan Waldern exhibited Virtuality, a VR arcade machine, at the Computer Graphics 90 exhibition in London. This was the first mass-produced VR entertainment system.
- **1991**: Sega introduced its VR-1 motion simulator in its SegaWorld arcades.
- **1991**: NASA scientist Antonio Medina designed a VR system for driving Mars robot rovers from Earth in real-time
- **1994**: SEGA released SEGA VR-1, a motion simulator arcade machine.
- **2010**: The first prototype of the Oculus Rift headset was designed.
- **2014**: Facebook acquired Oculus for $2 billion, and Sony announced the launch of Project Morpheus, a VR headset for its PS4 console.
- **2015**: Google launched Cardboard, which uses a head mount to turn a smartphone into a VR device.
- **2016**: The first generation Oculus Rift device was released, followed by Sony's PlayStation VR (PSVR).
- **2017**: Microsoft launched the Xbox One X, its VR-ready games console and headset.
- **2018**: Facebook released its untethered Oculus Go headset, and Lenovo's Mirage Solo, the first headset running Google Daydream, became available.
- **2019**: Sony announced that it had sold more than four million PSVR headsets.
- **2021**: More than 85 million VR headsets were in use in China.
- **2023**: Cloud-based VR gaming is expected to be increasingly prominent, supported by 5G networks.

These milestones represent the evolution of VR technology from its inception to its current state, with improvements in both hardware and software, as well as the development of ecosystems to support its adoption. The history of VR showcases a continuous drive towards creating immersive digital experiences that engage multiple senses and transport users into interactive virtual worlds.

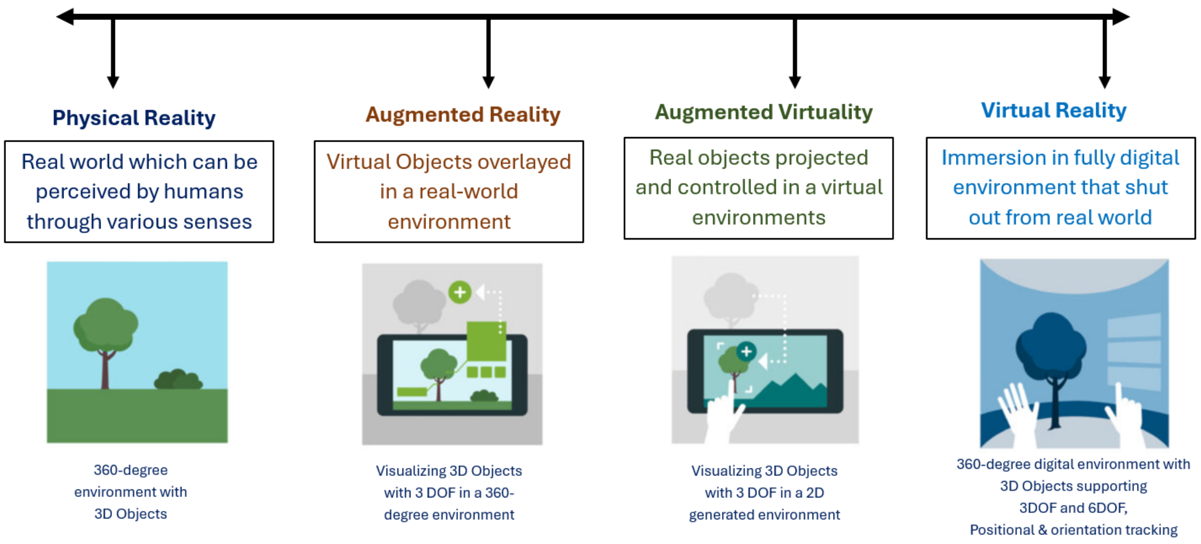

Physical Reality
Real world which can be perceived by humans through various senses
360-degree environment with 3D Objects
Augmented Reality
Virtual Objects overlayed in a real-world environment
Visualizing 3D Objects with 3 DOF in a 360-degree environment
Augmented Virtuality
Real objects projected and controlled in a virtual environments
Visualizing 3D Objects with 3 DOF in a 2D generated environment
Virtual Reality
Immersion in fully digital environment that shut out from real world
360-degree digital environment with 3D Objects supporting 3DOF and 6DOF, Positional & orientation tracking

Figure 2.1 Understanding the dimensions within Modalities of Reality

2.2 Modalities of Reality

Figure 2.1 provides a detailed illustration of modalities of reality. The modalities of reality refer to the various aspects or dimensions of reality that can be distinguished. In psychology and therapy, reality is often discussed through concepts like perception, cognition, emotion, and behavior.

- **Perception** refers to how individuals interpret and organize sensory information from the environment. In reality therapy, perception is closely linked to responsibility, as people are encouraged to take accountability for their choices and actions.
- **Cognition** involves mental processes such as thinking, learning, and problem-solving. Reality therapy emphasizes the present moment and encourages individuals to focus on modifying their actions to change behavior rather than delving into the past.
- **Emotion** are a crucial aspect of human experience and are often intertwined with perception, cognition, and behavior. Reality therapy does not focus on the underlying causes of emotional issues but rather on helping individuals take responsibility for their behavior and make more desirable choices.
- **Behavior** is a central component of reality therapy, which views it as a choice rather than a symptom of a mental health condition. The goal is to help individuals recognize the reality of their choices and choose more effective behaviors.

The above concepts vary with transitions in the modalities of reality. As shown in Figure 2.1, Physical reality is what we observe daily that can be perceived through our human eye as we move along the spectrum of reality modalities - augmented reality and augmented virtuality are converse to each other.

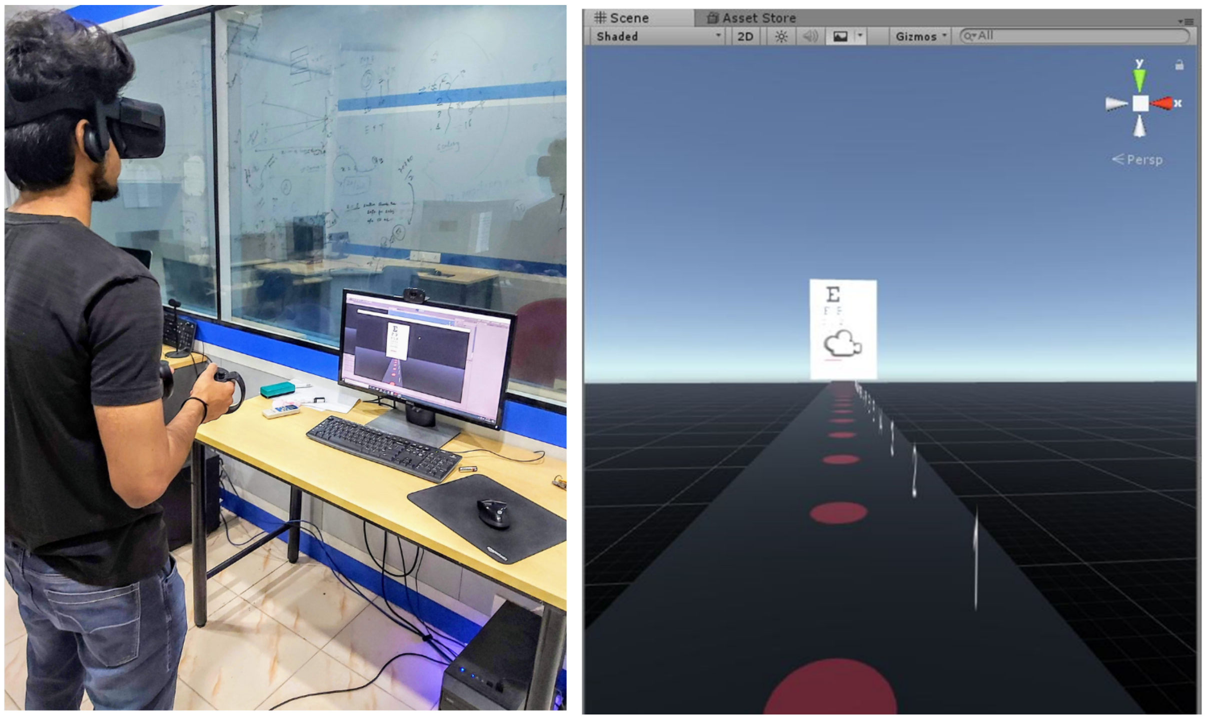


Figure 2.2 Participant exploring a Visual Acuity Test through a VR Scene

Augmented Reality, i.e., AR, is a projection of virtual objects overlayed in a real-world environment when visualized through a hand-held or a head-mounted glass device. Similarly, Augmented Virtuality, i.e., AV, is when real-world objects are projected and controlled in a virtual environment through a hand-held or a head-mounted glass device. To the end of the modalities of reality spectrum, we have Virtual Reality, i.e., VR, an immersion in a wholly digital environment that shuts off from the real world. In this thesis, we are primarily concerned with virtual reality (VR) technology, which is defined as follows.

Virtual Reality (VR) is an immersive technology that lets user be a part of a virtual environment where the interaction between user and simulation/computer-generated environment can be exploited by human sensory simulation like touch, vision, hearing and force perception [126].

Sensory simulation in VR may use a range of devices, from so-called ”immersive” wearable displays to conventional desktop or room-scale computer technologies, and from whole-/part-body motion capture or tracking devices and traditional gaming controllers to physiological interfaces (such as eye-tracking and brain-computer links). VR technology is increasingly being adopted in various domains like healthcare, gaming, aviation, etc. Fig 2.2 shows a participant wearing a VR Head-mounted device and experiencing a Visual Acuity Test VR Scene.

2.3 Key Concepts

In this section, we will go through few key concepts that are required for us to understand the length and breadth of Virtual Reality technology.

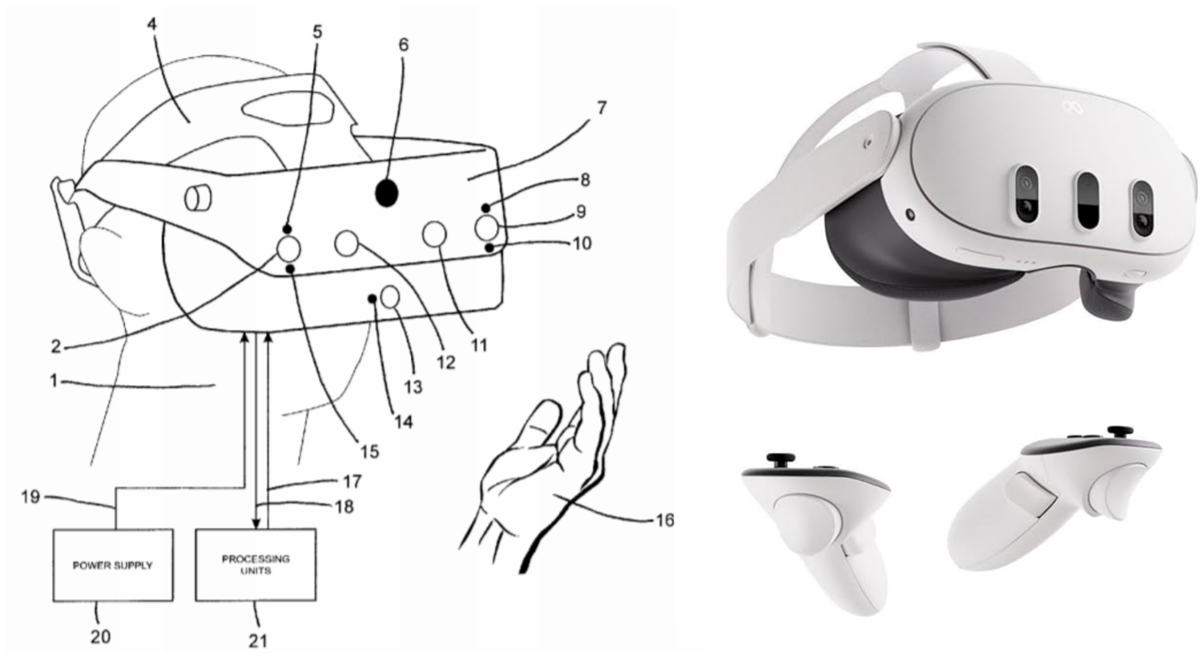


Figure 2.3 Meta Quest 3 Head Mounted device

Head-Mounted Display (HMD) - is a device that presents visual information directly to the user's eyes via a small display unit attached to a headband or helmet. HMDs are used in applications such as virtual reality (VR) gaming, military training, medical procedures, and engineering simulations. They can be monocular, featuring a single display for one eye, or binocular, with separate displays for each eye to create stereoscopic 3D effects. HMDs often incorporate other technologies, such as positional audio, eye tracking, and hand tracking, to enhance the overall experience. Fig 2.3 illustrated the patented meta quest 3 by Meta Inc[1].

Degrees of Freedom (DoF) - In context to real-world objects, DoF is used in various fields, including physics, engineering, and statistics, to describe the number of system parameters that may vary independently. In simpler terms, it represents the number of independent ways a system can be adjusted or changed. The concept of DoF is essential because it helps us understand the range of possibilities

[1]MetaQuest 3 Head-Mounted Device - https://patents.google.com/patent/US20140361956

within a particular system and the constraints that govern those possibilities.Thus, in simple terms, the degree of freedom DoF is a term that describes the axes along which an object can move.

In context to virtual reality, **2DOF** and **3DOF** refer to the degrees of freedom (DoF) that a VR head mounted device supports. DoF is a term that describes the axes along which an object can move. For example, your elbow joint has one degree of freedom (1DOF). In VR, headsets can track movement along multiple axes, and the number of axes tracked determines the DoF of the headset [169]

- A **2DOF VR headset** provides orientation tracking around the three axes where an object can be rotated. This is suitable for mobile VR headsets and standalone VR headsets like the Oculus Go, where users can look around in the environment but cannot move around physically.
- A **3DOF VR headset** provides orientation tracking and allows users to look around the environment. It is suitable for mobile VR headsets and standalone VR headsets like the Oculus Go. Users can visualize a virtual reality around them and look at its details.
- A **6DOF VR headset** provides both orientation and position tracking, allowing users to move in 3 axes (x, y, and z) and any combination of these three axes. This is necessary for users who want to walk around the reality they are experiencing, explore open-world virtual reality games, or interact with objects in the environment.

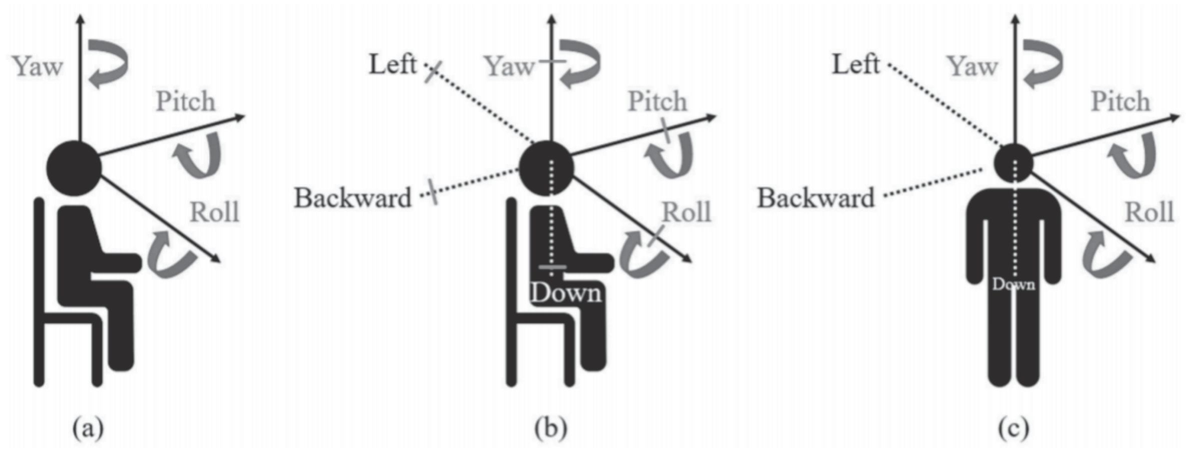


Figure 2.4 Viewing angle and degrees of freedom in VR: (a) 3-DoF, (b) 3-DoF+, (c) 6-DoF

The choice between 2DOF, 3DOF, and 6DOF depends on the application and the desired level of immersion. 3DOF is suitable for experiences where users do not need to move around physically. At the same time, 6DOF is necessary for experiences where users need to interact with the environment more realistically. The cost and complexity of the hardware and software required for 6DOF tracking are higher than for 2DOF or 3DOF tracking, which can affect the headset choice.

Virtual Reality Disruptive Capabilities: The following are significant capabilities of VR technology that play a disruptive factor in adopting this technology for building enterprise products.

- **Immersion**: It is the sensation of being present in a virtual environment, where the user feels as if they are physically present in the virtual world. It is achieved through high-quality graphics, 3D audio, and haptic feedback.
- **Interactivity**: It is the ability to interact with the virtual environment, such as by moving objects, manipulating the environment, or controlling the viewpoint. It is achieved through motion tracking, hand tracking, and other input devices.
- **Presence**: Presence is the feeling of being present in the virtual environment, where the user feels as if they are truly there and not just observing a simulation. It is achieved through a combination of immersion and interactivity.
- **Spatial audio**: It is the use of 3D audio to create a sense of space and depth in the virtual environment. Spatial audio can help users locate objects and sounds in the virtual world and can enhance the overall sense of immersion.
- **Motion tracking**: It is the use of sensors and algorithms to track the user's movements and translate them into the virtual environment. Motion tracking can be used to track the user's head, hands, and body movements, allowing for a more natural and intuitive interaction with the virtual world.
- **Haptic feedback**: It uses physical sensations to enhance the user's interaction with the virtual environment. Haptic feedback can include vibrations, force feedback, and temperature changes. It can help users feel more connected to the virtual world.
- **Real-time rendering**: It is the ability to generate high-quality graphics and animations in real-time, allowing for a smooth and responsive interaction with the virtual environment. Real-time rendering is essential for creating a convincing and immersive VR experience.
- **User interface**: The user interface is how users interact with the virtual environment, including controllers, gestures, and voice commands. A well-designed user interface can help users navigate the virtual world and interact with objects and other users.

2.4 Virtual Reality Technology Stack

In Virtual Reality, a 3D computer generated environment is expected to act and feel real. VR based systems can facilitate participants to interact with objects in virtual world and thereby create a perception of participants being a part of this virtual world. Just like any modern system, the major technology aspects of virtual reality are its hardware and software. The hardware aspect of Virtual Reality primarily constitutes Head Mounted Device (HMD) along with some external and internal sensors. The HMDs include a liquid crystal display, liquid crystal on silicon, Organic light-emitting diodes, eye trackers, hand-trackers, positioning tracking systems, head-tracking, and many other sensors which aid computer-generated imagery. There are several types of HMDs and evolution of HMDs can be traced back to a few decades [74]. Virtual Reality software systems are Object-oriented, scene-graph based, and three - dimensional graphic systems written in third generation programming languages like C++. It provides building blocks for composing three-dimensional scenes, animated three-dimensional objects, three-dimensional interaction, and the associated haptic controllers for tracking actions and gestures. Fig 2.5 gives an overview of the Technology Stack of VR software and hardware system. Software and

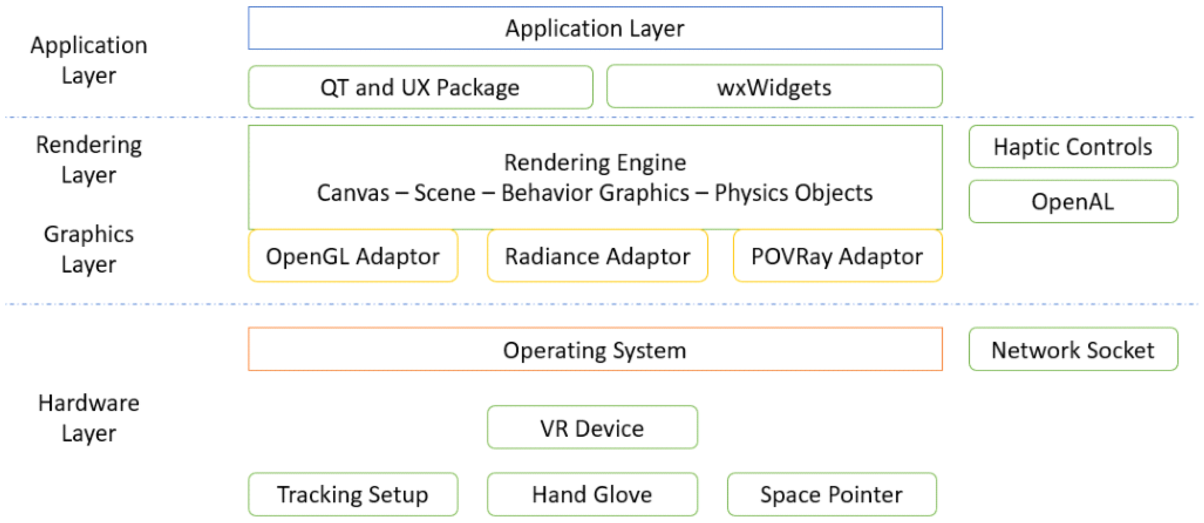


Figure 2.5 Hardware - Software Technology Stack of Virtual Reality Technology

Hardware support for VR are limited to the scope of human senses, which include sight, hearing, and touch only. Currently, VR has the potential to capture requests and responses from Human Eyes, Ears, Limbs, and speech. Researchers are exploring ways to extend the software-hardware support to smell, feel, and emotional stimuli. These senses deliver requests and responses from peripheral hardware devices like Head Mounted Devices (HMDs), Desktop Machines, Mobile Phones, Gamepad or Keyboard, and mouse to integrated GPU circuits (which include mobile, desktop, or custom hardware). These GPU circuits interact with a software kernel that can load drivers for gestures, speech, graphics, and latency. Fig 2.6 provides the high level taxonomy of virtual reality offering [185].

2.4.1 Virtual Reality Software

Traditionally, Virtual Reality applications are written in C++ using the 3D graphics library. Virtual Reality Modelling Language (VRML) is first introduced in 1993 to generalize design and modeling of virtual reality scenes. It also reduces hardware dependencies to some extent. Web3D Consortium, a working group under the World Wide Web Consortium set up in 1997 introduced a web-based 3D graphics standard. With advent of variations in hardware support for Virtual Reality, a variety of hardware HMDs released into the market. Formfactor, visual characteristics play an essential role in determining the VR hardware. These characteristics are considered by developers while developing a VR Scene for enterprise applications. In the 2010s' decade, VR hardware matured with introduction of a variety of sensory systems, which helped developers to capture human sensory data. Gaming Engines like Unity, UnReal extended its engines to accommodate Virtual Reality product development by introducing extended physics and ecosystem binaries to developers. WebVR was started as an experimental JavaScript library for realizing VR on Web, now supports all HMDs along with all mainstream web browsers. Following are few widely used Software Development Toolkits and Game Engines for VR Product development.

- **OpenVR** - is an SDK and an API that was developed by Valve in order to support HTC Vive and other related VR headsets.

- **PSVR-devkit/ Sony PlaystationVR** is created to develop games and apps for Sony playstationVR.

- **Oculus Integration/Oculus SDK** is one of the best VR SDK as it includes various samples, assets and specific kits as well as audio's packs to build VR applications.

- **Cross Platform SDK** - Google had developed Google VR SDK this sdk can used to build VR experiences in Daydream enabled Smartphone or google cardboard. Amazon had developed Amazon sumerian which supports wide range of Virtual Reality devices like Oculus quest2/quest, Oculus Rift, Google daydream, HTC Vive, HTC Vive pro.

- **Unity** is a popular game development engine that can be used to build powerful VR applications. Unity has asset store which can be used to import premade 3d models to build applications faster. Using unity we can develop applications which can run on Oculus quest/rift, HTC Vive, HTC vive pro, Google Daydream.

- **Unreal** also offers powerful VR development tools. We can develop applications which supports wide range of devices like Oculus, Sony PlaystationVR, HTC Vive, Daydream, Samsung gear VR.

- **Cryengine** is well known for Game developers. It also provides a powerful VR software development tool. This supports popular VR platforms like Oculus, Xbox , Sony PlaystationVR etc.,

- **WebXR Device API** provides access to input and output capabilities commonly associated with Virtual Reality (VR) and Augmented Reality (AR) devices. It allows you develop and host VR and AR experiences on the web. It provides access to input (pose information from headset and controllers) and output (hardware display) capabilities.
- **A-Frame** is a web framework for building virtual reality (VR) experiences. A-Frame is based on top of HTML, making it simple to get started. But A-Frame is not just a 3D scene graph or a markup language; the core is a powerful entity-component framework that provides a declarative, extensible, and composable structure to three.js.

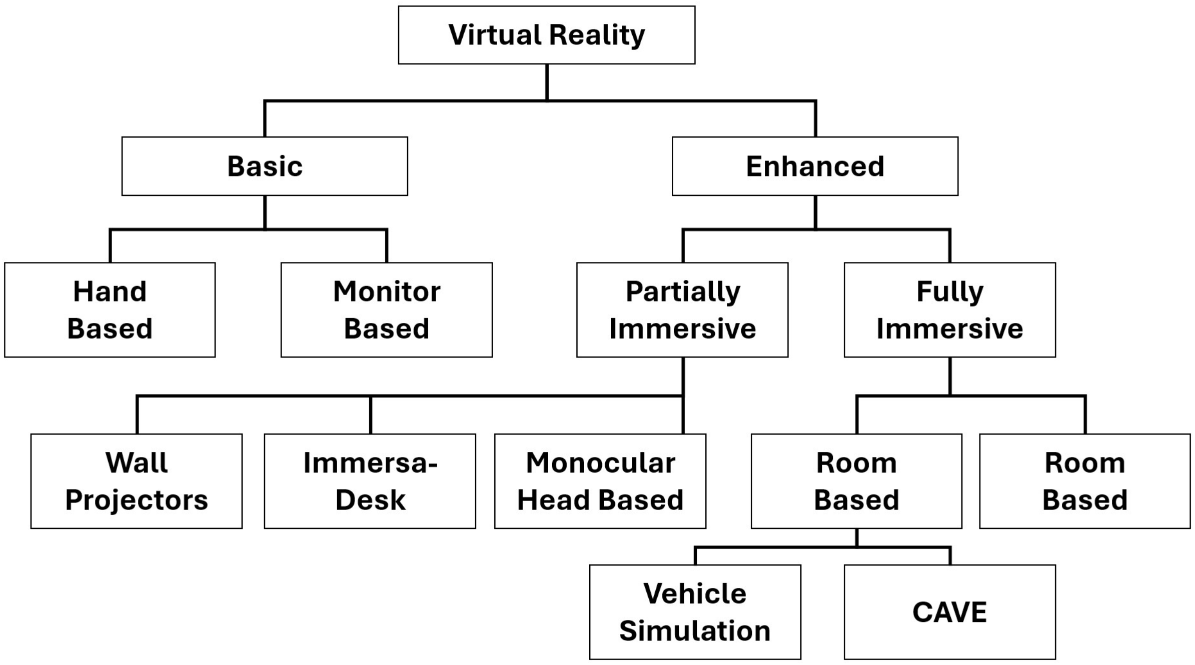


Figure 2.6 Taxonomy of Virtual Reality Offering [185]

2.4.2 Virtual Reality Hardware

Virtual Reality headsets now a days are based on smartphone technology which includes: gyroscopes, accelerometers and magnetometers which are used for tracking head, hand and body positions. Some VR devices even have eye tracking. Tiny HD screens for stereoscopic displays and powerful lightweight processors have resulted in VR devices being less expensive and usable. In addition to the development of affordable omnidirectional cameras, production of VR images and videos has increased, which are used to record 360 video photography at a relatively low resolution or highly compressed format that can be used again for streaming 360 videos. Special devices like CAVE or HMD are used to create an immersive feeling. Two images are generated and displayed from different perspectives to create a spatial impression. To make interactions in the virtual world special input devices are required like motion controllers, wired gloves etc. Controllers uses optical tracker for location and navigation of user to move freely without any wires. Additional other senses or feel can be obtained from omnidirectional threadmills along with vibrating gloves and suit which causes force input to the hand or other parts of the body.

Chapter 3

Virtual Reality Community Ecosystem

This chapter introduces the composition of the Virtual Reality developer community ecosystem. We present our detailed observations on how VR product development differs from conventional software development. We also present our insights on these observations, which helped us identify potential thrust areas that require attention from the software engineering research community to ease overall VR product development.

3.1 Is Virtual Reality Product Development Different?

Over the years, software product development has moved from adopting ad-hoc processes to structured processes. Empirical evidence suggests that standardized methods and tools strengthen the processes for Industrial practice [1]. Periodic tweaks to these methods and tools can improve product quality and help practitioners during product development. Given the rapid technological advances, clearly understanding the methods involved in building and executing disruptive and rapidly changing systems can impact product usage in the market [151]. VR is a disruptive technological advancement in recent times with an impressive commercial marketplace [2]. It is a computer-generated scene that resembles a pragmatic experience in a virtual environment [126]. It was primarily built to generate a simulated training setup and administer virtual exposure to real-life events using head-mounted devices (HMD). VR Games are considered traditional software products due to their rich exposure among gamers [219]. However, this view does not hold for enterprise VR software products.

The traditional software products are now undergoing a vital change by transforming themselves into a VR-based offering to provide a cutting-edge experience to its consumers [30]. Enterprise Software providers are investing in VR-based business solutions to reconstruct the traditional ideas for the new generation of consumers. As per Digi-Capital Market Research Report [2], there has been $3 billion dollar investment in the VR Enterprise product market for 2017. This includes a wide range of markets like Education, Health, Art/Design, HRTech, Services, Tourism, News, etc., to provide a personalized visual experience to respective end-users. These products are now being built by the same software practitioners practicing traditional software engineering principles while building a routine software product [107] for the conventional market. Game developers and designers have created an ecosystem for software developers to enter into VR space [31]. Game developers are good at design and strategy planning. Over a decade, they adopted a few software engineering principles and turned them based on their development needs [186]. They follow development guidelines and a game-based development cycle [15] to improve their internal processes for a better VR product release. With the amalgamation of developers from gaming and software engineering into the VR space, a new ecosystem has now emerged [265]. Using domain knowledge, software developers build enterprise products using game-based VR engines. Oculus, a leading VR tech leader, has opened up developer support programs [3] to fill the talent gap for enterprise VR product development.

Given such progress, there is particular value in studying the software development strategies for building enterprise software in VR. This is possible if an empirical study can be done on the currently indoctrinated practices. Such a study can help in addressing some *open research questions*:

- *What aspects of traditional software development methods are adopted by VR developers?*
- *What challenges are faced during the enterprise VR product development?*
- *Is VR Product development different from traditional software product development?*

In the following sections, we detail an empirical study conducted on 697 VR developers from various countries (predominantly North America, Asia, and Europe) to study the practices and detail some thrust areas that are to be treated differently in enterprise VR product development compared to regular enterprise software.

Methodology

Murphy et al. [186] conducted a study on understanding the significant differences between Video Game Development and Software Development. They followed a two-stage approach, which included qualitative interviews and quantitative surveys. Given the exploratory nature of our research, we developed a different plan for studying VR Ecosystem. We began our research with a simple VR practitioners' survey in Industry to understand the day-to-day experiences of VR practitioners while building VR Products. We realized that the survey was able to capture only superficial information. We could not document the day-to-day practices of VR Practitioners based on survey data. The survey results led us to conduct a focused study for capturing comprehensive data. While formulating the theory from the interview results, we explored other methods that may point us to additional relevant data. The online developer forums are sufficient as they have a unique data set supply. We gained insights into the research questions based on the results from these data sources.

Research Instruments *- Information gathered from a large set of VR practitioners, Focused interviews and Online Developer forums*

3.1.1 Survey with VR Community

Intent - The survey was primarily conducted to understand the *development practices of a wide set of VR developers while building VR Products*. This section provides exhaustive details of our survey setup.

Survey Instrument - Each question has a single idea. There is defined scope to consider, such as time period and tasks relevant to the question. The survey questions are written in neutral, simple and easy language to avoid leading respondent to a specific answer. Response options were made clear, consistent and included a full range of responses that might occur. For categorical responses, we ensured that the options were mutually exclusive and exhaustive and they could pick only one option. There exists a guide to respondents to provide the response in a consistent format and units.

Survey Design - The survey was primarily concentrated on the VR practitioner's day-to-day tasks. We wanted to understand the amount of time spent on respective tasks on a typical day and wanted to understand their contribution towards VR product development. These tasks included strategy building, product design, coding, maintenance, release, quality and usability testing. We also asked participants to share additional tasks performed by them to understand if there were any unconventional tasks.

Participation Selection - We have reached out to 1636 participants from the below four known target population:

- **VR Conference** - Participants from VR practitioner Meet-ups, Conferences, Group meetings and Sales Pitches. A large portion of our survey participation, around 1127 participants, were from this group.
- **VR Developer Community** - Participants from VR Design workshops, Developer workshops, Training centers and VR Unity™testing centers. There were about 140 participants from this group.
- **Online Community** - These participants are active moderators of online VR/AR developer forums and communities. About 310 participants were from this group.
- **VR Enthusiasts** - These participants were not active VR developers but aspirants looking to enter into VR product development. We were able to reach out to 59 such participants as part of our

offline survey in one of the VR Job fairs. Most these participants were from regular industry who were engaged in non-VR product development.

We included *VR Enthusiasts* as part of our survey population as we consider them to be the silver-lining group [283] who can share hidden facts that may not usually be captured in regular focused survey methods.

Analysis - We invited respondents from Online Community through personalized emails to participate in a survey on *"Your Experiences with VR Product Development"* [176]. In case of Offline survey respondents, we invited them to the designated booths of VR Conference, Meet-Up groups and Training Centers with an incentive to participate. The respondents could enter a booth with a free VR roller coaster ride near the booth after successful completion of the survey. We received a total of **697** responses (response rate 43%), of which **512** responses were from the population of **1127** *"VR Conference"* (response rate 45%) and **91** from the population of **140** *"VR Developer Community"* (response rate 65%), 72 from the population of **310** *"Online Community"* (response rate 23%) and 22 from the population of **59** *"VR Enthusiasts"* (response rate 37%). In terms of demographics, 22% of survey respondents are female and 78% are male. Respondents vary in terms of geographic locations: North America (12%), Asia (61%), Europe (8%) and rest of the world (19%). This indicates that participants were diverse and are individual contributors.

3.1.2 Focused Interviews with VR Practitioners

Intent - After a thorough review of the survey results, we formulated a focused interview to capture data from different dimensions i.e. the day-to-day activities, practices, challenges, methods etc. This section provides details of the interview setup.

Participant Selection - We interviewed various VR developer groups with varying thoughts and practices. The representative sample set was kept as diverse as possible using *purposeful sampling approach* [183]. As part of the approach, the goal was not to build a random and generalize sample, but rather to try and represent a range of the relevant sample with a relative exposure to what one is trying to study. To capture such multiple perspectives, we interviewed practitioners from multiple levels. There were few participant's roles which were unique to VR Industry but shared similar responsibilities unlike Non-VR software industry.

Founder is the title for individuals who founded a company with a vision and provide a strategy to build products for the respective market. *Product Managers* are product owners who build the product timeline and define feasible features. *Architects* are system designers who build the fundamentals of the product. *Software Developer* does coding and integrate the modules build by various teams. *UX/UI*

Table 3.1 Participants' designated role and experience

	Exp <5	**5 <Exp <15**	**Exp >15**
Founder	P3, P21	P22	P1, P27
Product Manager	P13	P11, P39, P41	P16, P29, P32, P46
Architect		P26, P47	P10, P30, P42
Software Engineer	P8, P18, P19, P45	P6, P7, P9, P35, P36, P37	P52
UX/UI Engineer	P40	P20, P48	P23
QA Engineer	P5, P14	P31, P33, P34	
UX/UI Designer	P4, P49, P50, P51	P17, P25, P27	
Release Engineer	P15, P53	P2, P43	
Performance Engineer	P12, P44	P24, P38, P42	P28

Designer is a 3D Artist and UX designer for VR environments. *QA Engineer* takes care of quality assurance and conducts testing process. *UX/UI Engineer* is a Usability Evaluator who conducts field tests of VR environment with various participants and provides confirmation of usage and usefulness of the VR product. *Release Engineer* takes care of build and versioning system of the product. They are responsible for maintenance and integration of code-base. *Performance Engineer* evaluates the performance of the VR ecosystem and generates results to Software Developers and UX/UI Designer for improvements. There are many other domain-specific functional roles which are merged with relative roles to avoid duplicity.

Interview Protocol - We reached out to VR Community at various platforms both offline and online, and 53 participants agreed to participate. We asked them to formally introduce their products and talk about their role in VR product development. We started with demographic questions and asked about their experience, level of expertise, experience in the current company, their designation and their specific role. Table 3.1 contains the list of participants by their designated role and their work experience (Exp). We explained the research intent and requested them to discuss their perspectives in an unbiased manner. We assured them that personal details pertinent to the participants and the company they represent will be kept anonymous to avoid operational challenges.

We asked about their day-to-day work schedule, processes they follow, team meetings patterns, other focused meetings (if any) and about understanding as well as actual contribution towards the VR products developed. We later talked in-depth about their practices and asked them about product development attributes significant for their respective role. The attributes are qualities which are followed while building a product [151]. We asked the participant to rank the relevance of the attribute as high or low based on their understanding. Additionally, we asked them about their thoughts on less significant attributes which were not related to their role and captured their rating. The participants also came up with few interesting attributes which helped us include them as part of our initial attribute list for further interviews. The average time of an interview was around 30 minutes and the details captured were based on participant's consent.

Analysis - The interviews were transcribed with relevant consent from the participants in regards to the taping or recording the interview [176]. We depended on an data analysis approach [12] called *'open coding'*. This approach refers to having an open mind while synthesizing the data and not being biased by existing literature or personal experiences. It involves a thorough review of the data to capture as many codes, context and categories as possible [183]. For better clarity, we provide an example of an interview transcript snippet from the *Code$\rightarrow$Context $\rightarrow$Category*.

Interview Transcript Snippet: *'I find field studies as a common tool for ecosystem testing. I personally don't see any challenges with it. It only concerns when we have the feedback based evaluation. There are no right tools to understand the VR environment. They are dynamic and not constant on every release. [P40]'*
Attributes: Process exists but lack of relevant tools causes evaluation issues
Code (Context): Lack of efficient methods/tools (Testing)

Participants were interviewed by the authors independently. The results were reviewed together and the common categories were identified [183]. We conducted a triage with other researchers in this research area and were able to finalize the categories by merging our categories together. The following sub-sections provide in-depth analysis of our observations recorded as part of this study.

3.1.3 VR Online Forum Study

Intent - Online or Virtual developer forums are viewed as knowledge transfer portals [46]. These forums generally tend to facilitate serious discussion among the associates. They can also be considered as a focused discussion group with a significant outcome. As part of an Online or Virtual developer forum, a developer arrives with a business use-case or a problem to obtain responses from domain experts. Different respondents share their thoughts and they arrive at a solution by the end of a discussion. The developer does have a choice to rate the best correct answer as it helps other associates with a similar problem to rely on the same answer. This discussion cycle is endless and over a period of time, the forum turns into valuable knowledge base. Thus, we believed that the content from these developer forums needs to be reviewed to understand thematic challenges in VR developer community.

Research Protocol - Almost all the content in a discussion forum is in natural language. Thus it is difficult to quantify the consequence of the discussion using qualitative methods. We adopted Interaction Analysis Model (IAM) [174] to study the content at different levels and generate the thematic issues and challenges faced during VR Product development. IAM is a knowledge construction approach. This approach is considered to be one of the efficient qualitative methods to assess themes of an online discussion [174]. There are multiple phases involved in IAM approach. We formulated the phases relative to our research without making significant changes to the original approach.

- *Phase I*: Identifying areas of agreement between participants.
- *Phase II*: Identifying areas of disagreement; asking and answering questions to clarify disagreement.
- *Phase III*: Negotiating the meaning of terms and negotiation of relative weights to be used for various arguments.
- *Phase IV*: Testing the proposed new knowledge against existing cognitive schema, personal experience or other sources.
- *Phase V*: Phrasing of agreement and applications of newly constructed meaning.

We used a mechanism suggested by Newman et al. [174] to rate the content based on *Newman, Webb and Cochrane protocol* using a rater scale code. The categories of rating are Relevance $(R+, R-)$, Importance $(I+, I-)$, Novelty $(N+, N-)$, Ambiguities $(A+, A-)$, Linking Ideas $(L+, L-)$, Justification $(J+, J-)$, Critical Assessment $(C+, C-)$, Practical Utility $(P+, P-)$ and Width of Understanding $(W+, W-)$. Each category has a *Positive* and *Negative* Indicator. There may be many other indicators adaptable for detailed analysis. However, as part of our study, we confined ourselves to a simplified version of rating codes. Positive Indicators indicate the state of being relative and Negative indicator indicates the state of being absolute. For each coding category (e.g. J, justification), one counts the number of positive ($x+$ in the above formula) and negative ($x-$) contributions and then calculates the ratio. This produces a measure that is independent of the quantity of participation, reflecting only the quality of the messages. Ratios for an individual category may range from a -1 (all uncritical, all surface) to +1 (all critical, all deep). For example, if we evaluate Relevance $(R+, R-)$ code with $count(R+) = 10$ and $count(R-) = 21$ then $CT = -0.35$. CT value between two or more raters will judge the criticality of the discussion. Such discussions are used for analyzing thematic issues of VR product development among VR developers.

Forum Selection - We conducted a heuristic study on available online VR developer forums. To avoid bias in forum selection, the authors independently reviewed the forums and came to a consensus about the specific forums that needed to be studied in detail.Based on our analysis, we found most of the developer forums to be trivial and not focused on VR Developer community. Thus we focused on VR focused forums apart from the forums hosted by VR Developer Engine providers. UnityForum by Unity™, OculusVR Forum by Oculus™ and VivePort Forum by Samsung Vive™ were three widely used discussion forums with reasonable and significant discussion content. Most of the other developer discussion forums had direct references to these three forums.

Analysis - We conducted this part of the study for about three months. 310 discussion threads were collected from various discussion forums using IAM approach. Codes were defined based on these discussions using *Newman, Webb and Cochrane protocol* and the relative Critical Thinking (CT) ratio was calculated. There were 211 threads which had $CT > 0$ i.e. they were considered to be critical and deep in discussion. We conducted a detailed commentary and listed out thematic process, challenges and practices in a VR Product development setup based on VR developer discussion. Following sub-sections contains in-depth analysis of the observations recorded as part of the research method.

3.2 Observations from Empirical Studies on VR Community

In this section, we present our observations recorded as part of above study setup.

3.2.1 Insights from Survey

Our Survey participants are classified into eight groups: Founder, Product Owner, Designer, Developer, QA Engineer, Usability Engineer, Release Engineer and Others (if any). This division was done based on generalized *perspectives of roles in the field of Software Engineering* [54]. As part the survey, we requested participants to provide details about the amount of % time spent towards VR product development tasks in a given 40 hour work week. Figure 3.1 shows various cluster blocks with details of hours spent by the respective stakeholder on specific tasks. The darker the block, the higher the time spent. This study gave us an understanding of a stakeholder‘s contribution towards a task during VR product development.

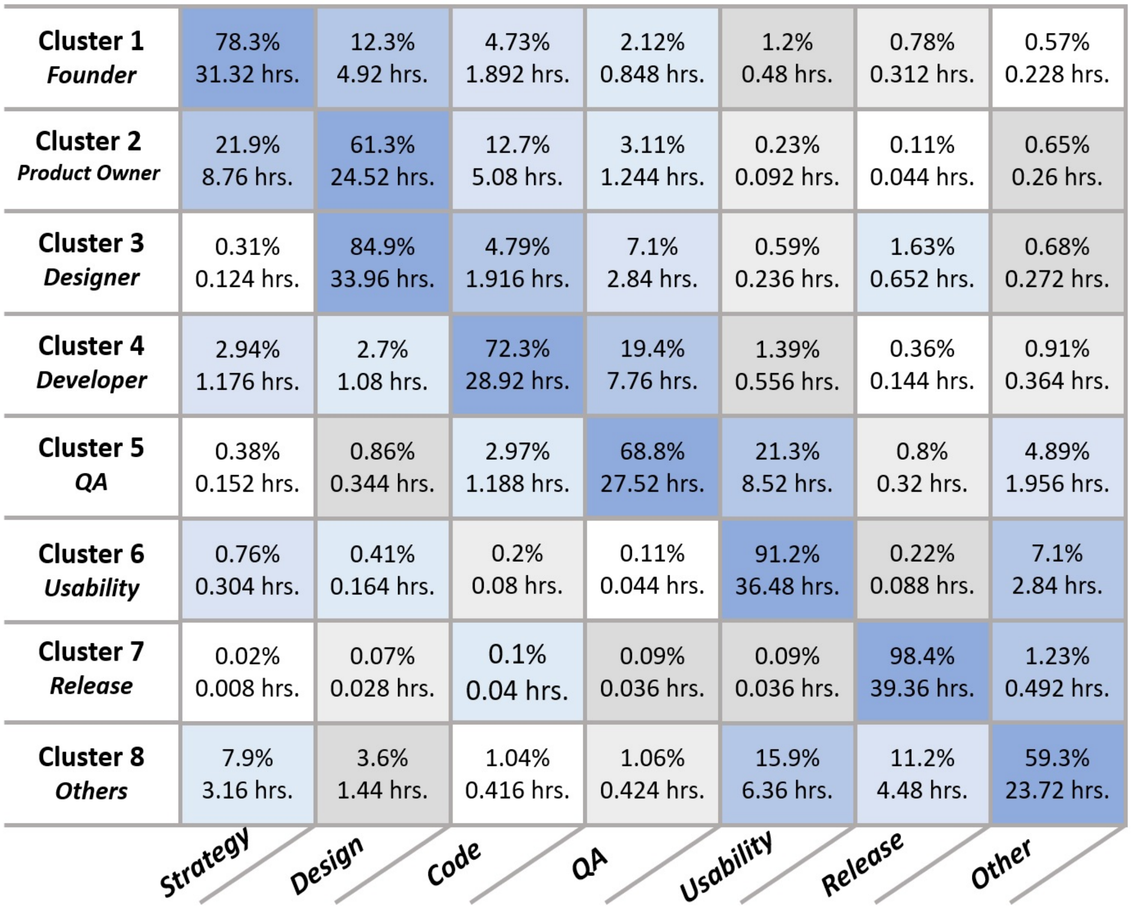
Using the survey data, we were able to conduct statistical T-Test on a few interesting use-cases to understand the significance of the data. Table 3.2 contains the self-explanatory data with significant coefficients of use-cases for a given demographic. A positive coefficient indicates a positive significance to the defined use-case over a demography. Similarly, a negative coefficient indicates a negative significance to the defined use-case over a demography. Comparison between demographics was not intentional. However, we found the results to be interesting as it gave raise to new research questions related to social influences in product development. At this point, we were unable to reach a conclusion about VR product development being any different from traditional software development. The major reason being the comparison of these results with non-VR product development teams. Given the inconclusiveness of the survey results, we used them as a starting point to construct a questionnaire for focused interviews.

3.2.2 Attributes from Interviews

Based on the Interview study setup discussed in previous section, the questions discussed with the participants are made available here [176]. We compiled the responses to each question from all the participants, split them based on user responses, and later re-ordered them based on similarity of intent. We removed redundant comments and grouped the responses based on common code [176]. The authors have independently handled this exercise and administered a triage session with some others researchers to identify the common codes. There was a clear discussion among fellow researchers on finalizing the codes. Figure 3.2 shows the identified common codes using an open coding approach [176].

Table 3.2 Coefficient ratings of Use-case by demographic

Demographic	Use-Case	Coefficient
Asians compared to North American	Involving customers during all phases of development	-0.54 ↓
	Conduct Usability Field Testing before release	-0.32 ↓
	Capture Requirements in plain text	0.71 ↑
North Americans compared to Europeans	Involving customers during all phases of development	0.31 ↑
	Conduct Usability Field Testing before release	0.51 ↑
	Capture Requirements in plain text	-0.22 ↓
Asians compared to Europeans	Involving customers during all phases of development	-0.55 ↓
	Conduct Usability Field Testing before release	-0.27 ↓
	Capture Requirements in plain text	0.77 ↑
Males compared to Females	Conduct brown-bag sessions for Knowledge transfer across Org	-0.33 ↓
Founders compared to Product Owner	Conduct Story board to design overall product flow	-0.37 ↓
QA Engineers compared to Usability Engineers	Maintain Bug Tracker to record defects	0.69 ↑



	Strategy	Design	Code	QA	Usability	Release	Other
Cluster 1 *Founder*	78.3% 31.32 hrs.	12.3% 4.92 hrs.	4.73% 1.892 hrs.	2.12% 0.848 hrs.	1.2% 0.48 hrs.	0.78% 0.312 hrs.	0.57% 0.228 hrs.
Cluster 2 *Product Owner*	21.9% 8.76 hrs.	61.3% 24.52 hrs.	12.7% 5.08 hrs.	3.11% 1.244 hrs.	0.23% 0.092 hrs.	0.11% 0.044 hrs.	0.65% 0.26 hrs.
Cluster 3 *Designer*	0.31% 0.124 hrs.	84.9% 33.96 hrs.	4.79% 1.916 hrs.	7.1% 2.84 hrs.	0.59% 0.236 hrs.	1.63% 0.652 hrs.	0.68% 0.272 hrs.
Cluster 4 *Developer*	2.94% 1.176 hrs.	2.7% 1.08 hrs.	72.3% 28.92 hrs.	19.4% 7.76 hrs.	1.39% 0.556 hrs.	0.36% 0.144 hrs.	0.91% 0.364 hrs.
Cluster 5 *QA*	0.38% 0.152 hrs.	0.86% 0.344 hrs.	2.97% 1.188 hrs.	68.8% 27.52 hrs.	21.3% 8.52 hrs.	0.8% 0.32 hrs.	4.89% 1.956 hrs.
Cluster 6 *Usability*	0.76% 0.304 hrs.	0.41% 0.164 hrs.	0.2% 0.08 hrs.	0.11% 0.044 hrs.	91.2% 36.48 hrs.	0.22% 0.088 hrs.	7.1% 2.84 hrs.
Cluster 7 *Release*	0.02% 0.008 hrs.	0.07% 0.028 hrs.	0.1% 0.04 hrs.	0.09% 0.036 hrs.	0.09% 0.036 hrs.	98.4% 39.36 hrs.	1.23% 0.492 hrs.
Cluster 8 *Others*	7.9% 3.16 hrs.	3.6% 1.44 hrs.	1.04% 0.416 hrs.	1.06% 0.424 hrs.	15.9% 6.36 hrs.	11.2% 4.48 hrs.	59.3% 23.72 hrs.

Figure 3.1 Time spent by Stakeholders in a week

Understanding Market health - We gained some understanding from the interview participants in regards to the importance of understanding the market before building a product. Based on our relative transcripts, it was evident that only few markets are matured for VR and most of them are not ready yet. Designing a targeted VR Product is very difficult as factors like culture, tradition, localization etc. need to be considered. It was clearly apparent from some of responses given by the participants.

[P27] *We together built a great attendance management product, but we couldn't woo clients who can buy it*

[P30] *Our third VR product was better than our regular mobile/web offerings. Clients were reluctant to switch as they didn't get the VR product*

[P43] *I found it to be strange as one of our old customers didn't want to choose a VR app over an old-fashioned web app due to resource challenges.*

[P13] *It was just hype for some clients. Very few wanted a clear shift into VR*

Table 3.3 Interview Questions

Q1	What are your responsibilities towards VR product development?
Q2	What tools do you use to Strategies/ Design/ Test/ Develop VR Products?
Q3	What are the approaches/ processes/ methods do you follow for VR Product Development?
Q4	What challenges do you face while following these approaches/ processes/ methods?
Q5	What are the measures you take or plan to take to mitigate your challenges?
Q6	How do you measure and ensure the quality out of your work?

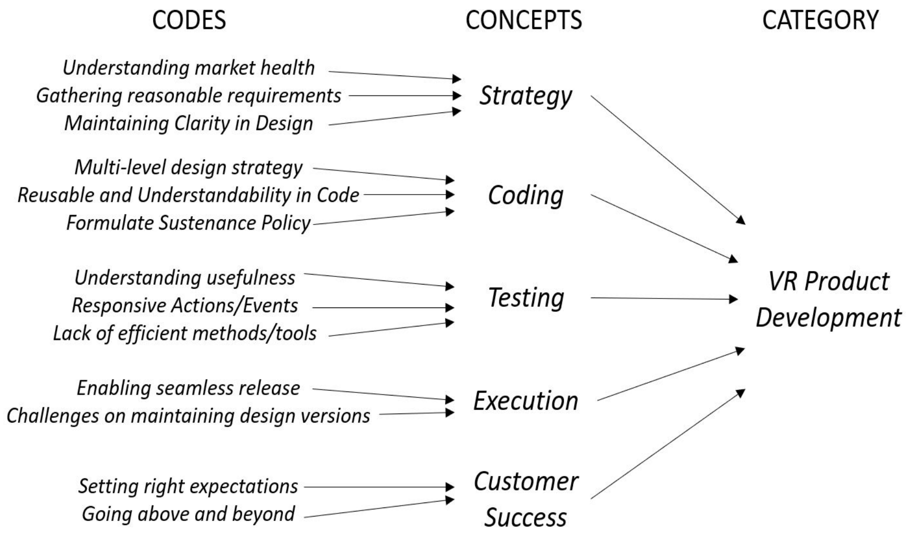

CODES
CONCEPTS
CATEGORY
Understanding market health
Gathering reasonable requirements
Maintaining Clarity in Design
Strategy
Multi-level design strategy
Reusable and Understandability in Code
Formulate Sustenance Policy
Coding
Understanding usefulness
Responsive Actions/Events
Lack of efficient methods/tools
Testing
Enabling seamless release
Challenges on maintaining design versions
Execution
Setting right expectations
Going above and beyond
Customer Success
VR Product Development

Figure 3.2 Common Codes from Interview Study

Gathering reasonable requirements - With a shift from Web/Mobile to VR based apps, requirements tend to flow from different directions. However, identifying the right requirements and capturing them within the scope of the VR product is crucial. We found practitioners adopting traditional methods to gather and analyze requirements were facing serious issues as the techniques did not work in the VR setting.
***[P21]** Clients tend to compare with fellow competitors and fail to provide us valid requirements. They just copy features and ask for enriched ones.*
***[P36]** I couldn't understand the requirements from Design team as they were very vague and unstructured.*
***[P3]** I know I choked my team with inputs. I was actually trying to find a better way to normalize them. But it was too late for some projects. We learned and progressed further.*

Maintaining Clarity in Design - Lack of simplicity in product flow can cause a lot of effort cost to product owners. This has been a traditional problem for software engineers. However, it is too expensive for VR products in terms of time and resources. VR Practitioners are heavily dependent on the design. Due to the limited prototyping capacity in VR, versions of design gets complicated over a period of time. This causes sheer confusion among the VR product groups.
***[P52]** I did see a flow of requirements which are changing every hour. There was no room for simplicity*
***[P8]** I was new to this VR dev space and it was totally design oriented. It is difficult for a new hire to match up as we are not sure on where to begin*
***[P45]** I always had a backup code for all design revisions and maintained the latest code for access. The rest ignore it and we land up in chaos while releasing the module code*
***[P44]** The slightest change in design layout will impact load time of the object rendered on the scene. It is costlier to estimate the need for an object in a scene*

Multilevel design strategy - VR Product Dev teams faced setbacks while building products using top-down approaches. We found that product owners tend to chose hybrid development models to design and develop VR products. They involve all practitioners across different verticals of product development to produce a better and satisfied design.
***[P11]** We understood the need for building a strategy for designing our VR. We conduct 3-5 storyboard sessions to make the design multi-level. This will help the non-designers understand the intent of the product*
***[P39]** - Features are to be set free from frequent updates. I bring uniqueness to each feature. This halts frequent updates and helps related developers and tester to be focused on current design*
***[P40]** - In our second VR project, our design approach was abstract. Our team lost control of code and overall idea. We made sure that we include everyone in a design phase so that they get what exactly we were doing*

***[P25]** - I found the core idea not being transcended to developers and testers from PM team as the design appeared obscure to us*

Reusable and Understandability in Code - The scope of coding in VR is mostly associated with building events, actions, performance and interaction between the objects in the VR scene. Most of the tasks tend to be unique to each and every scene. It is a requisite for a code developer to always be in sync with the designer. Any slightest change or miss could trouble to the overall product design flow.
***[P18]** I didn't code with coding standard in my mind and as a team, we now do face issues*
***[P6]** I hold triage sessions among the developers and we did gain confidence in understanding each others code*
***[P9]** Too much log notes. It is not helping us on how we code. Our leads should come up with a process to help each of us to understand the logic behind the defined scene actions. Expected Vs Actual are not correctly reviewed*
***[P19]** In the past, our code version control was a joke. We read about current methods and we are now trying practicing them. I look forward to a positive change*
***[P24]** I find it difficult to map the code and it's flow while referencing slow performance check-points in VR Scene*
***[P44]** Coders need to inculcate a habit of including comment tags or check-in tags while they conduct a change check-in. I cannot trace the scene which was actually working with the present one*

Formulate Sustenance Policy - Policies defined for large-scale product development were not aligned with the VR product development. As part of Sustenance mode of fixing defects, minute fixes in Scene/-Code/Sound etc. are recorded. These small changes end up becoming a pile of change trackers which becomes difficult for maintaining too many releases builds. This is an open problem, which requires a new release-strategy for VR products which can sustain small changes and help developers track them easily. It appears that there is a need for formulating a post-release strategy for VR products as they are different from traditional software production.
***[P33]** VR bug reports are bizarre. I found them unclear than traditional products*
***[P5]** I worked on C# projects before joining VR dev team. I found bug fixes to be not as per standard as they are not structured*
***[P50]** Design versions are not shared down to QA team correctly. They land up testing an incorrect design version, which is not of relevance to the planned fix*
***[P53]** I tried to understand how our QA works and later realized to help them chalk out a plan for a policy to mitigate invalid check-ins*
***[P52]** We started with a strong bugfix SLA and we do see improvement. However, as the project becomes larger, we found it difficult to pass on bugs to QA team due to too many Dev validation processes. A very less turn-around time causing a peer pressure*

Understanding Usefulness - Most of the practitioners did find obstacles on judging the usefulness of the VR app as there are no adequate conventions to study them while working with the customer. Based on the remarks from practitioners, it was clear that current practitioners are looking for a VR knowledge base which could help them strategize the intent of the product based on past experiences of other practitioners. Failures accumulated by a valuable VR Dev team who focuses on rich VR apps can greatly contribute to avoiding such barriers in VR Community.
[P18] *It is unfortunate that we do not understand whether to build a VR app is usable by the end user. Our end users taught us a serious lesson to our design and testing team to improve the layouts with more examination*
[P30] *I thought accessibility to be a key factor for end users to run the app. It rebounded when we ended up adding too many controllers. We never knew where to limit, we found the user experience testing be promising as it was the only way to find out if we are right and where to stop*
[P14] *It was odd to know that we delivered a unusable VR app to our target audience. Our processes didn't consider usability issues*

Responsive Actions/Events - VR apps provide personalized end-user experience. As all the actions and its relative reactions play a prominent role, it is necessary for practitioners to evaluate responsiveness of the VR app. As per remarks, traditional strategies are to be empirically examined as they do not suffice the needs for such current evaluations. Persona-based evaluations should be encouraged as the VR apps are highly focused on individual experience.
[P42] *Designers tend to ignore few elements which increase the load time of the scene. I removed them and moved the tested VR Scene to QA every time. All such unwanted events are to be filtered at Code level. I formulated an approach to avoid this, but as people are involved - we might find more time for adoption*
[P38] *User events and actions are undeclared and set open while running the VR Scene. The scene turns unresponsive and undergoes irrelevant code re-factoring. Due to lack of clear design and code standards, I am facing regular performance defects. This is unfortunate as with repeated red-flags, PM team does care of it. It has been like an elephant in a room single our initial release*
[P15] *I failed to release a responsive app for the first, second and third time. We re-worked found a flaw in our process. We still follow it as we don't have enough resource to experiment on a new one*

Lack of Efficient methods/tools - Tools are vital for building products. The absence of adequate tools exposes the reality of VR product development. Few of the practitioners were keen on possessing a version control system for design and efficient usability evaluation tools to handle their product maintenance and release.
[P26] *One of the tough challenges we faced is with tools. We do have good design engines and coding tools. But, lack of efficient version control system for design and Usability testing does cost us a lot*
[P3] *We were really not sure on how to test the VR app apart from conducting a standard field evalua-*

tion. We do not find any good tools for beta testing the app before moving it to release

Enabling seamless release - Product Owners are experimenting with existing practices like Kanban, Agile, Scrum etc. to build VR products. These approaches are deemed to be thriving to some degree. Few practitioners have come up with the tailored approaches to ease their overall VR development.
***[P39]** As I was part of a small dev team, I used to distribute the action items to our team and we switched our roles every week so that we learn and cover the actions proposed. We tracked down the health of each task every weekend and discuss possible improvements*
***[P10]** We followed KanBan approach design and release. We evaluate an engineer's contribution based on their availability towards the project*
***[P52]** As I was part of a large company which also works on non-VR products, we were abiding by our regular release processes*

Challenges on maintaining design versions - Design and its related patterns are vital for VR Scene and Content developers. It accelerates their work and helps them build more content in less time. However, with a lack of design version control storing too many versions of the same scene causes an immense hurdle for UI/UX designers.
***[P49]** Our design repository was stacking up with undesired data. We used a divide and conquer procedure to classify the wanted and the unwanted designs*
***[P51]** We started by building a VR scene repository. Our approach was to label the objects in all available scenes and re-use them based on a markup language. This has created a customized asset library which was dedicated to all our projects*
***[P4]** I was owning the design library and our team was out of control on managing the versions as there was no clear version control on designs. We used a tag based method to identify the version and its author. This solved a lot of manual designer effort over a period of time*
***[P17]** We moved our sketch boards of VR scenes to storyboard team every end of the day. In return the next day, we found the design to be totally different. We used a backtracking approach to identify the newly added objects and saved the new version for future reference*

Setting right expectations - Understanding the need and sharing the desired output is widely accepted by customers. With the lack of efficient methods, a product owner might fail to project the designed vs released version of the product. This creates a phenomenal trust issue between the product owner and the consumer. Currently, practitioners find it difficult to track and map the expectation during product release. ***[P31]** Actual code Vs Expected Code was always different. This is causing last-minute hassles in release*
***[P14]** We introduced scrum based processes to identify actual-vs-expected outcome evaluation. This saved us a lot of resource time*

Going above and beyond - Product Owners tend to woo customers by generating best possible results in tough times in spite of the disorganized approaches. The existing methods are not robust enough to handle quick changes or alteration on released products, leading to compromise in VR product development and delivery.
***[P29]** Our product offerings outperformed customer expectations as we delivered projects on time*
***[P1]** Our fourth project was successful as we did fine-tune our release approach*
***[P22]** We maintained a clear client retention policy. We did offer new features at a same cost which was first demoded to them*

3.2.3 Themes from Online Forums

We conducted an IAM analysis [174] on the VR developer forum data and have finalized 211 discussion threads which are rich in the discussion as their Critical Thinking Index was more than zero. We categorized the discussion threads based on their underlying need from a VR Practitioner's perspective. Below details provide the percentage against each category from the overall inquiry circle. This data helped us to a certain extent in recognizing the areas where the VR practitioners need help.

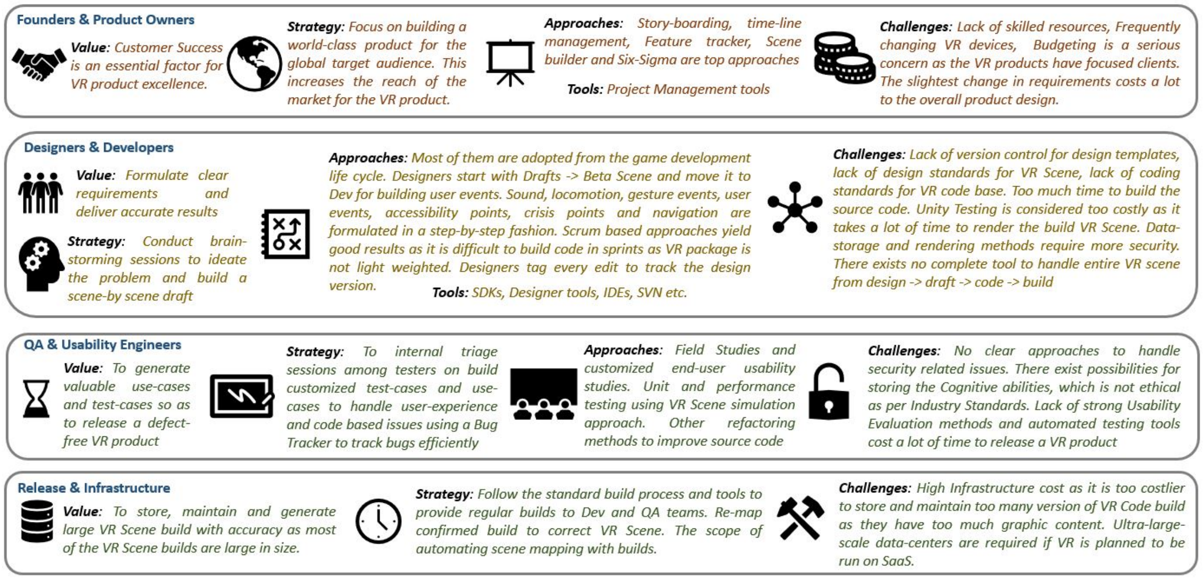
General - **7%** of the threads include regular inquiries about expected behavior, possibilities on new releases, setup, hardware, installation and How/Where to find desired resources. Examples - *"Does Unity 2017 bring something new to VR or can I stick with 5.6?"* and *"How do you build a React VR project as a native app a la VrShell/Oculus Home?"*

Code Level - **52%** of the threads include regular inquiries about code level implementation of novel features, understanding warning, display, build, SDK and errors etc. Examples - *"How to handle OVR-CameraRig.UpdateAnchors () errors?"* and *"Is there any way to set a defined X and Y as a starting point on the 360 video?"*

Documentation - **4%** of the threads include inquiries about the documentation details and ReadMe files of the SDK or JS toolkits. Examples - *"Vive / Viveport FAQ: Where do I get the free/bundled games with my Vive? How do I redeem the codes?"* and *"Docs/diagrams on communications flows between Rift -> Touch -> Constellation"*

Performance - **8%** of the threads include inquiries about the handling performance issues, managing performance level data and other details related to the performance of VR Scenes. Examples - *"How does dynamic clock scaling on S8 work?"* and *"Why are Exynos-based phones able to do 4x MSAA cheaply vs. SD-based ones (2x)?"*

Functionality - **19%** of total threads include inquires about the processes/steps to achieve customized functionality, aspired features which are not directly related to code but is related logic and process.


Founders & Product Owners

Value: *Customer Success is an essential factor for VR product excellence.*

Strategy: *Focus on building a world-class product for the global target audience. This increases the reach of the market for the VR product.*

Approaches: *Story-boarding, time-line management, Feature tracker, Scene builder and Six-Sigma are top approaches*

Tools: *Project Management tools*

Challenges: *Lack of skilled resources, Frequently changing VR devices, Budgeting is a serious concern as the VR products have focused clients. The slightest change in requirements costs a lot to the overall product design.*

Designers & Developers

Value: *Formulate clear requirements and deliver accurate results*

Strategy: *Conduct brain-storming sessions to ideate the problem and build a scene-by scene draft*

Approaches: *Most of them are adopted from the game development life cycle. Designers start with Drafts -> Beta Scene and move it to Dev for building user events. Sound, locomotion, gesture events, user events, accessibility points, crisis points and navigation are formulated in a step-by-step fashion. Scrum based approaches yield good results as it is difficult to build code in sprints as VR package is not light weighted. Designers tag every edit to track the design version.*

Tools: *SDKs, Designer tools, IDEs, SVN etc.*

Challenges: *Lack of version control for design templates, lack of design standards for VR Scene, lack of coding standards for VR code base. Too much time to build the source code. Unity Testing is considered too costly as it takes a lot of time to render the build VR Scene. Data-storage and rendering methods require more security. There exists no complete tool to handle entire VR scene from design -> draft -> code -> build*

QA & Usability Engineers

Value: *To generate valuable use-cases and test-cases so as to release a defect-free VR product*

Strategy: *To internal triage sessions among testers on build customized test-cases and use-cases to handle user-experience and code based issues using a Bug Tracker to track bugs efficiently*

Approaches: *Field Studies and customized end-user usability studies. Unit and performance testing using VR Scene simulation approach. Other refactoring methods to improve source code*

Challenges: *No clear approaches to handle security related issues. There exist possibilities for storing the Cognitive abilities, which is not ethical as per Industry Standards. Lack of strong Usability Evaluation methods and automated testing tools cost a lot of time to release a VR product*

Release & Infrastructure

Value: *To store, maintain and generate large VR Scene build with accuracy as most of the VR Scene builds are large in size.*

Strategy: *Follow the standard build process and tools to provide regular builds to Dev and QA teams. Re-map confirmed build to correct VR Scene. The scope of automating scene mapping with builds.*

Challenges: *High Infrastructure cost as it is too costlier to store and maintain too many version of VR Code build as they have too much graphic content. Ultra-large-scale data-centers are required if VR is planned to be run on SaaS.*

Figure 3.3 Consolidated Observations of a typical VR Product Development Cycle

Examples - *"How to InputTracking current LeftHand/RightHand?"* and *"How to Insert own content in Oculus Go?"*

As part of our study, we were able to capture relevant attributes from Interviews and relevant themes from online forum study. We consolidate the observations and discuss them in the next Section.

3.3 Thurst Areas for Research to ease VR Product Development

We formulate our observations by means of Inductive reasoning on *"Virtual Reality Product Development"* and overall practitioners' practices based on the results captured in the empirical case-study. In this section, we present the data in various dimensions and discuss them in detail. We reason that VR Product Development is diverse from traditional Software Product Development. Figure 3.3 presents us the details of practices followed by VR Practitioners. This includes day-to-day challenges of specific attributes for each stakeholder. These observations are consolidated as key traits essential for VR product development.

Key Traits: *Value, Strategy, Approaches, Tools and Challenges*

The *Value* trait provides the motive of the respective team towards the VR development. *Strategy* presents their plan towards achieving the task. *Approaches & tools* detail them about their current methods involved as part of their development process and finally, *Challenges* provides details of the problems faced during the respective portion of VR product development. These observations are in relevance with the Enterprise VR Industry setup. Additional details of these observations are listed as

Table 3.4 Consolidated observations by Thrust Area

Thrust Areas	***What is working?***	***What is not working?***
Planning	Project Management principles are effective to adopt. It is essential to conduct cost and effort estimation.	Identifying the right target audience and converting wrong product scope to a targeted set of end users.
Requirements	Conducting incremental specification reviews and constantly correlating them with design constraints can show good results.	No definite methodology to elicit requirements. It can be considered as a time-consuming activity due to its sophistication in detailed design.
Design	Some tools are available to design and prototype the scenes, tasks and its relative actions. Out-of-box design patterns are being practiced with minimal effort.	Lack of design versioning causes serious issues while tracking insignificant changes to the design. It takes a similar effort to manage an unimportant change and a vibrant change in design.
Security	Minimal security implementation is in current adoption. Standard security protocols can be realized based on the domain of the product.	No definitive understanding of security standards as VR provides the designers to capture the cognitive abilities of the end users. Security compulsions can be compromised if they are not considered serious. Domain-specific security protocols are in great need.
Usability	Traditional approaches are promising and can address minimal usability evaluation.	Currently, there are no automated methods to address usability concerns. No scope of support to 508 Compliance. It is practically difficult for disabled/ semi-disabled end users to use VR products. It is difficult to evaluate accessibility of VR products.

part of Figure 3.3 for better clarity.

Stakeholder Profiles - We found a variety of new stakeholders in the VR Product Development process which is not in common with regular Software Product Development. Job Profiles like VR Scene Director who primarily takes care of other practitioners like *VR Scene Designers, Aucostic Editors, Audio/Visual Developers, VR Scene Artists, Content Editors and Integration Specialists* are unique to VR product development. These practitioners are part of a core VR Scene development team who build the actual prototype administrated by Product Manager. They take care of sound, light, visual and aesthetic effects of the VR Scene. These profiles are traditionally adopted from Gaming and Multi-media software product development teams. Their expertise is considered to be essential for a rich VR Scene development. *VR based Usability Experts, Cognitive Scientists, and Field Testers* are other minor Job profiles apart from traditional software job profiles. The stakeholders who fit these profiles are primarily into Usability evaluation especially for studying the depth, vergence, and accommodation of the VR Scene.

Customer Impact - The experience of using VR apps is changing rapidly in commercial markets. Consumers tend to experience a variety of applications with lively features. The customer could have a wholly different experience from one VR headset to another. As VR apps are highly dependent on hardware, it becomes significantly difficult for product owners to release multiple versions of the same app. In other words, device-specific app development is not an easy process. As a result, product owners either restrict themselves to one VR headset or tend to release multiple versions (if commercially viable) of headset specific VR apps to capture the market share. As part of our study, we found that **58%** of product owners involve their customers during all phases of the product development to ensure that they are going in a right direction. **79%** of product owners involve their consumers during the beta testing (pre-release versions) phase to understand the customer experiences and reactions. Out of the above, almost all i.e **93%** of the product owners obtain a clear design-to-code sign-off before building the actual product. We could infer from the data that the product owners consider customer engagement as a key factor in successful product release. We could see that the customers largely influenced VR product development as VR apps are primarily personalized and persona based in nature.

Areas of Improvement - Heuristically we recognized thrust areas in VR product development. They can be loosely admitted as minimal attributes for constituting a relevant VR product. Based on our study, we could deduce the details of practices found to be favourable for VR practitioners. We were able to record hurdles that cause serious trouble to product owners during product development. We observed a need for further research on these thrust areas so as to visualize a seamless VR product development in future. Table 3.4 provides details of practices which are ***"working"*** and ***"not working"*** for VR practitioners today. It is remarkable that few of these challenges were predicted to be matter of concern in 1968's NATO Software Engineering Conference [221]. It is always arguable that in spite of progress in research, we still tend to face difficulties as the software evolves.

Thrust Areas: *Planning Strategies, Requirement Elicitation, Design Thinking, Usability centered human reasoning and Security*

Development Practices - Coding and Testing play a vital role in VR Product release. VR Scene object actions, events, and related work-flows ought to be handled through code. As part of our study, we could see that VR Developers were trying to accommodate software engineering practices in the course of VR development process. **76%** of product owners conduct regular triage sessions among designers, developers, and testers to have a constant track of product progress. Out of this **76%**, **81%** of product owners maintain product timelines which include regular targets and deadlines. The rest follow internal deadlines which may or may not contribute towards overall product progress. **89%** of the product owners encourage their employees to contribute towards brown-bag sessions so that it helps rest of the practitioners to be up to speed. **54%** of the practitioners are dissatisfied with the traditional usability evaluation approaches as they are not productive and do not yield real-world results.

Complexities in VR Development - Primarily maintaining VR Scene Repositories and its related versions is a difficult task. Trailing small defect fixes and managing its related code builds is tough for Release teams. Usability Evaluation is time-consuming and is costlier for developers in times of generating negative results. Tracking and constant comparison between wire-frame design and the end product is a cumbersome activity. Based on these observations, we conclude that enterprise VR product development is different from the enterprise software product development.

3.3.1 Conventional Software Products Vs VR Software Products

Following are the key difference on how VR software product development is different from conventional software product development.

- **Developing Experiences**: The major driver of VR software products (enterprise VR and consumer VR) is the level of immersive user experience that is closer to realism within confined virtual environments to simulate a realistic interaction. While traditional enterprise or consumer software products also look at user experience, the focus is on meeting the business goals, and user experience is just a contributory factor to meeting the business goals. However, the use cases that are too complex to comprehend in the real world rely on VR as a solution. For example, training designated specialists on protocols to access Bio-Safety Lab (Type 4)[1] is difficult as it contains too many stand-operating procedures for curating high-risk, maximum containment, and life-threatening viruses. Due to its limited access, training healthcare practitioners in a real-world Bio-Safety Lab (Type 4) setup is almost impossible.

- **Multimodality:** Traditional enterprise and consumer software products accept inputs from limited set of hardware sources like keyboards, point devices, etc. VR software products support multimodal interactions [175]. Some of them are Gestures, Gaze, Auditory stimuli using haptic devices, kinaesthetic (related to proprioception), temperature, pressure, pain-sensing, etc. The implication of using multimodal user interactions makes VR software development difficult. Thus it requires additional process and product supervision from the early stages of development.

- **Development Cycle:** In addition to the Requirements-Design-Code-Test-Release style of traditional software product development, VR product developers must address Acoustics, Spatial Presence, Environment Perception, Realism, Environment-User interactions, etc., during the various stages. The development should accommodate both hardware and software development cycles. The synergy between software and hardware practitioners should go hand-in-hand for better results.

[1]https://www.phe.gov/s3/BioriskManagement/biosafety/Pages/Biosafety-Levels.aspx

- **Stakeholders:** VR software products require skilled developers to ideate and implement sound, presence, light, gravity, cognition, and physics. VR Software development also involves hardware practitioners who provide firmware updates to the VR hardware for focused VR applications.

- **Tool Usage:** The Software Development Toolkit for VR Environment design and development is limited. There is a serious need for new tools at all levels of software design. Areas like Requirement elicitation, specification, Testing, and Release lack versioning.

- **Process and Practices:** Majority of the VR Developer community are from the gaming and design industry [107]. They have very low exposure to established conventional software development methodologies due to a lack of foundational training from a VR perspective. It is observed that VR community need help with releasing low-cost VR software as they lack effective methods of Product Planning, Requirement Elicitation, Specification, Design, Security, and Usability. These are observed to be the thrust areas that need better tool and methodology support for VR community.

- **Volatile Hardware:** In short, platform fragmentation in technology happens when a market comprises multiple highly-incompatible technologies or technology stacks, forcing prospective buyers of a single product to commit to an entire product ecosystem rather than maintaining a free choice of complementary products and services. There exists severe platform fragmentation between VR Software and Hardware. There is very low cross-platform support due to proprietary SDKs causing VR community to build VR software for conformant devices. This makes the VR community back away from serious roadmap planning. OpenXR, an open standard, is being formulated by the VR developer community to address this challenge to some extent [97]. However, this still requires community acceptance, maturity in practice and adoption.

Previous studies have shown a need for a new Software Development Life cycle for VR product development as none of the methods fit the VR product development [265]. To validate the same from an industrial perspective, we conducted this empirical study within the VR community to learn the current practices and challenges while building VR products in the Industry. As part of this empirical study, we uncovered that VR practitioners follow the approaches that fit their temporary project needs to succeed in their business. However, most product owners are attempting to experiment with the available diverse approaches as they need more confidence in the robustness of existing approaches and are unsure of the outcomes. They follow their predecessor's failures and plan their future product development to avoid loss. We find new areas that researchers can study based on our empirical observations to help VR product teams build better products with minimal loss. Usability has become a significant hassle for product owners as none of the current approaches help them understand their customers, as the VR products are persona-focused but not social. Accessibility is an area that researchers can look at in-depth, especially for VR products. The VR Ecosystem is ready to adopt new trends in Software Engineering. However, there is a substantial need for robust methods that can ease VR product development.

In the following sub-sections, we present our detailed literature reviews on the observed thrust areas like Usability and Software to understand their state-of-the-art methods in practice, determine gaps, and plan our potential research statement.

3.4 Usability Practices in Virtual Reality Products

Traditionally, VR software products are oriented towards Gaming domain [219]. Gaming companies have produced personalized games in single and multi-player format to provide real-time individual experience to players. They paved a path for creating next-level cutting edge means of entertainment. This virtual experience setup is set to transcend into Enterprise Software. However, Enterprise VR products are yet to hit the market at a full-scale due to various challenges like non-standardized HMDs, Usability, Accessibility, lack of guidelines, etc. To explore the spread of these challenges in Industry, researchers we conducted a focused empirical study to understand the underlying differences between Enterprise Software Product Development and Virtual Reality Product Development. Chapter 2 provides detailed observations from our empirical study. The study showed Usability to be one of the important qualities of VR products and its evaluation as one of contributing factors in the success or failure of VR products. Usability can be defined as the ease of use of a product where ease of use can be considered as a measurable attribute. Recent empirical research by Ulas et al. [265] has also stressed on the need for a new Software Development Methodology for building Enterprise VR products considering usability and maintainability as a significant challenge. The increased number of VR applications being developed by Industry and the need for Usability evaluation on these applications led us conduct a Systematic Literature Review in the existing literature to learn about the Usability Evaluation methods exercised in VR Product Development.

Software practitioners who practice Software Engineering (SE) principles while developing a regular software product are increasingly being used to build VR products [107]. Game developers and designers who are competent in design and strategy planning have created an ecosystem for software developers to enter into VR space [77]. Over a decade, they were able to adopt SE principles to meet their development needs. They are following development guidelines [219] and a game based development cycle [15] to improve their practices. This amalgamation of developers from gaming and SE into VR space opened new avenues and challenges for practitioners. The primary objective of some of the researchers was to *“**Understand the modalities of Virtual Reality Product Development in Software Industry**”*. Our empirical study from previous sections articulates these challenges and recorded vital insights. These insights include essential considerations on development practices, testing strategies and potential thrust areas for future research from the VR Industry community.

For a recap, following is the gist of our empirical study from previous sections on industry practitioners to record the differences between development practices specific to VR software products and non-VR software products. Some of the observations recorded from Industrial Practitioners study are given below:

- VR Product development is different from traditional Software Product development.
- VR Development process is complicated, unstructured and can be formulated based on the level of practitioners' participation.
- Design and Usability reflect VR Product sensitivities. They have a direct impact on product quality.
- There are almost no comprehensive testing strategies for VR Products to improve over multiple product releases.
- Usability Evaluation is considered to be a costlier affair as it requires a personalized setup. Most of the time they yield fluctuated results as the products tend to be persona based and hence different results are obtained from different participants.
- Design Versioning and Sustenance Maintenance are time-consuming and confusing at times for unstructured VR Product Builds.
- Support tools for VR product development practices was inadequate
- Stakeholder conflicts are far more given the wider variety of stakeholder involved in the development of VR products

Quality of Experience (QoE) is the central aspect of VR Products. As group or mob studies cannot distinguish the outcome of a personalized experience of a single participants, VR Product offerings are currently being generalized for a particular audience and are agreed upon a common ground. Usability is the underlying area which has an omnipresent impact across VR Product Development Cycle. That motivated us to understand the means of usability studies practiced in Industry while building VR applications for target audiences. Subsequently, this research can help future Industry practitioners to choose a better usability evaluation methods as per their future product needs.

Software Engineering considers usability as one of the key software quality attributes. Usability is a measure which can be employed for any product. ISO-9241-11 defines usability as *"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use"* [121]. Usability constantly reminds the product owners to consider the end user perception while designing the product. It has to be regulated at every stage of software product development so as to constantly improve the ease of usage. There are Usability Evaluation Methods (UEM) to address various end-user user-cases in a real-world scenario. However, VR practitioners are still unsure of the best available methods to conduct Usability Evaluation. In this section, we present our systematic literature review to record the usability evaluation methods suggested by different researchers while working on assessing the VR apps.

Research Questions

The Systematic literature study was performed considering the parameters defined by Kitchenham et al. [150]. The primary goal of our study is to capture the usability evaluation methods practised by various Industrial researchers while evaluating their VR products and applications as part of their research. We followed an evidence-based PICOC (Population, Intervention, Comparison, Outcome and Context) method [22] to formulate our research questions. As per PICOC method, our general concepts are presented in Table 4.1. Our work tries to address the following research questions:

R1: *What are the evaluation methods used to measure usability in VR products?*
R2: *What are the metrics captured while conducting usability evaluation in VR products?*
R3: *Is there a trend in implementing a certain usability evaluation method in regards to VR products?*
R4: *Is there anything unique about usability evaluation setup in Industry VR Products?*

Table 3.5 PICOC definitions for literature study on Usability Evaluation Methods in VR

Criterion	Description
Population	Virtual Reality related products and applications
Intervention	Usability Evaluation Methods
Comparison	Comparison between the results captured in various Usability Evaluation Methods
Outcome	Studies where Usability Evaluation Methods are applied on VR based products and applications
Context	Academia, Software Industry and Other Empirical Studies

Search Strategy

We began our literature review by identifying keywords and some search strings deduced from the research questions. The search strategy was set by a search string enabling the identification of studies that describe the execution of at least one usability evaluation method applied to a VR Industry software product. The search terms were chosen with concepts resulting from the PICOC method.

C1: “Virtual Reality” **OR** “Virtual Programming”
C2: “Usability method” **OR** “Usability technique” **OR** “Usability Engineering” **OR** “Usability Practice” **OR** “Usability Approach” **OR** “Usability Process” **OR** “Usability Test” **OR** “Usability Procedure” **OR** “Usability Study” **OR** “Usability Studies” **OR** “Usability Assessment” **OR** “Usability Evaluation” **OR** “Usability Inspection”
C3: “Industry” **OR** “Commercial” **OR** “Enterprise”
C4: “Publication Year” $>$ “2000”

The resulting string can be formulated as ‘C1’ **AND** ‘C2’ **AND** ‘C3’ **AND** ‘C4’.

While constructing the search string, we considered Virtual Reality and Virtual Programming keywords first to filter the papers from VR Space. As we are exploring Usability space, we included all possible potential keywords in regards to Usability. To identify the literature from VR Industry, we included Industry related keywords as the new level filter. In regards to the year of publication, we considered the year - 2000 as a limit to extract the literature. To the best of our knowledge, we believe there wasn't any significant work in the VR space before that. We conducted a multi-level analysis on VR research area and found that the VR technology was highly simplified after the year 2000 with the advent of new hardware. Additionally, we reviewed the search strings multiple times and incrementally developed them based on a peer-review approach. We worked with a few fellow researchers in the research area to build a robust string to generate effective results. Once the search string was finalized, the authors independently conducted the search activity and were able to record identical results. The authors have conducted a string search against all available attributes of a research paper including abstract, Contents of Paper, Keywords, etc. We filtered these attributes further to avoid miscellaneous papers.

Databases and Paper Selection

The search was conducted using the search string against electronic databases - IEEE Xplore, ACM Digital Library, Scopus and Science Direct. The search ordering was based on the databases that returned most of the results. The search fields were selected to assure that the search process is made similar across these databases. We omitted the grey literature and focused on active publications. Our review considers research papers published until August 2018.

Exclusion Criteria - Articles with poor details about the study setup on VR Products. Articles which had irrelevant information about HCI techniques, topics related to description of usability engineering or Non-Industry papers were excluded. We have also not considered papers which do not mention anything about the commercial aspect of the VR product built as part of their research.

Inclusion Criteria - The paper which contains the use of Usability Evaluation Methods as part of their title, or abstract or keywords considered. The article includes terms related to the search string. The paper consists of the study conducted in Industry or on an Enterprise Product was provided primary consideration. Only papers written in English and whose abstract or title mentions the relevance to our review considered.

Data Extraction

The process of data extraction was conducted based on the protocol proposed by Kitchenham et al. [150]. For every paper extracted based on the search criteria, we captured the attributes like Type of Paper, UniqueID, Author(s), Editor(s), Title of the paper, Pages, keywords, DOI, year of publication, ISBN, Publisher, Extraction date, Database. As part of data extraction, the search string returned ***84*** papers across all the databases. We implemented the inclusion and exclusion criteria over these research papers. We were able to filter them to ***36*** research papers which hold good for our literature review. Table 3.7 provides details of these finalized research papers.

3.4.1 Discussion on Usability Methods & Metrics for VR Products

Based on the data, we found that the Industrial researchers have seriously considered Usability Evaluation as a pointer for practical realization of their VR Product. The Table 3.7 contains the details of Usability Evaluation methods applied on VR apps categorized based on the metric and by the type of Industry. In this section, we discuss our research questions and discuss the relevant observations.

R1: *What are the evaluation methods used to measure usability in VR products?*

Widely Used Approaches - Controlled Experiments are found to be the most widely used usability evaluation approach for Industry-based VR applications. Industries like Health Care, Manufacturing, Simulation studies and Software products have considered this approach as a better way to understand the usability issues. Almost all these applications are built for training and simulation studies. The VR application developers have followed Presence Survey (discusses the mental experience of the user) and Informal Usability Survey (that do not require facilitators) to capture the participants' overall reaction post usability evaluation.

Focused Approaches - Selective domains have made use of Usability evaluation methods for a particular set of end users. Haptic centred controlled Experiments, Pluralistic Walk-through, Subjective and Objective Evaluation are implemented primarily in large-scale enterprise VR applications. Other minor methods include Psychometric Assessment, Workload Assessment and Use-case based Usability testing. Most of these methods are employed on applications like Mixed Reality based Simulator applications, Physiological Signal Sensor training applications and Venepuncture Training for Medicinal Practitioners.

Group Study Approaches - There are applications that require observations from targeted user groups. Usability evaluation methods like Focus Group Experiments, Pluralistic Walk-through, Cognitive Questionnaire and Task-based Heuristic evaluations are employed for VR Usability Testers to understand the multi-user experience in business sensitive VR applications.

Empirical Methods - Apart from conducting usability evaluation methods, VR Industrial practitioners have applied survey-based empirical approaches to capture the participant feedback to understand the impact of their usability evaluation. Standardized survey methods like NASA Task Load Index, IBM Ease of Use Survey, Social Response Scale and Systematic Usability Survey are implemented mainly on Simulation, Manufacturing and Healthcare related VR applications. The Usability practitioners have adopted these empirical approaches as part of their usability study to understand the Quality of Experience (QoE) of the VR product. All the VR products collected as part of our research followed a usability evaluation method to calculate the desired usability metrics. An empirical approach was also observed to capture the user feedback corresponding to the conducted usability study.

***R2:** What are the metrics captured while conducting usability evaluation in VR products?*

Regular Metrics - Table 3.7 contains a detailed portrayal of Usability metrics which are captured during the usability evaluation. Of these metrics, Effectiveness, Efficiency and Satisfaction are found to be the most widely captured metrics. There are few unique metrics like Inter-pupillary Distance, Physical Interaction Capacity, Bio-mechanical Feedback, Force Exerted, Galvanic Skin Response, Skin Temperature, Heart Rate and Spatial Orientation Test focused to specific VR applications which are built for handling sensitive tasks and their respective actions.

Unique Metrics - Different practitioners across various VR products calculated the same characteristic in a unique manner. Interaction Capacity is a metric which explains the capacity of a user to interact within the Virtual Environment. Practitioners have captured the user interaction in Virtual Car Interior [179] based on the time spent by a user over each task in the virtual environment. However, in a case of the tourism-based VR application [249], user interactions were captured based on navigation time across the virtual environment. Other studies include projection based applications [82] where the first person experiences were captured based on a wait and response time to calculate the interaction capacity of the participant in usability evaluation.

Novel Metrics - Efficacy is another unique metric which is being captured from a brain injury community-dwelling individuals [247]. The motive of this metric is to understand the ability of the participants to produce intended results decided by a medical practitioner. A degree of dizziness or Vertigo based metric is captured [158] to evaluate the user experience in immersive 3D Map environments. Apart from the above, the rest of the metrics are captured as per usability guidelines using survey-based empirical methods.

***R3:** Is there a trend in implementing a certain usability evaluation methods in regards to VR products?*

Shift in Approach - We find a minor trend as part of the methods and metrics employed in the published works on Industry VR products. There has been a progressive shift on using particular usability evaluation methods over a period. Notably, the training based education VR products have shifted their evaluation strategy from heuristic cognitive based methods to Group based evaluation methods. The probably reason could be due to increased levels of understanding of group learning setups. The domain-specific VR apps pertinent to Health Care and Manufacturing initially followed Controlled Experiments. Later, most of the products adopted Haptic based methods due to the significant change in sensitivity of the tasks involved in the evolving VR product.

Automation in VR - We see a slight trend in automating the usability evaluation methods across products in Industry. Industry researchers have claimed that automation helped the practitioners avoid manual biased results to some extent. However, there is no strong empirical evidence to support their argument. Business critical applications in the area of Construction [68], Health Care [247], Gesture Interaction [48] and Understanding force a certain level of automation with in their VR products. Haptic based virtual environment [234] has some automation efforts by practitioners. Spaceflight operation evaluation [83], Electronic device evaluation [29] has automated means to understanding usability challenges replacing a human-based usability evaluation approaches. A small set of distinct products in the field of Manufacturing [19], Retail Product View [142], Fashion [28] and Museums [82] adopted automated approaches to detect the usability metric and avoid human prejudice during usability evaluation.

R4: *Is there anything unique about usability evaluation setup in Industry VR Products?*

Product Liability - Industry VR Products are constrained by business decisions in line with the demands of the consumers. They are expected to provide rich set of features that are attractive to their consumers and at the same be released at the right time. Most of the products face setbacks in the Product Quality Evaluation stage. Whenever an abnormal feature flow is identified, the relevant intended behaviour requires serious inspection. The slightest change in a feature flow influences the relative design, code and work-flow of the entire VR Product. When compared to non-Industry VR products, Industry VR products are bound to be compliant as they have real-world outcomes. Any loss of data or life has severe consequences in real-world, and it is more concerning for Manufacturing and Healthcare. Most of these applications are involved in training and simplifying day-to-day activities.

Content Development - The results from Usability Evaluation provide more options to VR product stakeholders to enrich the product as needed. Every usability testing cycle requires review from the entire VR Product development teams to compare and critique the change implemented to address the usability issue. Change in feature flow and task-actions drastically impact original design and requires upgradation of the VR content to meet usability requirements. Content Development is comparatively expensive as it requires multi-level design strategy across all the VR Scenes. It is unique for Industry practitioners to comply and also have sanity in content design across the VR Product to provide a quality user experience.

Product Versioning - Industry VR product is poised to stay longer in the market as it is expected to generate revenue. Thus it is common to adopt product versioning methods which are unique to Industry VR Products. It is evident that it is complicated to manage and maintain design versions of VR Product when compared with code-based version repositories. Code repositories are robust and mature when compared with design version repositories. Unlike code repositories, where structure of the artifact is well know, design artifacts involved in VR products, for example, a VR scene design does not have a well established representation. Hence, support for versioning must be looked at from multiple perspectives. Commercial VR Products in Fashion [28], Simulation [290] and HRMS [301] have involved designers in maintaining two or more major design threads such that the design layouts are dynamically altered based on the target audience base.

Effort Estimation - VR Products a relatively recent phenomenon and developers tend to involve a variety of people who are unique to VR production. In contrast to traditional Software Product Development, VR teams include practitioners like VR Scene Designers, Acoustic Editors, Audio/Visual Developers, VR Scene Artists, Content Editors and Integration Specialists who are unique to VR product development. Usability Evaluation results create new work-load across these stake-holders which indicates a rise in product development cost regarding stake-holder effort. None of the VR Products measured initial effort estimation to plan for future releases. It is significant for a VR Product to have an effort estimation strategy to minimize the future product version cost.

Usability Guidelines - We found that guidelines are vital for constructing a usability evaluation of VR Products for focused domains like healthcare and simulation-based Industries. Usability Guideline based Usability Evaluation was found to be in health care products while capturing metrics like Reaction Time, Response Time, and Interaction Capacity. Product developers built VR products based on Usability Guidelines, and the VR Usability practitioners have evaluated the VR Scene based on the compliance of the applied guideline. Practitioners have transformed this activity as a validation experiment under the Usability Evaluation setup. The Simulation-based VR Applications [83] [19] [247] utilized guideline-based evaluation in most of the cases.

Observations

Fig 3.4.1 illustrates the yearly trend of adoption of Usability Evaluation methods on Industrial VR Products over a decade. Based on these results, we observe that focused applications require focused Usability Evaluation methods like Haptic based techniques, Heuristic Reviews, and Psychometric Assessment. These results of these methods have a significant impact on the workflow of the VR Scene. Most of these methods are followed in HealthCare and Simulation-based studies. We also observe that Controlled Experiments and Cognitive Walkthrough are two widely used methods across the years. These two methods are implemented across various Industry domains, thus explains to be a distinct

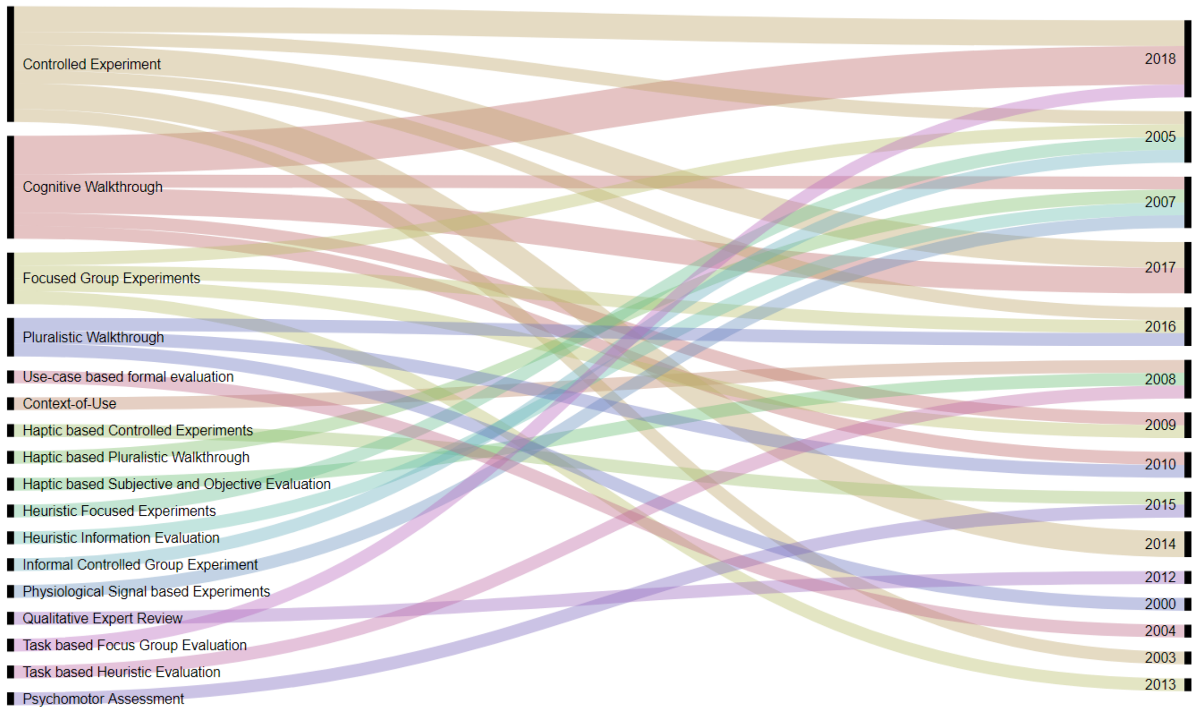


Figure 3.4 Yearly Trend of Usability Evaluation method Adoption in VR Software Development

usability study method for quick results. Regarding the existing literature, we could reason that both Controlled Experiments and Cognitive Walkthrough approaches provide the following benefits for every VR Scene.

- Controlled Experiments and Cognitive Walkthrough are easy to set up and execute.
- These methods yield prompt results and help the practitioners deduce quick inferences.
- They do not involve in complex workflows.
- It is accessible and easy for the test subjects to participate in the experiment.
- They are compact and flexible with the changes to the requirements of a VR Scene.

Constructing a Simple Evaluation - Based on our review, we perceived that some of the practitioners had not followed a particular protocol to execute a Usability Evaluation using a specified test setup. However, some researchers were clear and strict about the workflow of their study setup. Cases where the motivations of the VR Scene were clear, usability evaluation was performed over all stages of VR Scene development. We gathered such instances and have come up with a simple protocol to conduct or establish as Usability Evaluation Setup for a VR Scene.

- *Define Motive* - The Usability Practitioner has to define the motivation against VR Scene design. These motives should be in line with requirements specified for a VR Scene. One has to ensure that the evaluation should not alter the initial specifications but should facilitate a way to meet those specifications. Practitioners [301] [28] were clear about their motives and have cascaded the same during their entire product development, which aided them in generating better test results.

- *Identify Test Subjects* - The VR Product should have a purpose and should be developed with the intent to deliver a solution to the target audience. The participants of the Usability Evaluation Setup should be the sub-set of the respective target audience. The practitioners should expect 360-degree feedback on the VR Scene and should be able to revisit the specifications if the usability is not up to the mark. Practitioners, as part of focused application [247] [79] [25], have ensured that their target audience is from the right domain to have efficiently gather results.
- *Define Tasks and Actions* - The Usability Practitioners should be able to define the workflow of the test setup. In the case of a Car Interior VR Scene [179], the practitioners have set the flow of events for a test subject to perform. The Tasks are clearly defined in a step-by-step fashion so that the participant is clear about the next steps in the evaluation. The Practitioner should be able to gather the actions out of the tasks performed by the participant and later conclude those actions.
- *Design Experiment* - The Design of the Experiment should be inline with the scope of the VR Scene. [83] [147] [276] have developed a methods to design the experiment for conducting simulation studies. Practitioners have to define the cause using the independent and dependent variables of a VR Scene. This intent results in the effects out of the experiment for a respective cause.
- *Review and Redo* - After gathering the results, the practitioners have to evaluate the data pointers from used metrics and correlate them with their expectations - the Expected Vs. Actuals should be reviewed with the VR Scene developers. The product owners will have to work on the caveats identified as part of the studies and should be able to redo the VR Scene development. Practitioners [293] [82] here have involved their developers in reviewing the test results so that they can plan to revisit the Scene and address the challenges from time-to-time.

Scope of Automation

In our review, we observe that the demand for usability has increased [137]. It is due to the rise in the reach of VR Products to end-users and the novelty involved with them. Now-a-day, Usability Evaluation is being executed even after the first sprint of a VR Scene development. In such a state, automation can assist practitioners in having to minimize the effort and time while performing Usability Evaluation. Recent Studies on automated usability evaluation have appeared [110], where a fully automated approach has been developed to perform predefined tasks in a fixed test setting. Here task trees are generated that records the actual usage of the scene during the Scene development. Later the recording will be used to analyze the usability smells. The defined approach is applied to a large case study and has recorded diverse issues related to user efficiency. We may argue that the proposed approach [110] may help practitioners identify usability issues but may not be able to specify the misunderstandings of users from a different test population. It is considered to be a challenge for usability practitioners to automate. However, a semi-automated usability test setup seems promising for Usability evaluation as human perception is difficult to be automated. A task-driven automation method for VR developers and a participant based evaluation for VR Tests followed by a developer-designer review may help capture observations from all directions.

3.4.2 Sufficiency & Necessity of Usability Evaluation of VR Software

This work intrigued us to explore and discuss the levels of necessity and sufficiency required for conducting a Usability Evaluation in VR products. Notably, in regards to VR Products - Usability Evaluation is a costlier activity [137] when compared with the rest of the tasks involved in VR Product Release. While working on VR products, Usability practitioners should examine the testing strategy by classifying the available usability metrics and usability evaluation methods into necessary and sufficient categories. The necessary attribute is an attribute that must be present for an event to occur may include metrics like Response Time, Attention, Reaction Time, User Satisfaction and Effectiveness. These metrics may help determine usability at a practical level. These metrics appear to be most reasonable for any VR based product(s). In regards to usability evaluation methods, Controlled Experiments and Cognitive Walkthrough seem to be reasonable and widely used methods for sensible results. Sufficiency here is an attribute of being adequate or sufficient for conducting the usability evaluation. Visual Feedback may be a useful usability metric as a sufficient attribute. Task-based Focused Group Study may be an ideal sufficiency usability evaluation method. The above observations are based on our review. Usability Practitioners may choose customized usability metrics and customized usability evaluation methods for focused results.

3.5 Software Quality Metrics for Virtual Reality Products

ISO Standard 9126 [84] quality model classifies quality into a collection of quality characteristics and sub-characteristics. Various metrics can be used to measure these quality characteristics. Product owners have to adopt diverse software quality metrics to track the health of the software product after every product release. Software quality of an enterprise software product involves quality assessment, quality assurance, and quality evaluation. Researchers have proposed various approaches and metrics to address software quality problems at different stages of software production. Shihab et al. [246] conducted a literature review of more than 100 published research papers on software defect prediction. It was found that most of the approaches did not provide guidance on industrial adoption and rarely considered the impact, risk, and dependency associated with the predicted or forecasted defects. The practical adoption of software quality methods in the industry is limited as software industry tends to be reactive. Compared to traditional software developers, VR practitioners have fairly limited knowledge about the state-of-art metrics needed for quality management of VR products [137]. In this section, we detail a systematic mapping study performed on existing VR literature to explore software quality

Table 3.6 PICOC definitions for literature study on Software Quality metrics in VR

Criterion	Description
Population	Virtual Reality related software products and applications
Intervention	Software quality metrics or indicators
Comparison	Comparison between the results captured in various software quality metrics
Outcome	Studies where software quality metrics/indicators are applied to VR product and apps
Context	Academia, software industry and other empirical studies

metrics or indicators used by VR developers while developing VR products.

Research Questions

The Systematic Mapping Study uses the guidelines suggested by Petersen et al. [204]. The primary goal of the study is to capture the details of software quality indicators or metrics adopted by VR developers while developing their respective VR Products. We followed an evidence-based approach called PICOC method. PICOC is an acronym for Population, Intervention, Comparison, Outcome, and Context [22]. Application of the PICOC concepts to VR products study is shown in Table 4.1. PICOC method helped formulate the research questions effectively and document the scope of mapping study. Following are research questions:

R1: *What are the existing software quality metrics/indicators used as part of VR product(s)/app(s)?*
R2: *Is there any trend in adapting certain software quality metrics/indicator in VR product(s)/app(s)?*
R3: *Are there any domain specific VR Software Quality metrics?*

Search Strategy

For any systematic mapping study establishing a search string is key. As a first step, keywords pertinent to Software Quality metrics in VR applications were identified. The research questions R1, R2, and R3 are related to each other and hence only one search string was constructed. The search strategy was set to enable identification of studies that describe the presence of at least one software quality metric or indicator applied to VR Software Product.

*Search String: The search terms were chosen with concepts resulting from the PICOC method. Below are the details of search strings.

C1: “Virtual Reality” **OR** “Virtual Programming” **OR** “Virtual Reality Product” **OR** “Virtual Learning” **OR** “VR” **OR** “Virtual Environment”
C2: “Software Quality” **OR** “Software Metrics” **OR** “Software Indicators” **OR** “Software Evaluation” **OR** “Metric” **OR** “Metrics” **OR** “Indicator” **OR** “Indicators” **OR** “Quality Assessment” **OR** “Quality Improvement” **OR** “Quality Evaluation” **OR** “Quality Measurement”
C3: “Publication year” $>$ “2000”

The resulting string formulated for addressing research questions R1, R2, and R3 is ‘C1’**AND** ‘C2’ **AND** ‘C3’. We considered Virtual Reality and Virtual Programming related keywords as the initial search filter. Software Quality Metrics is a major factor, hence we expanded the search space to consider all potential keywords pertaining to quality metrics. We conducted a multi-level analysis [150] on the Virtual Reality research area and found that with the advent of new hardware there was a significant shift in VR technology and corresponding software after the year 2000. Hence, the year 2000 was considered as a limit for publication year to extract the literature.

*Search Quality Assessment: We reviewed the search strings multiple times and incrementally developed them based on a peer-review approach [150]. We also conducted a manual search of the search string incrementally and compared the results in every validation cycle. Our peer-reviewers graded the manual search validation results and helped us finalize the search string. We worked with fellow researchers from similar research areas to our finalize the search string. Once the search string was finalized, the authors independently conducted the search activity against all available attributes of a research paper including abstract, contents of the paper, keywords, etc. and recorded the respective results. We filtered these attributes further to avoid miscellaneous research papers. Our review supplement data can be found here [181].

Databases and Paper Selection

We conducted this search activity against major electronic research databases including IEEE Xplore, ACM Digital Library, Springer, and ScienceDirect. The search order was based on the databases that returned most results. The search fields and search string was formulated to assure that the search process is made similar across these electronic research databases. We omitted the grey literature (research produced by organizations outside of the traditional commercial or academic publishing and distribution channels) and focused only on active publications. Our review considers research papers published till September 2020.

Exclusion Criteria - Articles short on metric description details were ignored. Research papers with no clarity on their VR product setup were ignored as it is critical for judging the quality metric used in a

VR scene. Articles that focused only on software quality processes/models/techniques, topics related to the description of software quality engineering, or industry white papers were excluded from our study. Also, papers that did not mention anything about the quality aspect of the VR product built as part of their research was not considered. Books were not included as part of this mapping study as books tend to cover broader data that is more useful for in-depth analysis rather than mapping study.

Inclusion Criteria - Papers that discussed the use or implementation of a Software Quality Metric(s) or Indicator(s) in their title, abstract, or keywords are considered. Peer reviewed publications that contain clear details about the VR products and documents were given primary consideration. Only articles written in English were considered as part of our study.

Data Extraction

Data was extracted using the process proposed by Kitchenham et al. [150]. For every paper extracted using the search strategy, attributes like type of paper, uniqueID, author(s), editor(s), title of the paper, pages, keywords, DOI, year of publication, ISBN, publisher, extraction date, database type from the research database was extracted. The search string returned ***491*** research papers across all the databases. Inclusion and exclusion criteria was applied to these research papers to filter down the results to ***227*** research papers. Upon further review, ***77*** research papers were found to have used at least one software quality metric/indicator with a reasonable explanation and some mechanism to gauge the metric value. Figure 3.5 provides a grouping of the quality indicators identified based on indicator type. Some of the quality indicators could have been classified in more than one category, however to keep it simple we placed them in a single category based on our own experience in software quality engineering. The details of our mapping study analysis from various research databases are made available in our resources [181]. This resource contains step by step evaluation and filtration of research papers conducted over a period of time.

3.5.1 Discussion on Software Quality Metrics, Usage Trend across Field-of-Interests for VR Products

Quality is not orthogonal to product development, it involves the usage of various validation mechanisms and corresponding measurable attributes through various stages of Product Development. Hence the software quality indicators identified span the entire spectrum of VR product development. In the review process, we came across several papers where researchers used software quality metrics for validating their VR Prototype or Product.

Based on the review of literature, we conducted a coding based thematic analysis [59] of the gathered research papers. Based on the coding results, we broadly categorized the gathered data into six themes: *VR Audio Quality*, *VR Scene Quality*, *VR Video Quality*, *VR QoE (Quality of Experience)*, *VR Image Quality* and *VR Code Quality*. In this section, we analyze the research questions and discuss the relevant observations.

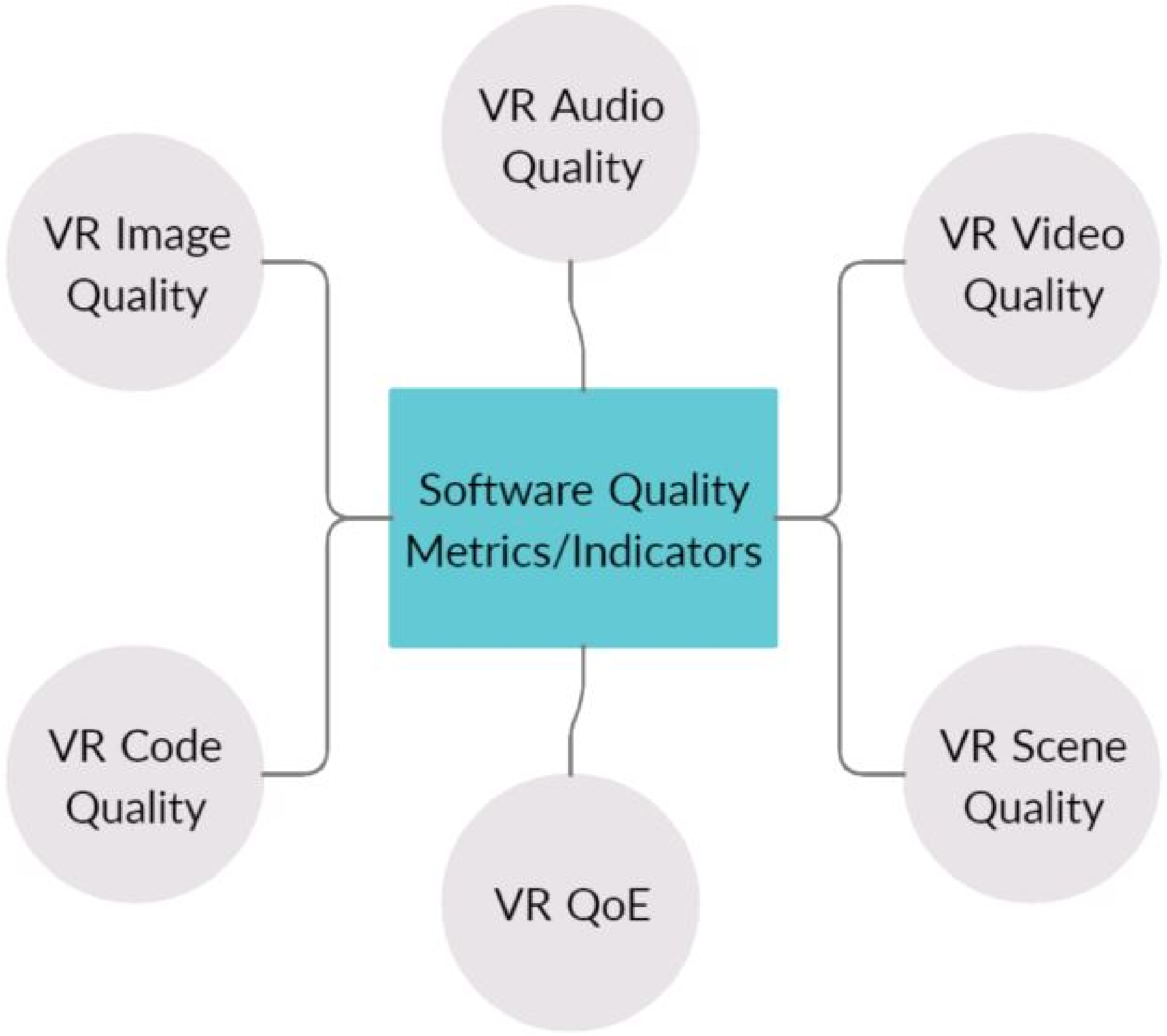


Figure 3.5 Software Quality Spread in VR

R1: *What are the existing software quality metrics/indicators used as part of VR product(s)/app(s)?*

Figure 3.5 presents the software quality themes in the context of VR Product Development. The discussion in this section is based on the observed themes.

VR Code Quality: Ghrairi et al. [93] were the first to directly analyze queries of VR practitioners from a Software Engineering perspective using the StackOverflow forum. They provided insights on various practitioners' challenges on day-to-day VR product development. They also conducted an exploratory study on the code quality of open-source VR projects from GitHub. They observed that most of the VR projects had weak-level of comments, almost all the projects had at least one God Class and other code smells. They also found the dominance of open-source VR Engines in Open Source VR Projects. Source Line of Code (SLOC), Cyclomatic Complexity (CC), number of methods per class, and the Depth of the Inheritance Tree (DIT) are the most widely used quality metrics by the VR practitioners. In effect, there were no specific metrics for VR code quality.

VR Video Quality: Jiachen et al. [288] conducted a subjective evaluation of 3D Panoramic Virtual Reality Video Quality Assessment using 3D Convolutional Neural Networks. They conducted an empirical study on 3D Panoramic VR Video Dataset and determined the initial quality of these videos using a subjective score for each data sample. They proposed a 'fusion strategy score' to rank and determine the quality of the VR Video project process. They considered MultimediaQA, VRVideoQA, and ImageQA indexes as a measure for the quality of their assessment. Their future plans include the application of their proposed score on large VR video databases to help VR Practitioners adopt the proposed score and metrics as part of conventional VR product development. Alireza et al. [296] conducted a quality assessment comparative study between tile-based method and truncated square pyramid (TSP) method of projecting VR Videos. These methods are primarily involved in Streaming VR Videos which are subject to latency issues. Quality-Assessment-View (QAV) Index was used to assess the quality of VR Video. Subjective evaluation was performed and the observed data was analyzed to determine the merits and demerits of these methods. Sijia Chen et al. [51] reviewed Omnidirectional VR Videos and conducted an objective evaluation to calculate the spherical structural similarity index (SSSI) and compare this quality metric results with traditional heuristic methods. They also conducted an experimental assessment to determine the relationship between two domains to determine the video quality.

Sebastian et al. [240] made attempts to understand the methods to render virtual viewports from supplementary depth information to create a better VR video quality. Depth-image-based rendering and Peak-Signal-to-noise ratio are two quality metrics used to evaluate the VR video quality. They conducted a subjective evaluation and published their findings. Naty et al. [248] conducted an experimental study on understanding the performance and computational complexity of 360 degrees VR Video. They were part of a research group that developed new coding tools to address video encoding and decoding to avoid noise and a better bit rate in VR Videos. They have used Weighted-Peak-Signal-to-Noise Ratio as a custom metric to evaluate the 360 degrees VR Video quality. Shu Yang et al. have introduced a quality assessment method for panoramic videos which is based on multi-level quality factors. This is calculated based on the region of interest in a given VR Scene [289]. They conducted a subjective valuation using a few panoramic scenes and captured insightful results. Their observations shows that the quality assessment method is easy to implement, when compared with traditional video quality assessment method. Carlos et al. proposed two novel metrics to study the user behavior under 360-degree movie cuts [173]. This is to examine the influence of user perception on a 360 degree movie cuts over large scale video scenes.

VR Image Quality: Wei et al. [258] conducted a subjective quality valuation of compressed virtual reality images in VR scenes. They performed a correlation study with popular objective quality measures and published their observations. They used Single-Stimulus method to collect the subjective scores from participants and have computed the MultiScale Structural Similarity Index (MSSI) to determine the quality of the images in a VR Scene. Huiyu Duan et al. [69] established an exhaustive VR Image dataset and worked on perceptual quality assessment of Omnidirectional images in VR. The image quality assessment measures were applied on their dataset and suggestions were provided on the improvement of images based on VR Scene setup. Junfei Qiao et al. [212] reviewed 3D Synthesized views used for the rapid development of high-quality VR Scenes. They formulated an algorithm as part of their previous work and compared it with their existing dedicated No-Reference Image Quality Assessment (NIQA) method to assess the quality of the Image in a high-quality VR Scene. They set up an experimental study to comprehend image quality degradation and distortions. Further, they have recommended guidelines for low-level compression of images in VR Scene to avoid rendering issues. Rahim et al. [215] introduced a content-dependent objective quality assessment procedure to evaluate the distortions that occur while building the viewport in VR Scenes. They used a supervised learning method to classify their dataset to determine the viewport quality of 360 degrees images in a given VR Scene. They set up an experiment to predict the proposed metric against the viewport quality of the image set with reasonable accuracy.

VR Audio Quality: Miroslaw et al. have reviewed issues with spatial audio as part of their previous work and have now reviewed the quality aspects of compressed audio on emerging HMDs [188]. They adopted the MUSHRA test methodology (Multiple Stimuli with Hidden Reference and Anchor) to assess the quality of soundscape of a 360 degree streaming VR for immersion setup. They conducted a subjective evaluation of raw audio content and captured the quality of user experience. They reviewed the options of compressing the spatial audio and proposed a few guidelines for practitioners. They plan to develop an objective spatial audio quality metric as part of their future work. Jules et al. [85] were the first to study continuous movement recognition and real-time sound parameter generation in a VR Scene. They conducted series of experiments to understand the mapping between the design process and user interaction through a VR Scene using machine learning approach. They used Auditory Feedback as a quality measure to determine the health of their study. David Triantafyllou et al. [264] are pioneers on studying sound in VR scenes. They conducted two experiments to determine the relationship and shortcomings between physical-world sound and VR Scene sound. They identified the difference between these two experimental conditions and used auditory feedback to assess the quality. They have published a few guidelines to practitioners on building combinations of surfaces with better natural sounds in virtual environments. Ceenu et al. [90] investigated whether audio signals and haptic feedback can act as indicators for real-world boundaries, such as objects, walls, and people. They used NASA TLX Survey and Presence questionnaire to gather feedback on presence and workload from the participants in their study setup. Adrielle et al. studied Audio localization in a VR Scene using spatial audio metrics like the NASA TLX questionnaire to capture workload while performing actions like gaze pointing and wand pointing towards the projected audio in VR Scene [182].

VR Scene Quality: Blaine Bell et al. [33] are first to investigate the quality of rendered VR plane. As part of their work, authors focused on view management in a 3D view plan and determine the properties of objects, position, size, transparency, and shapes of the virtual world. They proposed a layout decision approach and conducted a subjective evaluation. Further, they proposed a custom quality metric to determine the view plan representation in a virtual world. Ying Zhang et al. [294] created a multi-model interface for the virtual environment for an assembly application. They evaluated multi-model feedback on assembly task activities through simulation in the virtual world. They were the first to build such an interface and conduct a heuristic auditory and sensory feedback study to assess the quality of their environment. They also conducted a subjective evaluation and captured the observations via a questionnaire. They provided recommendations to the practitioners on building an efficient task performance based VR Scene. Shun Li et al. worked on a virtual surgical simulation setup with an intent to understand the thermal damage of a bone tissue using bone drilling [160]. They formulated a virtual surgical process and evaluated the virtual scenes using a customized quality metric called 'Temperature Distribution'. Dongdong et al. explored the differences of visual discomfort caused by long-term immersion between virtual environments and physical environments [103] across a variety of VR scenes. Visual fatigue scale (VFS), change of pupil size (PS), and accommodation response (ACR) are captured as metrics. Bilal et al. conducted a task-based subjective evaluation of a driving simulation [223] using the user-experience questionnaire and Gaze interaction metrics.

Carvalheiro et al. [49] proposed a haptic based interaction system for virtual reality products. This includes a combination of tracking devices for hands and objects in a given scene. Their solution receives haptic stimuli by manipulating real objects mapped to virtual objects. They conducted a subjective evaluation via an experiment setup and proposed a quality metric called 'Simulation Awareness' to understand the stimuli experience of the end-users in a virtual world. Rohan et al. [50] presented a novel algorithmic framework to optimize the depth of camera placements for a given virtual environment. As part of the study, they utilized a quality metric called simulated annealing and depth inaccuracy to evaluate the quality of the construction of the scene in virtual space. In [148], the author investigated a unique visual comfort model for the real prediction of a 3D VR Scene, which is based on the physiologic mechanism. They conducted an experimental study to evaluate r model by utilizing a customized quality metric called multimodel interactive continuous scoring. This helped them understand the stability and perception of 3D VR Scene images. Jann et al. conducted a VR Scene tolerance study using a custom scale called [88] Cybersickness Susceptibility Questionnaire. The scope of this metric is to predict the Scene tolerance of the participant.

Hak et al. [146] proposed a quality metric of exceptional motion in VR Video Contents for VR sickness assessment. Their metric was developed to improve the quality of VR scenes to avoid sickness issues. They validated the work using Simulator sickness questionnaires in VR environments. Viktorija et al. studied Levitation Simulator using VR based shooting scene to capture workload using the NASA TLX Task-based survey [198]. Jeremy et al. proposed an innovative method to navigate into VR Scenes by opting for acceleration parameters from the users in real-time [205]. The motivation was to address VR Sickness issues. They have conducted a case study and used customized quality metrics called Motion-Sickness Dose Value and Electro-dermal activity to judge their results. Ke Gu et al. proposed a novel referenceless quality metric for depth-image based rendering of VR Scenes [98]. This is to ensure that the synthesized free-viewpoint videos are generated with higher accuracy and with less geometric distortion. An experiment was conducted to review the quality method and was compared with traditional models. Yingxue et al. worked on improving the immersive viewing experience for VR Scenes [298]. They proposed a display protocol and evaluated it against panoramic VR Scene videos to review the distortions and video coding compression. They conducted a case study on all these VR Scenes and heuristically reviewed the quality of the scenes using the mean opinion score method from the participants. Deba et al. proposed a method for defining a quality metric on context-aware intelligent environments with inference to address physiological processes [229]. Their work was focused on VR Scenes which requires a visual feedback model, and developed a Supermarket application for validation. They used the electrodermal activity as a quality metric and conducted a tasked based experiment to assess their proposed method.

Brendean John et al. [130] worked towards investigating pupillary light response in the virtual scene setup. They used a custom quality metric called pupil light response as a reference scale to detect and identify the rates of cognitive-emotional responses from the participants who were part of the case study. They built a task-based activity experiment and proposed guidelines for building effective scenes with reasonable pupillary interaction. Markus Wirth et al. investigated interaction techniques in a virtual reality scene setup focused on a diagnostic radiology application. This is first of its kind domain-specific VR Application where a VR Scene based quality metric was evaluated from Software Engineering perspective [282]. Attractiveness, Pragmatic/Hedonic Quality attributes were evaluated as part of a thorough experiment for radiologists.

VR QoE (Quality of Experience): Mapar et al. [172] and Akpan et al. [14] adopted a heuristic-based personalized quality metric to evaluate a space flight simulation setup and an assembly product. They conducted a study to determine the quality of experience and used egocentric depth perception as a metric [131]. Jarvinen et al. [125] conducted a series of experiments on the spatial setup in VR scenes and evaluated their experiment VR scene to evaluate the quality of experience in spatial memory. They used a customized test called Spatial memory test to gauge the quality. Ruddle et al. [228] conducted a VR scene evaluating using travel time, collision index, and speed profile index as the quality of experience in their case study VR Scenes. Markus et al. [281] assessed the personality traits of athletes using a

VR football game scene using the Presence Questionnaire. Monthir et al. compared Game-pad and Naturally-mapped Controller Effects on Perceived Virtual Reality Experiences [17] and captured customized metrics Self-reported Presence, Self-reported Engagement, and Self-reported Accuracy. Peng et al. [202] conduct a comparative study between a PC and VR based presence evaluation of emotional challenge-based games.

Lugrin et al. [166] conducted a subjective evaluation of a task-based assembly activity to analyze the quality of experience of a participant through quality metrics In-game performance index, In-game navigation, Multi-Screen usage index to determine the adaptable of VR Scene. Charles et al. [208] built a workstation in a VR environment to understand the physical risk factors in humans. The quality of the experience was evaluated in a subjective evaluation using rapid upper limb assessment, averaged muscle activations, Total task time as quality indicators. Hamam et al. [105] built a game to analyze the quality of user experience in a VR scene by capturing User State Measure, perception measure and, physiological measure as quality indicators and shared significant insights. Steinicke et al. [254] built a simulation system to study the presence of a participant using a presence evaluation method. Aristidou et al. [20] investigated the motion capture technology for virtual spaces with a specific focus on folk dance motion analysis. They present a framework to identify the styles in dance motion and use Laban Movement Analysis to study the quality of experience of the generated Scene. Gayathri et al. studied spatial memory w.r.t heights in adults and teens using a Virtual staircase [187]. Metrics like Turning error, Latency, and Corsi Scores are captured as a case study. Precision and Task Completion time are captured along with a Subject Questionnaire to evaluate collaborative tasks in VR [87].

Additionally, various other quality metrics are implemented on prototypes for focused applications through a subjective evaluation to understand the quality of experience. They include metrics like Interaction Modality [196], Locomotion Index [41] [86], Haptic based systems include Realism, control, interface quality, ability to examine, performance, and haptic subscales [24], Speech, and language-based therapy applications used Electrodermal Activity, Heart Rate, Miss Clicks [143], Rehabilitation Game employed Hand Movement Velocity as a customized metric [170]. Heuristic and Survey based evaluations were conducted as part of studies [116], [199], [270], [227] where metrics like perceptual ration, estimated path length, recall time and immersion score were used as metrics. Andreea et al. conducted a quality evaluation of the effect of thermal visual representation on users grasping Interaction in Virtual Reality application [38]. A novel metric called Grasp Aperture was presented and used as part of the study.

The VR Applications which are focused on healthcare can broadly be categorized as detection applications and intervention applications. Given that VR is still nascent, studies have focused more on detection rather than intervention. Metrics like Electrocardiograph signal, galvanic skin response, blood volume pressure, electrodermal activity [232] [144], Flow Experience Analysis for task-based activities [118], Miss Rate, Merge Rate, Fragmentation Rate in case of rehabilitation and trauma-based VR applications [213], Effect of Reach-ability [70], Vibrotactile Feedback [192], Interface quality, and Realism Index along with an immersive tendency survey [241], Heart Rate, and Skin Conductivity [73] are used to assess the quality of applications. In [271], PPG signals and EEG signals are captured to study the participant's attention in terms of learning VR applications. This study was conducted to understand the multi-dimensional physiological characteristics of the participants towards immersed learning. Elena et al. were the first to study anti-stress adaptation to a new educational environment for foreign students [261] using a VR tool called Emotional Experience designer. This tool captures emotional and muscle relaxation as quality metrics. Multi-user Isness medical condition experiences were studied using a discussion-based questionnaire [94].

VR applications which are focused on providing high quality of experience through enhanced bandwidth use interesting approaches. Krogfoss et al. studied the impact of 5G bandwidth and its impact on improving the VR Audio and Video quality. They conducted a QoE assessment with customized metrics like content resolution (cod – coding), start delay (s), and stalling ratio (t) to assess the experience of a VR scene using 5G bandwidth [153]. Robertro et al. developed a two-stage method to enable efficient Streaming via QoE-aware Mobile Networks for AR/VR Scenes [260]. The proposed Dynamic path selection approach uses a new QoE model for video streaming. Pupillometry based metric was proposed by Kenya et al. as a Method to evaluate Reading Comprehension in VR-based Educational Comics [231]. Maria et al. Human-Centered QoE approach was adapted to assessing Delay, Jitter and Packet Loss in VR Applications [266].

R2: *Is there any trend in adapting certain software quality metrics/indicator in VR product(s)/app(s)?*

As part of our study, we observed that metrics that can be considered as a common set used across VR software applications, are fairly limited. Most of the quality indicators or metrics for VR applications are limited to particular usage and need. The trend seemed to be more towards researchers opting for customized quality metrics and building a methodology around the quality metric with a subjective or objective evaluation. Most of these customized metrics are unique and are found to be used as part of focused VR applications like rehabilitation, trauma, education, and fun task-based prototypes. Metrics like Temperature distribution [160], Electrodermal Activity, Heart Rate, Miss Clicks [143], Hand Movement Velocity [170], and Pragmatic and Hedonic Quality [282] are widely used health care based application over a decade. Other metrics like Rapid upper limb assessment, averaged muscle activations, Total Task Time [208], Heuristic Evaluation [172] [14], Auditory and Sensory Feedback [294], Immersion Score [270] are distinctively followed in the manufacturing domain over a decade. Due to distinction in scene design across various domains, no traces of a generic software metric(s) or indicator(s) are found to be practiced.

In some cases [90], researchers relied on a common quality indicator for multiple quality factors like workload and presence. It shows that there is a need for new approaches for use-case based adoption. It clearly shows us that the practitioner's attitude towards adopting a software quality metric is unique and varies based on the intent of the scene. Most of the practitioners have not considered at least one metric in common to address essential quality requirements. These observations motivate the need for a **Software Quality Evaluation framework** for VR applications, which includes strategies for addressing Image, Video, Code, and Audio quality challenges in a generic way. Such frameworks can be realizable if and only if multiple empirical studies are attempted on large VR product data sets. This will help future VR practitioners to adopt generalized quality metrics as a basis and then develop focused quality metrics on top of the framework, based on their business needs. We plan to explore this avenue as part of our future work and will attempt to work towards formulating a generalized Software Quality Evaluation framework for VR products.

Scope for Automation - As part of our review study, it was surprising to note that practitioners or VR product(s) are not using automated methods or metrics to assess software quality. Developers are making progress in building frameworks/tools that can be used for VR testing. While most of the VR Quality metrics are defined to achieve Quality of Experience, a large number of these metrics are being evaluated manually. Metrics like lines of code [93], lack of cohesion [93], Omnidirectional IQA [69], and depth image-based rendering [98] are currently captured using semi-automated methods. New automated methods as well as a moving current semi-automated methods to fully automated methods can shift the make the assessment more objective. There is tremendous scope for developing automated approaches to assess/improve VR quality. For example, in VR simulation software, Interface testing software, etc. automated assessment of various qualities can significantly help in making the applications better.

***R3:** Are there any domain specific VR Software Quality metrics?*

Table 3.8 illustrates the details of the metrics used as part of VR product quality assessment across specific Quality Themes (or) Domain. Domain - It is not a type of industry or business but a common theme under which a VR product is developed. To categorize, we consider VR applications that are involved in assembling the objects in a specific order that may come under the *Assembly* domain. VR Apps which require the users to perform tasks and generate actions in the given VR scene are regarded as *Task-Action* domain. VR Apps based on games for fun or serious games are acknowledged as *Gaming* Domain. VR Apps, which provides health care solutions, comes under *Healthcare* Domain. The categorization of domain here is not specific to a business need, but it is a heuristic set portrayed as a domain. To present the metrics in Table 3.8 in a well-defined form, we categorized them into below types.

Widely Used Metrics: Heuristic Evaluation with Presence Survey is the most widely used quality metric by most of the simulation-based VR applications [172] [298] [202] [14]. Auditory Feedback is another metric is used by practitioners to assess VR audio quality [85] [264]. Apart from these two approaches, the rest appear to be either distinct to a particular VR product or not relevant for generic usage. QoE based metrics like Jitter, Delay, and Packet Loss were captured using Subjective Evaluation [266].

Unique Metrics: We observed a few quality metrics which are only one of its kind; unlike anything else. Flow Experience Analysis [118] is used in the Task-based quality assessment methods. This metric can be customed to a specific scale and updated based on the practitioner's strategy. User state measure [105], Realism Index [24], Spatial Memory Test [125], Simulation Awareness [49], and Laban Movement Analysis [20] are the quality metrics which are unique on gathering results and their evaluation. In one case, researchers used the Presence Survey and NASA TLX Survey for capturing quality data for both Audio and Haptic feedback, which was uncommon in the case of other works [90]. Visual fatigue scale (VFS), Change of pupil size (PS), and Accommodation response (ACR) are captured to evaluate Fatigue Rate due to prolonged Immersion in a large VR Scene [103]. A Gender-based case study was conducted to understand VR Scene tolerance [88] by introducing a metric called Cybersickness Susceptibility Questionnaire.

Novel Metrics: We observed a few novel metrics which are not found to be used in traditional software product development. Effect of Reachability [70], Pupillary Light Response [130], Miss Rate - Merge Rate - Fragmentation Rate [213], Temperature Distribution [160] and Vibrotactile Feedback [192] are few of the metrics. We observe that these metrics are novel and can be adopted by upcoming VR products that are planning to focus on Quality-of-Experience and VR scene quality. New QoE metrics content resolution (cod – coding), start delay (s), and stalling ratio (t) to assess the experience of a VR scene using 5G bandwidth [153]. Frames to reach a ROI (framesToROI) and Percentage of total fixations inside the ROI (percFixInside) are new metrics developed to study user behavior in a 360-degree video scene [173]. Pupillometry based metric like capturing pupil size and dilation time were captured to study reading comprehension in Educational comics [231]. Customized Metrics like Self-reported Presence, Self-reported Engagement, and Self-reported Accuracy [17] are captured in comparison studies. Grasp in VR is studied through the Grasp Aperture metric [38].

Empirical Metric Evaluation Methods: The domain-specific VR applications relied on empirical approaches to gather the metric data. Question Survey [93] [143], Subjective Survey [288] [172], Presence Survey [41] , and Presence Questionnaire [24] [90] [281] have primarily used as part of *Task-Action* based VR applications. Mean Opinion Scores [298], Comparative Analysis [212] and Case Study [205] [296] [208] [69] [50] [98] [160] [248] [229] [144] [105] [213] [88] [94] [130] based empirical methods are practiced in *Gaming* based VR applications. Experimental Setup [131] [232] [118] [14] [228] [116] [70] [199] [49] [294] [146] [270] [73] [125] [192] [33] [85] [264] [282] [20] [166] [87] [261] [223], Temporal Visual Comfort Model [148], Kinect skeletal model [170], Kennedy-Lane Simulator Sickness Questionnaires [254] based empirical methods are used in the *Healthcare* domain. Single-Stimulus [258] and Immersive Experience Questionnaire [227] [271] [196] are followed in *Assembly-simulation* based VR applications. Susanne et al. conducted the effectiveness of Questionnaires in VR User Studies as a quality aspect and found that questionnaires reduces Break in Presence (BIP) and theoretical bias [210].

Customized Quality Models: Few researchers proposed Quality models on improving the streaming of Audio and Video data [153] [260]. These models are formulated based on network bandwidth. LTE and 5G bandwidths play a vital role in these quality models where Network key quality indicators and QoE quality indicators are compared with the varied scale of network bandwidth. to judge the streaming quality. Cloud Quality Assessment Model using local binary patterns were proposed to improve the screen quality of VR HMDs for rendering effective scene quality [66].

Conclusion: In this chapter, we present the overall practices of the VR Community while building VR software products. We also present our observations from our empirical study, a review of usability and software quality practices. Improving software quality can be achieved by focusing on better requirements during the development process of software products. By ensuring that the requirements are well-defined, complete, and unambiguous, developers can avoid misunderstandings and misinterpretations that can lead to errors and bugs in the software. Clear and precise requirements also help to ensure that the software meets the needs and expectations of its users, reducing the likelihood of rework and revisions. Additionally, involving stakeholders in the requirements gathering and validation process can help ensure the software meets their needs and expectations, improving its quality. The following chapter presents detailed results of a systematic literature review on Requirement engineering (RE) practices within the VR community. Such a study will help us identify the gaps in current RE practices and help us investigate novel RE methods for the VR community to ease overall VR product development.

Table 3.7 Usability Evaluation Methods - Categorized based on metric and by Industry Type.

Industry	**Usability Evaluation Approach**	**Calculated Metrics**	**Empirical Method**	**Primary Studies**
Automobile	Heuristic Focused Group Experiment	Interaction Capacity Response Time	Informal Usability Tests	[179]
Education	Psychomotor Assessment	Inter-pupillary Distance	Systematic Usability Scale NASA TLX Survey	[35]
	Pluralistic Walkthrough	Understandability	Presence Survey	[47]
	Cognitive Walkthrough	Engagement , Endurability	Pictorial Rating	[11]
Electronics	Focus Group Experiments	Ease-of-Use User Satisfaction	Usability Survey	[29]
Fashion	Cognitive Walkthrough	Effectiveness	Usability Survey	[28]
Healthcare	Controlled Experiments	Response Time Attention , Efficacy	Informal Usability Survey	[247]
	Focused Group Experiments	Reaction Time Response Time	Usability Survey	[79]
	Cognitive Walkthrough	Average Fixation Duration Effectiveness Response Time Index	Social Communication Questionnaire Social Response Scale	[25]
	Informal Controlled Group Experiment	Ease-of-Use Overall Satisfaction	IBM Ease of Use Survey	[65]
HRMS	Focused Group Experiments	Ease-of-Use Understandability	Usability Tests	[301]
Manufacturing	Cognitive Walkthrough	Efficiency User Satisfaction	Usability Survey	[68]
		User Satisfaction	Presence Survey	[117]
		Ease of Usage	Informal Usability Survey	[67]
	Task based Heuristic Evaluation	Interaction Capacity Response Time Effectiveness	Use-case based Usability Tests	[165]
	Heuristic Information Evaluation	Effectiveness Visual Feedback	Effectiveness Survey	[19]
	Haptic based Subjective and Objective Evaluation	Biomechanical Feedback Physical Interaction Capacity	Subjective Evaluation	[290]
	Controlled Experiments	Understandability	Usability Survey	[262]
	Pluralistic Walkthrough	Efficiency Satisfaction Effectiveness	Systematic Usability Scale	[21]
Retail	Use-case based formal evaluation	Ease-of-use	Informal Usability Test	[286]
	Controlled Focus Group Experiment	Interaction Capacity Efficiency User Satisfaction		[142]
Simulation	Controlled Experiments	Response Time	Subjective Evaluation	[83]
	Haptic based Controlled Experiments	Force Exerted Response Time		[234]
	Physiological Signal based Experiments	Skin Temperature Heart Rate Galvanic Skin Response		[147]
	Context-of-Use	Effectiveness Efficiency , Satisfaction	Cognitive Questionnaire	[267]
	Haptic based Pluralistic Walkthrough	Biomedical Feedback Ease-of-Use	Subjective Responsive Index	[251]
	Task based Focus Group Evaluation	Reaction Time Completion Time	Systematic Usability Scale NASA Task Load Index	[276]
Software	Cognitive Walkthrough	Visual Feedback	-	[89]
	Controlled Experiment	Immersion , Accuracy Comfort , Fun Non-fatigue , Non-dizziness Overall Satisfaction Degree of Dizziness	Presence Survey	[158]
		Visual Feedback	Usability Survey	[48]
		Spatial Orientation Test		[111]
	Qualitative Expert Review	Ease-of-Use	Informal Usability Survey	[250]
	Controlled Experiment	Manipulability Comprehensibility	HAR Usability Scale	[236]
		Interaction Capacity Effectiveness	Informal Usability Survey	[249]
Tourism	Pluralistic Walkthrough	Understandability	Systematic Usability Scale	[293]
	Cognitive Walkthrough	Effectiveness Interaction Capacity	Informal Usability Survey	[82]

Table 3.8 Software Quality Metrics mapped with respective type and VR Theme

VR Quality Domain (or) Theme	*Software Quality Metrics Used*	*Type of VR Product*	*References*
Streaming VR Video Quality	Quality Assessment View (QAV)	Streaming VR Video	[296]
VR Audio Quality	MUSHRA test methodology	Use-Case Specific VR Product	[188]
	Auditory Feedback	ML based Approach	[85]
	Auditory Feedback	Questionnarire	[264] [90] [182]
VR Code Quality	LOC, CC, %Lack of Cohesion	Case Study	[93]
VR Image Quality	MultiScale Structural Similarity (MSSIM)	Use-Case Specific VR Product	[258]
	Spherical Structural Similarity Index	Use-Case Specific VR Product	[51]
	DIBR-synthesized IQA metric	Use-Case Specific VR Product	[212]
	perceived viewport quality, Content Dependent Objective quality metric	360 degree video/image	[215]
VR Paranomic Scene Quality	BP-based quality assessment of panoramic videos	Use-Case Specific VR Product	[289]
	Heuristic	Coding Lab	[298] [260]
VR QoE	Electrodermal Activity, Heart Rate, Miss Clicks	Speech and Language Theraphy	[143] [144]
	Immersive Experience Questionnaire	Use-Case Specific VR Product	[227]
	Interaction Modality	Use-Case Specific VR Product	[196]
	Hand Movement Velocity	Rehabilitation Game	[170]
	Interface quality, Realisum Index	Use-Case Specific VR Product	[241]
	Locomotion Index	Use-Case Specific VR Product	[41] [38] [86]
	UserState Measure, Perception Measure, Physiological Measure	Game	[105]
	Egocentric Depth Perception	Task Based Activity	[131]
	electrocardiographic signal, galvanic skin response, blood volume pressure, electrodermal activity	Task Based Activity	[232] [271]
	Flow Experience Analysis	Task Based Activity	[118]
	Heuristic	Assembly Product	[14]
	Presence Evaluation	Use-Case Specific VR Product	[254] [281] [17]
	Subjective Evaluation	Exercise IoT App	[213] [266]
	Subjective Evaluation	Use-Case Specific VR Product	[228] [87]
	Subjective Evaluation	Task Based Activity	[116] [231]
	Subjective Evaluation, content resolution, start delay, stalling ratio	Task Based Activity	[70] [153]
	Subjective Evaluation	Walk-In-Place activity	[199]
	Heuristic Auditory and Sensory Feedback	Case Study - Automobile Driving	[270]
	Heart Rate, Skin Conductivity	Case Study - Public Speaking	[73]
	Spatial Memory Test	Case Study - Spaces	[125] [187]
	Vibrotactile Feedback	Task Based Activity	[192]
	realism, control, interface quality, ability to examine, performance, haptic sub-scales	Haptic Based Case Study	[24] [202]
	Laban Movement Analysis	Task Based Activity	[20]
	In-game Performance, In-game Navigation, Multi-Screen usage	Task Based Activity	[166] [94]
VR Scene Quality	Simulated Annealing, Depth Inaccuracy	Use-Case Specific VR Product	[50]
	Motion-Sickness Dose Value, Electro-Dermal Activity	Use-Case Specific VR Product	[205]
	depth image-based rendering	Use-Case Specific VR Product	[98]
	Multi-modal interactive continuous scoring	Use-Case Specific VR Product	[148]
	temperature distribution	Virtual Surgery - Bone Drilling	[160]
	electrodermal activity quality metric	Super Market	[229]
	Subjective Evaluation, Visual Discomfort	Task Based Activity	[130] [103]
	Subjective Evaluation	Custom Haptic Setup	[49]
	Heuristic Auditory and Sensory Feedback	Assembly Product	[294]
	Heuristic Auditory and Sensory Feedback	Case Study	[146]
	View Plane Representation	View Plan - Visible Surface Determination	[33]
	Attractiveness, Pragmatic and Hedonic Quality	Diagnostic Radiology - Interaction	[282]
VR Simulation Quality	Heuristic	Space Flight Simulation, Driving	[172] [223]
VR Task Quality	rapid upper limb assessment, averaged muscle activations, Total Task Time	Assembly Product	[208]
VR Video Quality	MultimediaQA, VRVideoQA, Image QA	Use-Case Specific VR Product	[288]
	depth-image-based rendering, peaksignal-to-noise ratio	Use-Case Specific VR Product	[240]
	omnidirectional IQA	Use-Case Specific VR Product	[69]
	weighted Peak Signal to Noise Ratio	Use-Case Specific VR Product	[248] [173]

Chapter 4

Literature Review - Requirements Engineering for VR Product Development

This chapter presents our detailed systematic mapping study on Requirement Engineering practices within VR community. This exercise identifies the methods for given thrust areas from the literature and also helps us comprehend the scope and intent of each method for a given VR application domain.

4.1 Mapping Study - RE for VR Software Products

Karakostas et al. [164] have articulated various aspects of Requirement Engineering (RE) like Requirement Elicitation, Requirement Analysis, Requirement Specification, Requirement Validation, and Requirement Management. They clearly distinguished what Requirement Engineering should focus on for any given technology. Vegendla et al. have conducted a Systematic Mapping Study on RE in various software Ecosystems [268]. They recorded all possible methods practiced in the software ecosystem. Pacheco et al. have studied the maturity of RE methods in traditional software products [123]. Vivian et al. conducted qualitative studies on using VR technology to gather requirements for VR product development [104]. Their observations show that Virtual Environments help gather requirements in developing VR simulators and VR training environments. However, their work focuses on requirements-gathering processes in Mozilla Hubs and AltspaceVR-specific applications without explicitly mentioning a specific method in detail. Fariha et al. explored open-source Unity VR projects to understand how developers optimize VR applications [193]. They focused primarily on performance optimization in the context of non-functional requirements, and there was no mention of a requirement engineering method required to optimize a VR project.

Requirements gathering and analysis are still challenging for mainstream software products [216]. Over the years, advancements in requirement engineering methods have enabled the development of processes wherein systematic elicitation, analysis and specification techniques are used rather than ad-hoc requirements-gathering methods. While researchers have written about the need to move from gathering to eliciting requirements, Jaramillo et al. [124] were the first to conduct an extensive systematic literature review on requirement elicitation methods practiced on mainstream software products. The study covered research papers spanning 25 years. In the study, the authors claim that the results remain relevant even now in understanding the state of art in Requirements Engineering. Their work provided insights into various perspectives of requirements to be considered for mainstream software products. Sufian et al. conducted a large-scale survey of software requirements prioritization Techniques [257] for mainstream software products. They found various prioritization techniques and tools used in the requirements-gathering process. One of the objectives of that study was to identify tools & techniques that new researchers in the area can use. Studies by Sufian et al. [257], Pacheco et al. [123], Jaramillo et al. [124], and Vegendla et al. [268] help understand the potential usage and maturity of existing RE methods in conventional software and do not provide any insights on the methods in the VR domain.

Santos et al. [60] conducted a systematic literature review on consolidating existing evidence regarding the requirement-gathering methods for building Virtual Reality systems and also studied the use of Virtual Reality technology to conduct the Requirement gathering in general. Their results indicate the need for studies that show the use of the RE process for VR systems as technological peculiarities need to be considered [60]. Their conclusions are based on a small dataset (only 12 primary studies). Their study lacks a detailed exploration of areas like elicitation, analysis, and specification of requirements for VR products. Most of the studies from the existing literature do not provide any insights into the RE methods used in the VR community. These studies primarily highlight the gaps in RE methods during VR product development. As the rigor required in understanding RE methods for VR is high, there is a need to study the literature in detail. Such a study can provide an overview to the VR community about RE practices and help them possibly adopt the best methods for better VR product development.

Mapping Study Steup

This section discusses the mapping study process, including search strategy, search string, databases, exclusion & inclusion criteria, paper selection, and data extraction. *"Requirement Engineering"* is a multi-layered process that constitutes the following stages:

- ***Requirement Elicitation*** - a practice of researching and discovering the requirements of a system of users, customers, and other stakeholders.
- ***Requirement Analysis*** - a practice of determining user expectations for a new or modified feature or a product.
- ***Requirement Specification*** - a practice of documenting requirements in a prescribed purpose and order of implementation.
- ***Requirement Validation*** - a process of confirming the specified requirements agree with the stakeholder's requests.
- ***Requirement Management*** - a practice of ensuring overall goals of product development are successfully met during the development and release of a software product.

Table 4.1 PICOC definitions for literature study on Requirements Engineering approaches in VR

Criterion	Description
Population	Virtual Reality related products and applications
Intervention	Requirement Elicitation, Analysis and Specification Methods
Comparison	Comparison of results captured in various RE Studies
Outcome	RE methods applied to VR product and apps
Context	published academic and industrial studies

A requirement analyst follows these steps to provide precise inputs to the underlying design, development, testing, and release teams. This clears ambiguity and confusion among the underlying developer teams. Among these stages - Requirement Elicitation, Requirement Analysis, and Requirement Specification primarily orient toward various methods & techniques. In contrast, Requirement Validation and Requirement Management (including change management) tend to be subjective in practice and implementation. Thus, as part of our mapping study, we focused on the methods and techniques practiced during Requirement Elicitation, Analysis, and Specification.

Research Questions

The Systematic Mapping Study described in this section uses the guidelines suggested by Petersen et al. [204] in conjunction with the PICOC method [22]. PICOC (Population, Intervention, Comparison, Outcome, and Context) can be used to define the scope and context among reviewers who conduct an investigation. The general notions of the PICOC method applied to our study is shown in Table 4.1. Using the Intervention and population criteria of PICOC, we constructed the potential research questions of the mapping study under the set context (published academic and industrial studies) defined as part of PICOC. Below are our research questions:

RQ1: *What are the requirement elicitation methods used while building VR product(s)/app(s)?*
RQ2: *What are the requirement analysis methods used while building VR product(s)/app(s)?*
RQ3: *What are the requirement specification methods used while building VR product(s)/app(s)?*
RQ4: *Are there any requirement engineering methods for VR product(s)/app(s) specific to a field of Interest?*
RQ5: *What are the specific attributes captured during elicitation, analysis, and specification of requirements for VR Software Product(s)?*

All the above research questions are formulated to explore the requirement engineering methods adopted as part of various stages of VR product development requirements and the adoption maturity. Here, *"maturity"* of a method is meant to signify a widely used and accepted method over time by a wide range of VR community. For the rest of the paper, we describe *"VR community*" as VR researchers from academia and industry who illustrated their work in the available literature only that are found using respective requirement engineering methods for our investigation. We do not consider individual VR practitioners from Industry who practice requirement engineering methods in their day-to-day deliverables as part of this mapping study as it requires an additional ethnography study to understand their practices in detail. Research questions ***RQ1***, ***RQ2***, and ***RQ3*** are formulated to explore the requirement elicitation, analysis, and specifications methods separately for better illustration and mapping. Santos et al. [60] used their observations on a very limited dataset, whereas we studied literature available on various databases and examined methods adopted in various stages of VR product development requirements. Research question ***RQ4*** considers adopting the observed methods from elicitation, validation, and specification in the field or domain of interest. Here the *"field of interest"* is interpreted as a class or a domain of products that fall under a particular type of industry. This categorization is only to understand how RE methods vary across industries. Still, it does not mean the RE method is unique to a particular domain or should not be employed on other domain applications. ***RQ5*** was to articulated to understand what VR attributes are considered by VR community for eliciting, analyzing, and specifying requirements for VR software products. This facilitates the creation of the bare-minimum set of attributes that constitute VR products and, therefore the development of VR-specific RE templates that may be used by VR community while managing overall RE process.

Search Strategy

An essential part of the mapping study is the identification of keywords used for searching the databases. More often than not, this tends to be an iterative process. The research questions - ***RQ1***, ***RQ2***, and ***RQ3*** are related in many aspects. Hence, the initial iteration involved a shared search query set. However, peer reviews on the resulting papers found too many outliers that did not fit the intervention criterion specified in Table 4.1. Thus we had to formulate a separate search string for elicitation and another for analysis and specification.

Relevant work and observations for ***RQ1***, ***RQ2***, and ***RQ3*** are detailed independently. No new search query strings are defined for ***RQ4*** and ***RQ5***. Based on the search results for ***RQ1***, ***RQ2***, and ***RQ3***, observations for ***RQ4*** and ***RQ5*** are deduced.

Search String: The finalized search string deduced after multiple iterations is given below:

C1: "VR" **OR** "Virtual Reality*"
C2: "Requirement Elicitation" **OR** "Software Requirement*" **OR** "Requirement Engineering" **OR** "Requirement Management" **OR** "Requirement process" **OR** "Requirement Evaluation" **OR** "Functional Requirement" **OR** "Non-Functional Requirement"
C3: "Requirement Analysis" *OR* "Requirement Specification" *OR* "Functional Requirement" *OR* "Non-Functional Requirement"

'*C1*'**AND** '*C2*' is the resultant search string formulated for research question ***RQ1***.
'*C1*'**AND** '*C3*' is the resultant search string formulated for research questions ***RQ2*** and ***RQ3***.

Given the focus on VR, the query string ***C1*** considers the possible variants of the Virtual Reality keyword. The requirement Engineering domain is the core of our study; thus, all likely keywords regarding requirements are considered. The initial version of the ***C2*** query string contained 'Requirement' as one of the keywords. We had to remove the keyword 'Requirement' as it returned too many papers that were not relevant to the study. Further, we narrowed down the list of keywords as per RE process definitions [164]. Given that VR technology is relatively recent, we did not consider the year of publication as a filter for the search string. The query string ***C2*** focuses on Requirement Elicitation, and ***C3*** focuses on Requirement Specification and Analysis. Overall, we considered papers that mention requirement elicitation, analysis, or requirement specification method(s) to develop a VR product and have moderate to significant information about the VR Product. For example, at minimum, the papers that provided details about the product input/output, domain, target users, scope and extent of interactions, etc. were considered. Investigating the metrics and demerits of the observed methods over each other is not in the scope of our current mapping study as there isn't enough literature available to establish the facts. However, we illustrate the benefits of a few methods by using the observed field-of-interest as a data point.

Search Quality Assessment

We followed Petersen et al. [204] guidelines on formulating and finalizing the search string. We developed an initial search string and reviewed it among this mapping study's authors and other fellow researchers from our research center. The peer-review approach [204] helped us revise the search string closer to our research questions. The authors of this paper have independently conducted the search activity after finalizing the search string, and the results were independently recorded. This activity helped us address search filtration biases. It is an iterative process that sometimes generates outcomes conflicting with each other's observations. We used "*Inductive Reasoning*" to address the conflicts by mapping the specific observation of a peer reviewer with our research questions to arrive at a generalized conclusion that agrees with all the reviewers. This process helped us avoid bias in decisions, deviations from our study, and duplicated results. After thorough consideration, the generalized conclusions are considered for developing the search string for our mapping study. The search was conducted against the research paper's attributes like abstract, contents of the paper, title, and keywords. Further filtration was done on these attributes to eliminate unmatched research papers. Our supplement data with collections of the gathered papers is available here [181].

Databases and Paper Selection

The study considered published research papers until February 2023. Major electronic research databases - IEEE Xplore, ACM Digital Library, Springer, and Science Direct were used for conducting the search. The dataset includes special edition journals, articles, industry talks, prototypes, etc. Grey Literature, i.e., research that is either unpublished or has been published in the non-commercial form (for example - Policy documents, standards, etc.), were omitted. We focused only on active research publications. There were no additional filters placed on these databases. The database included publications of indexed journals, conference proceedings, review papers, workshop proceedings, late-breaking works, industry track papers, and companion proceedings. The search order performed on the databases returned most of the results. The search fields and search string were formulated so that the search process was similar across all the electronic research databases.

Exclusion Criteria - Articles with unnecessary information about requirement engineering, topics related to the description of requirement engineering, or industry white papers were excluded. Articles with sparse details about the requirement elicitation, analysis, or specification method and with no related study setup on VR products were ignored. Papers that did not mention anything about the requirements of the VR product built as part of their research were also not considered. Newsletters, editorials, press notes, and magazine articles were excluded from the study as they did not have enough literature for our research. Reports were excluded from the study as they have insufficient details about the practice of requirement engineering methods. Books were excluded from the study as they are too comprehensive for a mapping study and are likely candidates for literature review rather than a mapping study. As shown in Fig 4.1, all the paper counts in each stage are exclusive to a particular database, and we avoided repetition of the same paper across the databases and considered it under one database.

Inclusion Criteria - Papers that proposed or revised or discussed requirement engineering method(s) and approach(s) were considered as part of our study. Book chapters, conference proceedings, conference articles, and early-access research papers were considered for review. Full-length journal proceedings were included as part of the journals. An equal match of the search keywords against the title, abstract, keywords, or some part of the paper was also considered. The paper consists of the studies conducted in academia or on an enterprise product that were given primary consideration. All papers were included immaterial of the publication date. Papers published only in the English language were considered.

Paper Extraction

We followed the paper extraction guidelines proposed by Petersen et al. [204]. Attributes like Paper UniqueID, Paper Type (Journal/Conference/Review Article), Author(s), Editor(s), Title of the paper, Pages, keywords, DOI, year of publication, ISBN, Publisher, Extraction date, and Type of DataSource are extracted for each paper. As shown in Fig 4.1, we conducted data extraction for Requirement Elicitation and Requirement Analysis-Specification separately based on the search strings designed for respective research questions. Query strings **C1** and **C2** are utilized for extracting papers for **RQ1**, i.e., for Requirement Elicitation. Query strings **C1** and **C3** are utilized for extracting papers for **RQ2** and **RQ3**, i.e., for Requirement Analysis and Specification.

Our initial search of query strings **C1** & **C2** generated ***1771*** papers from respective databases. We filtered these papers using the exclusion and inclusion criteria and later conducted forward and reverse snowballing. This resulted in the final set of ***60*** papers for requirement elicitation methods practiced by VR community. Our second search query string **C1** & **C3** generated ***932*** papers in total from all the databases. We filtered these papers based on the exclusion and inclusion criteria and separated them into two buckets. One with all papers on requirement analysis and the other with papers on requirement specification. After applying forward and reverse snowballing, we arrived at a finalized list of ***36*** papers for studying requirement analysis methods in VR and ***10*** papers for studying requirement specification methods in VR. The overall search filtration data is made available as part of supplemental material [181].

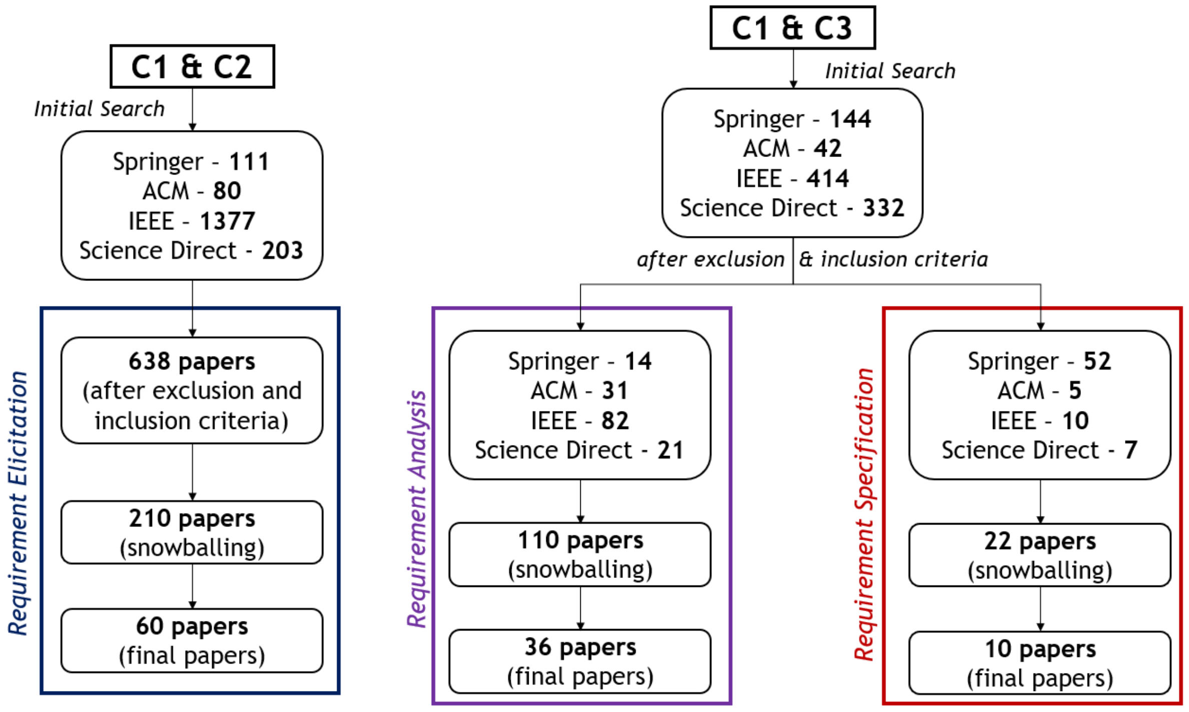


Figure 4.1 Paper Extraction Process and corresponding results

Artifacts

We conducted the mapping study based on the guidelines proposed by Peterson et al. [204]. The resultant artifacts of the work are a collection of research papers downloaded from various research paper databases. We initially downloaded research papers using dedicated search strings for the respective research paper databases. Using this initial download paper dump, we filtered papers at every step of our mapping study. Each research paper database provides a paper download in different formats. For example, IEEE and Springer provide paper titles and meta-data information in CSV (comma-separated values) file format. ACM will only provide you with each paper's BibTeX Bibliographical Database file. Science Direct will provide you with the BibTex file and the paper's PDFs for download.

We narrowed the initial paper dump to the final paper set by conducting filtration methods in the following order as per systematic mapping study guidelines [204] - search quality assessment, peer-review by a domain expert, applying exclusion and inclusion criteria, and forward-backward snowballing. Using the final paper set, we illustrated our mapping study results. These artifacts are reproducible using the same study setup and filtration process. The artifacts produced are documented as part of supplemental material [181].

4.1.1 Requirement Elicitation Methods for VR Software Products

The search strategy and filtration criteria ensured that each paper reviewed had some aspect of elicitation, analysis, or specification of requirements for VR products or prototypes. In this section, we discuss our observations on the types of methods identified, a trend identified by a method over the years, and methods specific to a field of interest as the responses to our research questions.

RQ1: *What are the requirement elicitation methods used while building VR product(s)/app(s)?*

The elicitation methods used by VR community are categorized into four different themes. The themes are categorized based on our analysis of filtered literature and the nature of usage/implementation of the requirement elicitation methods in developing a VR product or a prototype:

- Widely Used Methods
- Focused Methods
- Customized Methods
- Distinct Methods

Widely used methods are predominantly described and applied similarly in the relevant literature. Focused methods are limited to specific categories of VR products. Customized methods are methods claimed to be a common RE method but are customized by the VR community to achieve their goal. Distinct methods are RE methods that are not conventional RE methods but are developed for particular VR products.

Widely used methods

In instances where VR community are unaware of the initial product design or have very limited knowledge about the VR product, *'Direct Inspection'* method has broad acceptance across all the domains. The direct method is simple and empowers the requirement analyst to figure out the requirements by themselves. In the early stage of product development, with no clear way forward, requirements are determined based on the potential interests of the end users. No other structured methods are followed by the practitioners to record and formulate the initial requirements in these cases. A few of the use cases include - building collaborative design apps using VR for better Design [239], building user-centered data visualization in VR [40] for personalized view, simulation-based evaluation of user interfaces in automotive industry [207] for multi-user and multi-task assistance, implementing a second-life virtual world platform for effective human interaction [127] for addressing social phobias, haptics based editor for designing immersive experiences in VR [63], and NASA's simulation-based ground test and protocol analysis tool for spacecraft [91].

In most domain-specific applications, specifically domains like healthcare, VR community seek inputs from *'Domain Experts Review'*. Simulation-based medical planner for patients with cardiovascular disease [252], Training medical practitioners on conducting minimal invasive abdominal cancer surgery [280], educating surgeons about endoscopic surgery using simulation-based methods [295], training dental practitioners on addressing dental anesthesia [58], implementation of patient-specific cognitive engine for robotic needle insertion during surgeries [259], conducting eye-tracking based evaluation techniques for a nursing professional to treat pediatric patients [201], Spinal Cord Injury re-hab [133], Care ecosystem for ADHD patients [253] etc. require domain experts to be a part of the development process. Other prototypes included the implementation of remote handling operations for a JET [233], virtual training systems for Industry workers [195], Game-based education on Safety for Industries [128], facilitating building model adoption in the construction industry [99], AR-based application for controlling resources on construction projects [149], developing a visual inspection system for an in-vessel robot in spaceships [274], motion-based games for building physiotherapy applications [200], improving social skills in autistic children [26]. Studies on domain expert reviews in other domains like entertainment and tourism include the implementation of virtual Indonesian musical instrument [108], developing a 3D city model for real-estate market development [235] and VR Tour Guiding [167]. A Likert based questionnaire is used to prioritize the need for a feature in a VR-based training tool called MediTool [168]. User feedback is captured by providing 2D and 3D-based training setup to understand the impact of VR-based training on medical surgeons. VR Cycling Scenes are developed to use them as part of an empirical field study on understanding the physical environmental factors related to cycling in older adults [162]. Practitioner's reviews are captured to develop the flow of the cycling scene to ensure that they are relevant to the context of cycling.

'Survey Questionnaire' is another requirement elicitation method preferred by VR community. Use cases like luxury brand virtual shopping [18] with diverse options for selection as features, building virtual worlds for personalized data and process visualization [102], understanding human emotional

responses concerning arousal and valence [75], providing virtual experiences through social VR [101], understanding students' perspectives on mixed reality in higher education [279] are few of the research prototypes that followed survey-based questionnaire method.

Focused Methods

VR community use the existing *'Domain Knowledge'* method while formulating requirements. In this case, the domain expert elicits the requirements and then reviews them by VR community to assess whether they can be built as a VR product. This is in contrast to the widely used method of *'Domain Experts Review'* where in the requirements that are previously elicited are reviewed by the domain expert.

Enhancing UML modeling using VR [37], developing a structured teaching system using virtual platforms [242] and building location-based services for virtual college campus tour [277] are research works where VR community have relied on existing domain knowledge. In [189], authors discuss a VR-based simulator training in the maintenance of Railway Wagons.

While solving real-world problems, VR community have relied on the *'Practitioner's Studies'* or literature of existing practitioners. Use cases include a therapy system for arm and hand rehabilitation [211], Building a haptic-based dental simulator and its related patient examination system [273], examining multi-modal interactions for foot reflexology [55]. Practitioner's Studies are widely preferred for use-cases involving protocols and standards. In cases where VR community require precision in requirements, *'Subjective Inspection'* method is followed. Software Development Process Education [100], Therapy for overcoming Glossophobia or fear of public speaking [113] and building a haptic feedback system for assessing wrist motor functions for patients with upper motor neuron lesions [163] are use cases where this method is used.

VR community follow *'User Case Studies'* in cases where VR applications require user-centered observations. VR community built novel solutions for large-scale collaborative VR environment simulations [39], multi-user VR environments [122], social interaction using ubiquitous virtual environments [157], developing ontologies for real-time interactive systems [278], developing virtual class service using a predefined SERVQUAL methodology [64], Edge Computing based Human Behaviour Recognition System [243] building intelligent systems for virtual roaming [52] and user-centered design for facial surgery training [177]. All such cases required a focused understanding of users and their behavior in a virtual environment.

Customized Methods

Customized methods are formulated to address complex requirements, specifically where a common RE method is unavailable to achieve its goal. It applies to instances where VR community must build a particular solution for a targeted VR problem. *'Case Studies'* based method helps VR community understand the complexities behind end-user demands and capture customized observations to address the product needs. Virtual Negotiation Training application [43], Designing a game-based interactive stroke rehabilitation system [109] followed case study-based method. Our study observed some other minor methods apart from the above methods. *'Situated Cognitive method'* used to study the social conventions on learning for non-natives and low literates [238]. *'Participatory Workshops'* method is used to develop serious games for providing and promoting healthy habits in the northeastern Brazilian countryside [61]. For heuristic evaluation of virtual museum *'Structured Interviews'* are conducted from end-users via smartphone [184]. *'Behavioral Questionnaire'* method is used for examining the participants for studying therapy for a sense of presence and meta-cognition to VR exposure of Social Phobias and Fear of Flying [197]. *'Role Play Flow Analysis'* method is used while developing a role-play game-based app for a virtual environment. VR community develop and practice these custom methods to address VR products' important requirements and market needs.

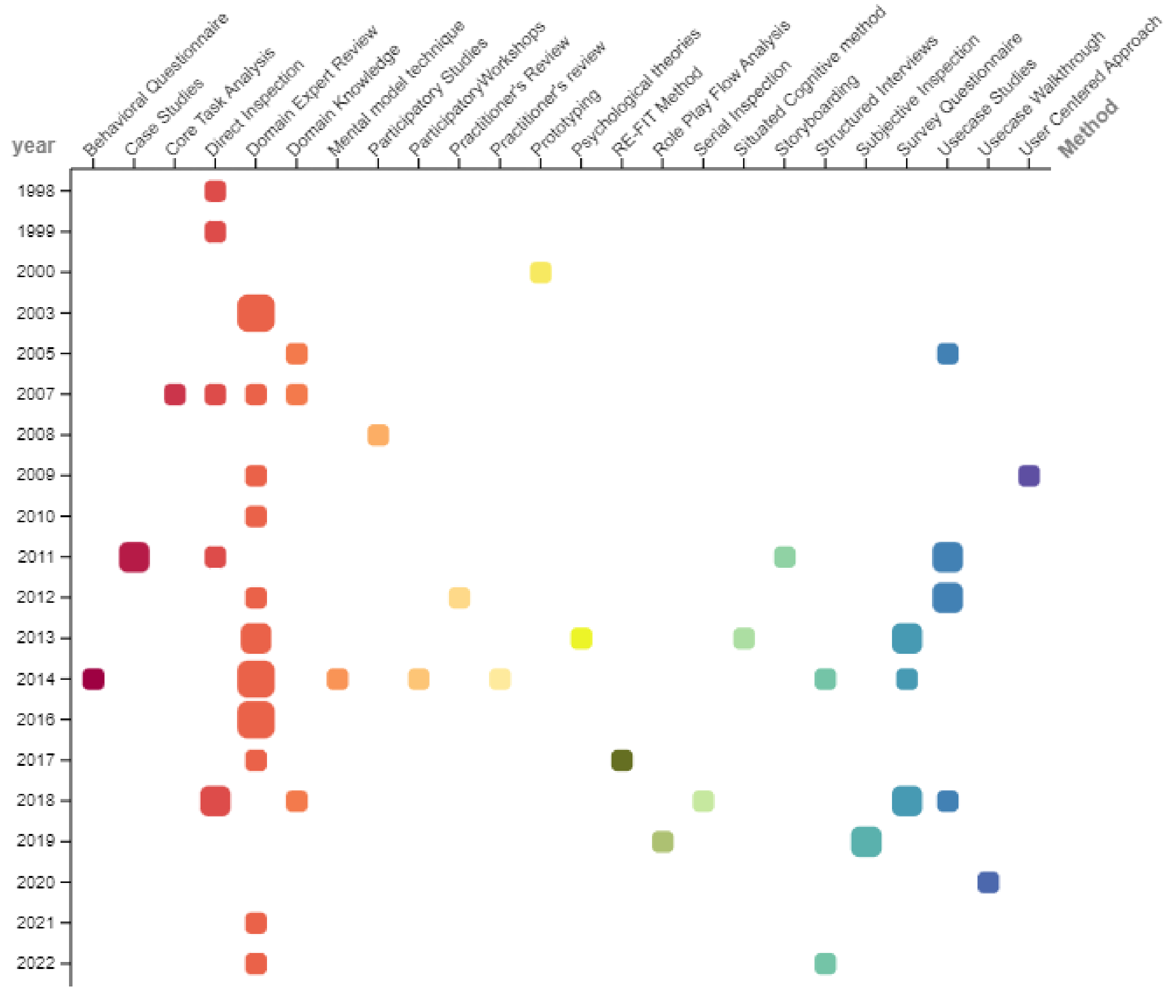
Distinct Methods

In a few use cases, VR community followed unconventional methods to meet the product requirements. Such methods are referred to as Distinct methods. Practitioners followed the rapid *'Prototype'* method to address frequent changes in requirements [71] while building virtual actors for a collaborative learning environment. It helped them to address dynamic requirements and enhance the prototype in a timely fashion. Another method called *'Core Task Analysis'* is used for developing a mobile augmented reality tool for maintenance work [237]. A method like this can help developers drill down the necessary features based on the tasks defined in the application. VR developers can easily correlate the intent and motive behind a defined task while building the VR product. *'Story boarding'* method is used while building training systems for bridging gaps between end-users and developers in the Software Development ecosystem [222]. It provides an end-to-end flow of responsibilities for every task in a VR product. A method like *'Psychological theories'* is used for developing a virtual environment for early diagnosis of dementia patients [272]. It follows a steady and thorough multilevel process to capture requirements before finalizing the features. *'Mental model technique'* is another method used to construct an image-based spatial presence of virtual experience [180]. Requirements are generated based on the tailored metal model defined per participant. A customized method called *'RE-FIT method'* proposed by Bhimani et al. [36] uses virtual experience for empowering people with special needs. This method steadily captures the requirements and correlates with the end-users needs.

Maturity of the requirement elicitation methods in practice

RE elicitation methods have matured over the past three decades. The practices in some methods have evolved while others remain unchanged. We broadly categorize the unchanged methods as Traditional Methods and all others as Evolved methods. Fig 4.2 describes the year-wise requirement elicitation methods used by VR community and the methods' maturity over time. Each squircle in the figure represents the count of papers relevant to that RE method published in a given year. The size of the squircle is correlated to the number of RE methods. Squircles with a darker red shade are traditional methods with an unchanged overall elicitation process. The Squircles with a darker shade of blue color are evolved methods whose elicitation process has changed over time. The Squircles in the middle that falls between red and blue shades are distinct methods that address specific VR problems.

Figure 4.2 Requirement Elicitation Method and Maturation levels



Traditional Methods Based on the available data, we observe that the adoption of requirement elicitation methods in VR started more than two decades ago with simple methods like *'Direct Inspection'*, *'Prototyping'* and *'Use case Studies'*. As per Fig **??**,*'Direct Inspection'* is unchanged and widely used. Researchers conducted a direct inspection to determine the value of the product scope for target users and then formulated constraints of the product idea by not implementing any specified techniques to build an early prototype version of VR products. Most early VR products are highly focused on a specified need or activity, which might not make VR community think about novel methods while capturing requirements. Early VR applications for collaborative design-based VR environments [239] and User-centered data visualization [40] are no longer usable as these VR apps are highly sophisticated and dependent on customized external hardware for generating immersive experience to the end-users. The value of VR simulation gained prominence in the healthcare industry, leading VR community to adopt *'Domain Expert Reviews'* as a method for gathering requirements to build simple applications for healthcare practitioners.

Evolved Methods - With the advent of simplified hardware for VR, new VR applications emerged around 2009. As per Fig 4.2, there is a shift in requirement elicitation methods used for building VR applications that require domain knowledge or domain expert reviews. The need for VR community to interact with real-world practitioners to gather and correlate their gathered requirements with real-world scenarios has gained prominence. With the rise in addressing complex problems through VR, VR community built customized methods to meet the requirements of the target audiences of respective VR applications. *'Situated Cognitive method'* and *'RE-FIT Method'* are distinct methods that are created by VR requirement analysts to elicit and finalize requirements. Other distinct methods, including *'Participatory Workshops'*, *'Role Play Flow Analysis'* for multi-user applications, and *'Core Task Analysis'* for task-based applications with complex actions are recommended for VR developers. Most remarkably, methods like *'Psychological theories'* recorded by domain practitioners and *'Mental model technique'* help VR community to develop better User-centered social or personalized VR applications in the future.

Among all the elicitation methods studied, usage of the automated or semi-automated method was not observed as part of prevailing practices in the VR development ecosystem. VR community uses manual elicitation methods as part of their requirement elicitation process. Although, this was not the initial criterion for our inquiry in this study, we make this observation for future work.

4.1.2 Requirement Analysis Methods for VR Software Products

RQ2: *What are the requirement analysis methods used while building VR product(s)/app(s)?*

The Requirements Analysis methods published and available in the various databases are less than the requirements elicitation methods. Table 4.3 illustrates most of the requirement analysis methods practiced by the VR community. Based on the available literature, it can be observed that VR community have customized conventional requirement analysis methods to determine user expectations for a new or modified product.

Requirement Analysis Methods in Practice

In this section, we collated all the requirement analysis methods practiced widely in detail in most of the observed literature. For clarity, we describe the requirement analysis method's name and usage on an example system.

'Object Oriented Method' is observed to be the most widely used analysis method, primarily for VR products in healthcare like developing electro-vibration haptic feedback simulation system [299], exposure therapy for acrophobia [16], Ankel Rehabilitation [81], and Endotracheal Incubation Simulation Training [218]. Other customized versions of the Object Oriented method are *'Object-Entity modeling'* and *'Object Oriented domain-centered method'*. These methods are used in recreation and simulation-based education VR products [300] [206].

Methods like *'Control Flow Chart Method'* are primarily used for building VR products in Manufacturing [62] [112]. *'Use case flow diagrams'* [23] [132] [209], *'Heuristic (Human-centered approach)'* [226] [224], and *'Physical mock-up analysis methods'* [34] [45] are also widely used.

A few distinct methods were employed for specific VR products like *'Abstract-to-Concrete Gap Analysis'* to develop a Mechatronic Assembly Simulation Scene [244], *'Strategic Dependency Model (i* star)'* for developing Multi-agent 3D Game for Cognitive Simulation for manufacturing [9], *'Thematic Analysis'* on developing a Vehicle Pedestrian Communication for Vision Impaired pedestrians [56] and *'Workflow Technique'* for Micro-robotic cell injection training system [80].

Maturity on Usage of Requirement Analysis Methods

Early works that used requirement analysis methods for VR products primarily relied on model-based validation. For example, *Flow Chart* for building Virtual Campus System [287], *Sequence Diagram* for Surgery Simulation [27], *Unified Modeling Language* for VR Conceptual modeling [37] and *Abstract-to-concrete Gap Analysis* for developing Mechatronic Assembly Simulation [244] are few early VR scenes that relied on model-based requirement validation which is conventional software engineering practice. Later, we observed that most requirement analysis methods are centered around *Objected Oriented method*, its variants, and *Flow Chat Analysis* over time. Apart from these observations, no other significant insights could be generated with the available data on the maturity of requirement analysis methods in VR product development.

Although this is separate from the initial investigation criteria and by considering the observed requirement analysis methods, it isn't easy to automate requirement analysis methods when compared to requirement elicitation methods. The observed requirement analysis methods are more potent in examining the technical feasibility of VR product capabilities during the analysis phase. These methods must be modified to manage requirement traceability if the requirements are incrementally captured. VR product development is not amenable to incremental development life cycle as it significantly increases development costs [138]. Thus there is scope for automation and development of novel traceability-based requirement analysis methods.

4.1.3 Requirement Specification Methods for VR Software Products

RQ3: *What are the requirement specification methods used while building VR product(s)/app(s)?*

The available literature on Requirement specification methods is significantly low. There is limited evidence of VR community using VR-specific requirement specifications methods or adopting conventional software engineering methods for VR products. We observed *10* studies from the available literature that orient towards the following methods illustrated in Table 4.4. *Functional SRS Documentation* is a predominantly used requirement specification method by VR community. There are other variants of this method, such as *UML/Legacy SRS documentation* [214], *UML/Design Research Documentation* [13], *Templatized Specifications* [194], *Plan Documentation* [129] and *Task Tree Documentation* [57]. These variants are customized versions of the prevailing SRS documentation method from traditional software engineering. For example, task tree documentation and templatized specification methods have a distinct way of specifying requirements post-elicitation. Task tree documentation specifies requirements in a tree-branch-node form. Templatized specification method specifies requirements in collapsible-expandable form rather than a linear SRS documentation.

Common Scene Definition Framework [32], *Conceptual VR Prototyping* [171] and *Mental Modal Technique* [180] are distinct requirement specification methods that are used on sample VR scenes and studying spatial presence. In contrast to requirement elicitation and requirement analysis from previous section, discussing the maturation of requirement specification methods over time is difficult due to the very small dataset. Elicitation and Analysis of requirements in VR are widely practiced and matured over time. However, this is not the case with requirement specification methods. However, Functional SRS documentation appears to be the only widely used method by almost all VR community. Thus, an exploratory study among VR community is required to understand their requirement specification practices in detail.

4.1.4 RE Methods with specific Field of Interest for VR Software Products

RQ4: *Are there any requirement engineering methods for VR product(s)/app(s) specific to a field of Interest?*

Fig 4.3 illustrates the overall requirement elicitation, analysis, and specification methods adopted by VR community to build potential VR products. Table 4.2, Table 4.3, and Table 4.4 provide a broad categorization of requirement elicitation, analysis, and specification methods used by VR community grouped by field of interest. We have classified the identified research works into the potential field of interest like *'Design'* - for product owners and developers, *'Analytics'* - for data visualization, *'Education'*

- for both social learning and technical education, *'Health care'* - for physiotherapy, general medicine, surgeries, and rehabilitation; *'Space'* - for vessel inspection, simulation studies, and training; *'Automotive'* - for safety regulation; *'Construction'* - for quality control; *'Entertainment'* and *'Social'* Industries for respective business centered applications. *'Manufacturing* was distinctly observed in requirement analysis papers with detailed practitioner simulations for large-scale industries. *'Tourism'*, *'Simulation'*, *'Recreation'*, and *'Military'* industries practice requirement analysis and specification methods for their respective domain needs. This classification does not claim that the observed methods are limited to only a particular field of interest. The methods illustrated under a given field of interest can be employed in another. However, we only deduced these classification based on our observations from the gathered literature.

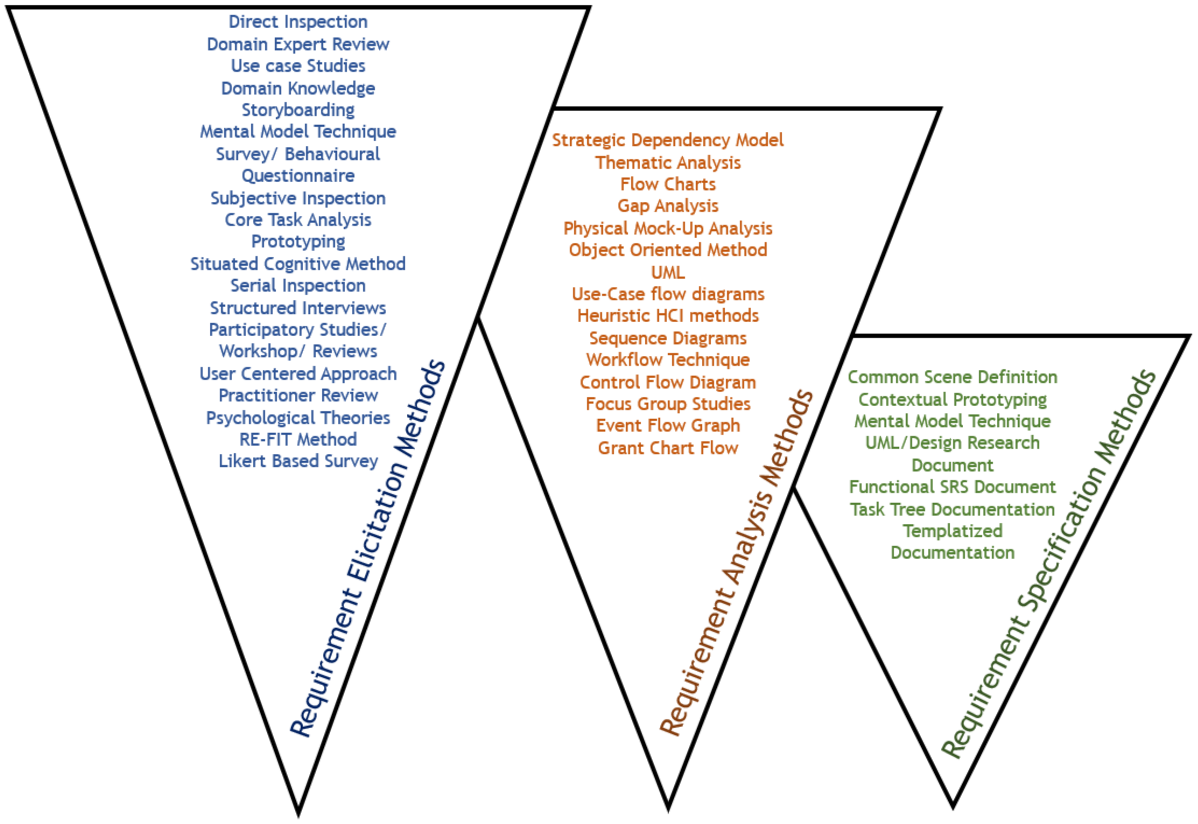


Figure 4.3 Overall Requirement Elicitation, Analysis and Specification Methods observed as part of Mapping Study

The study shows that the Healthcare domain is an early adopter of VR-based applications to address complex problems. Table 4.2 and Table 4.3 show that requirements elicitation and requirements analysis methods are widely practiced in this domain. VR applications in healthcare involve training activities, simulation-based surgery studies, rehabilitation practices, medication planning, and treatment planning. Practitioners have adopted the *'Domain Expert Review'* elicitation method as the best fit in such healthcare applications. In a few cases, they rely on *'Practitioner's Reviews'* and respective theories to elicit requirements. One reason for choosing such a focused method could be the real-world implications of these products on end-users. In contrast, there is limited usage of requirement-specific methods in Healthcare.

'Object-Oriented method' is a detailed requirement analysis method compared to other requirement analysis methods like *'Workflow Technique'*, *'Focus group studies'* and *'Heuristic - Human-centered methods'* practiced in Healthcare domain. In general, almost all the domain-specific tasks are well managed by experts from the respective fields. Thus involving such experts as part of requirement elicitation will yield better results and pave a path for a robust VR product. Other elicitation methods like *Case Studies*, *RE-FIT method*, and *Psychological theories* that have distinct features and offerings are also used.

Products built for the social network domain widely rely on *questionnaire-based* requirement elicitation method. The questionnaires are subjective and objective in structure. They are designed to capture target-user understanding in detail and help the VR community to prioritize and plan the feature after every release. Entertainment, Design, and other minor domains follow various questionnaire-based requirement elicitation methods. Education is a unique domain that adopted processes like *Core Task Analysis* and *Situated Cognition method* to build VR-based educational products. These elicitation methods are structured and iterative in their setup. This aids requirement analysts in conducting requirement studies after product enhancement and subsequent release. On the other hand, *Use-case flow diagrams* are observed to be widely used requirement analysis methods in the Education domain. The *Core Task Analysis* elicitation method appears to be ideal for VR applications requiring frequent requirements changes. It also decreases the cost of analysis as this method has the flexibility to capture the understanding of change in specification. Military is one such domain that appears to have applied *Flow Chart analysis* method along with a *Plain documentation* method for the specification of requirements. Recreation-based VR applications also utilized *Functional SRS documentation* as a specification method, whereas employed *Object-orient method, Flow Chart, and Use-case flow model* for conducting requirement analysis.

Overall, the observed requirement engineering methods practiced in Virtual Reality Products - which include requirement elicitation, requirement analysis, and requirement specification, are illustrated as part of Table 4.2, Table 4.3, and Table 4.4. These methods are grouped by field of interest to display their adoption across these areas. This information may help requirement analysts to choose and adopt an appropriate method for conducting hassle-free VR product development.

4.1.5 RE Attributes to Capture VR Product Requirements

RQ5: *What are the specific attributes captured during elicitation, analysis, and specification of requirements for VR Software Product(s)?*

Requirement analysts are expected to have some degree of domain understanding for the product that they are gathering, analyzing and specifying the requirements. In general, a meta-model-based domain understanding is a common practice for most requirement analysts to interpret the minimum set of attributes of a particular domain. For example, fields like Multi-Similarity Systems [203], Distributed Embedded Systems [44], etc. are few such cases that require an understanding the domain specific attributes and possibly a creation of domain-specific languages. Based on the reviewed literature we were able to formulate a minimal set of attributes required to conduct requirement elicitation, conduct analysis and then specify for VR product development. We term them as bare-minimum attributes. To start with, VR community can use these essential attributes to conduct elicitation, validation, and specification of requirements for VR product development. The following are the bare-minimum attributes deduced heuristically using the ***"abductive reasoning"*** approach (a method of reasoning that involves inferring which of several explanations for particular observed facts is the most compelling one) for conducting elicitation, analysis, and specification of requirements for a VR Product. These bare-minimum attributes are building blocks for understanding VR products. They are detailed as follows:

- **Scene Properties**: Capturing the 3D environment properties is key for any VR product. It is required to understand the dimensions, degree of freedom, and boundaries of the play area of a 3D environment from the respective stakeholders. For example, different variants can offer a VR therapy product for overcoming Glossophobia or fear of public speaking [113]. The VR community should understand the scope and degree of the scene to deliver potential layouts considering the requirement.

- **Articles**: These are the assets or objects of a 3D environment that play a significant role in the interaction. For example, a VR training product for facial surgery requires specific medical hand-held tools to interact with the persona in the scene. VR community should be able to record and co-relate the intent and usage of these tools for executing a surgery training scene.

- **Action-Responses**: VR community must research the possible actions between the articles and participants in a scene and their potential responses to one another. These action responses between the available articles in a VR scene could be synchronous and asynchronous. This defines the behavior of an article in a given scene. For example, a VR product for early diagnosis of dementia patients [272] requires a careful understanding of the articles and dementia participant traits to address the diagnosis. The interactions of such participants could be sensitive and discreet.

- **Acoustics**: audio is a vital attribute of a 3D Environment that defines the authentic aesthetics of a scene. It provides an advantage to the scene while addressing the intent of the scene. For example, a VR Tour guiding product [167] aims to provide the remote aesthetics of a tour. The acoustic effects of a tour, monument, or place will elevate the participant's experience in such a VR scene. VR community should obtain an abstract view of such acoustic effects while working with the stakeholders during the elicitation.

The above-defined bare-minimum set of attributes is inevitable and necessary for the elicitation process for VR products. All other attributes can be considered add-ons and may add more richness to the VR product. The following attributes can be considered as necessary and sufficient for elicitation of requirements for a VR product.

- **Control-flow of Scene**: It is significant to register the expected journey of a scene across a timeline from the stakeholders. A constructive control flow of the overall scene will position the scope and limits of a VR product. The observed methods - Story boarding, Role Play Flow Analysis, and Mental model technique help VR community elicit the control flow of the scene to some extent. For example, a VR product that aims to construct an image-based spatial presence of virtual participant experience [180] requires eliciting the possible journey of a participant for a quality VR scene.

- **Action-flow of Articles**: Synchronous and Asynchronous flow of events in a 3D scene will help VR community define the articles' timeline in a given VR scene. Article action flow is crucial to define the behavior of a scene. For example, VR products that provide massively multi-user interactions [122] require eliciting all possible action events of all the articles in the scene to generate potential response outcomes.

- **Data-flow**: Logging the data in all levels of VR scene execution is optional. However, understanding the data flow is required in the case of VR products that need to judge participants' journeys in the scene. For example, VR products that implement a patient-specific routine for robotic needle insertion during surgeries [259] require eliciting a sequence of events and their respective data points to evaluate the correctness of needle insertion.

Overall Observations: Based on our observations from previous section, methods like *Domain Expert Review, Direct Inspection, Survey Questionnaire, and Use case-based Studies* are widely used for their ease of execution. It is easier and faster to elicit a VR product using these methods. However, it requires multiple iterations to reach precise specifications as most of the target products are domain-specific. Based on our observations from previous section, VR community have developed new methods like *Situated Cognitive method, Participatory Workshops, Structured Interviews, Behavioral questionnaires, and Role Play Flow Analysis* to avoid multiple iterations. These methods help VR community to elicit requirements briefly, eventually reaching detailed specifications of a VR product.

Most of these methods emerge from conventional software engineering practices and show some results that are difficult to determine if they positively or negatively impact VR product development. Despite the method, VR community must thoroughly understand VR as a domain to elicit, analyze and specify requirements. Our study reveals less evidence that observed methods do not imply domain understanding and are generic in practice. This implies more responsibility on requirement analysts to capture proper and correct requirements in the context of the VR Domain. The elicitation output of experienced VR community is anticipated to be more productive than a naive VR community. This is a severe gap within the participant's expertise and should be addressed for better requirement gathering. Despite the expertise of the VR community, the method should be able to provision the bare-minimum aspect of the VR domain on gathering requirements. In such cases, a human-in-a-loop may confuse VR developers as the requirements become unclear. There is an excellent scope for introducing a domain-centered model-backed requirement-gathering method(s) that can help VR requirement analysts to gather requirements irrespective of their level of expertise. Almost all the observed methods do not address this gap. New tools and methods are to be formulated to address this gap and improve the requirement-gathering process for VR product development.

Scope and Extent of Automation: Overall, the RE Methods for elicitation, analysis, and specification for VR products discussed in previous sections are primarily manual. Few of them are domain-specific, and few others are laborious and require significant human effort. There is negligible possibility of automating the observed methods. Automated RE methods may minimize the chances of errors, leading to enhanced accuracy. Although this was not the initial criterion for our inquiry in this study, we believe this is an important aspect and could be of interest to the VR community. While automation of RE methods enabled conventional software practitioners in Test case generation [217] and Reuse and configuration [161] etc., it also facilitates software practitioners to accelerate model-driven development [154] of software products. However, based on our observations - such automated practices are yet to be developed and adopted by the VR community.

Here, automation can be defined in two-fold i.e., firstly - *automation of the RE method itself* and secondly - *using any given RE method in an automated development pipeline*. Considering the complexity of VR Software, the automation of RE methods may require detailed ideation and further research. We may also need metrics to define the quality and traceability of the requirements generated through these methods. However, using existing RE methods for VR in an automated development pipeline seems feasible if the output of existing RE methods practiced by the VR community is formalized using a minimal set of elements and attributes illustrated in previous section. In that case, we can facilitate model-driven development for VR software. Text-to-image models like DALL.E [220] can play a pivotal role in easing VR product development and help mitigate challenges illustrated in part of this chapter.

Conclusion: This chapter presents our detailed observations from the systematic mapping study conducted to understand requirement engineering practices in virtual reality. Overall, we infer that almost all the methods observed across these thrust areas are well known to the conventional software engineering domain and, in most cases, are force-fitted to address the need of the hour. There has yet to be an attempt made by the VR community to evaluate the impact, influence, and relevance of these methods on VR technology in practice. This is primarily because of an unclear understanding of Virtual Reality as a technology domain and what it takes to create novel methods that operate for VR software products. These observations are closer to our conclusions from Chapter 3. This motivated us to examine the bare-minimum attributes of a VR technology domain to develop novel methods for the VR community.

Table 4.2 Requirement Elicitation method mapped with respective reference

Field of Interest	Requirement Elicitation Method	About the Product/Prototype	Reference
Analytics	Direct Inspection	Data Visualization	[239]
Automotive	Direct Inspection	Simulator based Automotive UI	[40]
	Domain Expert Review	Train Wagon Maintenance Simulator	[189]
Construction	Domain Expert Review	Building Modelling in Construction Industry	[71]
	Domain Expert Review	AR for Construction Control	[252]
	Domain Expert Review	VR for Safety Training	[280]
	Domain Expert Review	City Model Planner	[233]
Design	Direct Inspection	Collaborative Design Setup	[39]
	Usecase Studies	Collaborative Virtual Environment	[37]
	Domain Knowledge	UML using VR	[195]
	Domain Knowledge	VR Content Authoring	[242]
	Storyboarding	Design and Code Review within Teams	[237]
	Usecase Studies	Intelligent Interactive and User Cognition	[207] [243]
	Mental model technique	VR for Spatial Presence Experience	[211]
	Survey Questionnaire	Arousal and Valence Studies for VR	[177]
	Direct Inspection	Haptic Editor for Immersion Experience	[295]
	Use case Studies	Roaming in VR Scene	[99]
	Subjective Inspection	VR for Haptic Feedback	[222]
Education	Prototyping	Collaborative Learning VR App	[127]
	Domain Expert Review	Industrial Training	[122]
	Core Task Analysis	Maintenance Training VR App	[157]
	Usecase Studies	Virtual Class Services	[43]
	Situated Cognitive method	Learning App for Non Natives and Low Literates	[109]
	Serial Inspection	SE Process Education	[278]
	Survey Questionnaire	MR for Higher Education	[149]
Entertainment	Direct Inspection	Second Life 3D Platform	[64]
	Survey Questionnaire	VR Shopping	[273]
	Survey Questionnaire	Personalized Virtual Worlds	[238]
	Structured Interviews	Virtual Museum and Tour Guiding	[18] [167]
	Domain Expert Review	Teaching Virtual Indonesian Musical Instrument	[200]
HealthCare	Domain Expert Review	Medical Planning for cardiovascular disease	[272]
	Domain Expert Review	Abdominal Cancer Surgery Training	[102]
	Participatory Studies	Arm and Hand Rehabilitation therapy	[26]
	User Centered Approach	Maxillo-Facial Surgery Training	[180]
	Domain Expert Review	Endoscopic Surgery Simulation Training	[55]
	Case Studies	Interactive Stroke Rehabilitation	[128]
	Practitioner's Review	Dental Simulation Studies for Dentists	[61]
	Domain Expert Review	Physiotherapy Motion Games and Spinal Cord	[184] [133]
	Psychological theories	Early Diagnosis of Dementia	[197]
	Practitioner's review	Traditional Foot Reflexology Study	[58]
	Participatory Workshops	Promoting Health Education in Brazil Country Side	[75]
	Domain Expert Review	Dental Anesthesia Training	[201]
	Domain Expert Review	Virtual Pediatric Patient Training System for Nurses	[259]
	Domain Expert Review	Robotic Needle Insertion Training and ADHD	[108] [253]
	Likert Based Survey	Health Care Education	[168]
	Practitioner's Review	Physical impact on Cycling in Older Adults	[162]
	RE-FIT Method	VR for Special Needs	[274]
Social	Use case Studies	Multi-User Virtual environments	[36]
	Use case Studies	Social VR using Sensors System	[235]
	Case Studies	Virtual Negotiation Training	[101]
	Domain Expert Review	Social Skills for Autistic Children	[63]
	Behavioral Questionnaire	Therapy for Social Phobias	[100]
	Survey Questionnaire	Social VR	[52]
	Domain Knowledge	Campus Virtual Tour	[279]
	Role Play Flow Analysis	Role-play in VR Game for Elders	[277]
	Subjective Inspection	VR for Glossophobia	[91]
Space	Domain Expert Review	JET Flight Remote System	[292]
	Domain Expert Review	In-Vessel Visual Inspection System	[113]
	Direct Inspection	Space Simulation Studies	[163]

Table 4.3 Requirement Analysis methods mapped with respective Domain and its reference

Requirement Analysis Method	VR Product/Prototype	Field of Interest	Reference
Thematic analysis	Vehicle Pedestrian Communication for Vision Impaired pedestrians	Automotive	[56]
Flow Chart	Virtual Campus System	Design	[287]
Gap Analysis using CAD	Engineering Design Review		[284]
Physical Mock-Up analysis	Design Review and FEMA Analysis System		[34]
Object Oriented method	Electrovibration Haptic Feedback in VR		[299]
Object-Oriented Domain centered method	Human Anatomy Puzzle	Education	[206]
Functional flow charts	Computer Hardware Assembly VR Training Scene		[263]
UML	Conceptual Modelling		[37]
Object-Oriented method	Note taking Interface in VR Learning Environment		[53]
Use case flow diagrams	VR Campus tour		[23]
Use case flow diagrams	Basic Electronics training boards		[132]
Heuristic - Human centered approach	Computer Assembly VR Training		[226]
Workflow Technique	micro-robotic cell injection training system	HealthCare	[80]
Heuristic - Human centered approach	Train Patients with Dementia to accomplish daily routines		[224]
Object-Oriented method	Exposure Therapy for acrophobia		[16]
Sequence Diagram	Virtual Surgery Simulation		[27]
Control Flow Chart method	Diagnosing Dyslexic Children Potential		[106]
Object-Oriented method	Ankle Rehabilitation		[81]
Object-Entity modeling	Endotracheal Intubation Simulation Training		[218]
Focus group studies	Motor-cognitive exergames for neuro-related diseases		[225]
Prototype/Physical model	Rehabilitation VR Serious Games		[78]
Event Flow Graph	Complex Commercial Manufacturing Systems	Manufacturing	[62]
Iterative - Control flow study	Automotive Simulation Feedback tool		[10]
Abstract-to-Concrete Gap Analysis	Mechatronic Assembly Simulation Scene		[244]
Control Flow Chart method	Special Equipment Inspection System		[112]
Object-Oriented method	Welding Training Simulation		[291]
Strategic Dependency Model (i* star)	Multi-agent 3D Game for Cognitive Simulation		[9]
Flow Chart	Battlefield Simulation	Military	[297]
Object-Oriented method	Virtual Archery Experience	Recreation	[95]
Object-Oriented method	First Person Shooting with self-transforming controllers		[152]
Flow Chart	Virtual Kiosk		[230]
Use Case flow model	Archery Instruction		[209]
Object-Entity modeling	Rescue Training in Earthquake ruins	Simulation	[300]
Grant Chat flow	Industry robotics simulation		[72]
Flow Chart Analysis	Cruise Ship Travel Training	Tourism	[256]
Physical model Adoption	Interaction System for Digital Heritage		[45]

Table 4.4 Requirement Specification methods mapped with respective Domain and its reference

Requirement Specification Method	VR Product/Prototype	Field of Interest	Reference
CSDF (Common scene definition framework)	Sample VR Scene	Design	[32]
Contextual VR Prototyping	Sample VR Game		[171]
mental model technique	Navigation Spatial Presence		[180]
UML and legacy SRS Documentation	SE Presentations in VR	Education	[214]
UML and Design Research Document	Programming Education using VR		[13]
Templatized Specifications	VR Foot Reflexology Therapy	HealthCare	[194]
Plain Documentation	Defence Training Simulation for Multiplayer Shooting	Military	[129]
Functional SRS Documentation	VR Relax-Refresh System	Recreation	[55]
Functional SRS Documentation	Halden Reactor Project	Simulation	[96]
Task Tree based Documentation	Space Debris Education	Space	[57]

Chapter 5

Understanding Virtual Reality Domain through a Model Template

This chapter presents our journey toward illustrating the bare-minimum aspects of the Virtual Reality technology domain through a meta-model from a software engineering perspective. This will aid the VR community in developing novel tools and formulating methods across all the observed thrust areas to ease their overall VR software product development experience.

5.1 Attributes for VR Software Requirement Specification

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 [156]. 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. Several attempts are made to standardize VR as a domain through standards like VRML (Virtual Reality Modeling Language) by W3C - 1997 [269], COLLADA (COLLAborative Design Activity) by Sony - 2004 [120], O3D by Google - 2010 [119] etc. These standards have failed due to a lack of interoperability support of VR software with VR Hardware [42]. Following are few detailed insights on these prominent standards and the reasons for adoption failure.

VRML (Virtual Reality Modeling Language) [269] is a file format for describing three-dimensional (3D) interactive vector graphics, designed particularly for the web. It allows the creation of 3D scenes and objects to be viewed in an web browser or a standalone VRML browser, called a "player". VRML files can be created and edited using various 3D modeling software, and can include textures, lighting, physics properties, animations, and behaviors. VRML has been replaced by X3D (ISO/IEC 19775-1) as the standard for 3D web graphics.

O3D was a web-based 3D graphics API developed by Google as an alternative to Flash and other 3D technologies. It was designed to enable the creation and display of interactive 3D graphics in web browsers, without the need for additional plugins or software. O3D provided a JavaScript API for creating and manipulating 3D scenes, objects, and animations, and supported features such as lighting, texturing, and physics. O3D was intended to make it easier for web developers to create and share 3D content, and to provide a more immersive and interactive web experience for users.

COLLADA (COLLAborative Design Activity) is a royalty-free, XML-based file format for 3D digital assets. It was developed by the Khronos Group, an industry consortium focused on the creation of open standards for 3D graphics, and is used for exchanging 3D models and scenes between different applications and platforms. COLLADA supports a wide range of 3D modeling features, including geometry, textures, lighting, animations, and physics. It is designed to be platform-independent and can be used to exchange 3D assets between different operating systems, hardware platforms, and software applications. COLLADA files can be exported and imported by a variety of 3D modeling tools, game engines, and other software, making it a popular choice for interoperability in the 3D graphics industry. COLLADA is often used for creating and sharing 3D models and scenes for use in real-time 3D applications, such as games, simulations, and virtual reality experiences. It is also used in the film and television industry for creating 3D visual effects and animations.

Despite initial interest and support, all of these standards failed to gain widespread adoption due to a number of challenges as discussed below:

- **Limited browser support:** At the time of VRML's and O3D's introduction, few web browsers supported the format, making it difficult for users to view and interact with VRML content.
- **Complexity:** COLLADA is a complex and powerful file format that supports a wide range of 3D modeling features. This complexity can make it difficult for some users and developers to work with, and can lead to issues with interoperability between different applications and platforms.
- **Performance issues:** VRML and O3D files could be large and complex, leading to slow loading times and poor performance on many systems.
- **Lack of standardization:** VRML lacked a clear and consistent set of standards, making it difficult for developers to create consistent and reliable content.
- **Limited authoring tools:** There were few user-friendly tools available for creating VRML content, making it difficult for non-technical users to create and share their own 3D models.
- **Competition:** VRML faced competition from technologies like Flash and Java 3D, while O3D was challenged by WebGL, Three.js, and Unity. Similarly, COLLADA competed with other 3D file formats such as FBX and glTF. These alternatives offered similar capabilities and gained wider adoption, reducing the appeal of VRML, O3D, and COLLADA.
- **Limited use cases:** VRML and O3D was primarily used for creating 3D models and environments for the web, and did not have a wide range of other applications, limiting its appeal.
- **Lack of community and developer support:** COLLADA and O3D did not have a strong community or developer support, making it difficult for the technology to gain momentum and attract a critical mass of users and developers.

On other end, formats like FBX (Filmbox) and glTF (GL Transmission Format) and gaining wide popularity as their proprietary in nature.

FBX (Filmbox) is a proprietary file format used for exchanging 3D digital assets between different applications and platforms. It was developed by Kaydara and owned by Autodesk Inc. was used for exchanging 3D models, animations, and other assets between different 3D modeling tools, game engines, and other software. FBX supports a wide range of 3D modeling features, including geometry, textures, lighting, animations, and physics. It is designed to be platform-independent and can be used to exchange 3D assets between different operating systems, hardware platforms, and software applications. FBX files can be exported and imported by a variety of 3D modeling tools, game engines, and other software, making it a popular choice for interoperability in the 3D graphics industry. It is often used for creating and sharing 3D models and scenes for use in real-time 3D applications, such as games, simulations, and virtual reality experiences. It is also used in the film and television industry for creating 3D visual effects and animations. As it is proprietary format, it is not open to the public. Thus users must have a license to use the FBX software development toolkit and underlying tools for working with this format. However, FBX is widely supported by many 3D modeling tools and game engines, and is a popular choice for interoperability in the 3D graphics industry.

glTF (GL Transmission Format) on other end is an open, royalty-free file format for 3D digital assets. It was developed by the Khronos Group, an industry consortium focused on the creation of open standards for 3D graphics, and is used for exchanging 3D models and scenes between different applications and platforms. glTF is designed to be lightweight and efficient, with a focus on real-time 3D applications, such as games, simulations, and virtual reality experiences. It supports a wide range of 3D modeling features, including geometry, textures, lighting, animations, and physics, and is designed to be platform-independent and easy to use. glTF is an open standard, which means that it is freely available to the public and can be used and implemented by anyone. It is supported by a wide range of 3D modeling tools, game engines, and other software, and is becoming an increasingly popular choice for interoperability in the 3D graphics industry. glTF is often compared to other 3D file formats, such as COLLADA and FBX, which offer similar capabilities. However, glTF has several advantages over these formats, including its lightweight and efficient design, its focus on real-time 3D applications, and its open and royalty-free status. As a result, glTF is gaining popularity as a modern and flexible file format for exchanging 3D assets between different applications and platforms.

In contrast, despite being proprietary (in case of FBX) and open, royalty-free (in case of glTF) they are not widely acceptable due to volatility in their underlying meta-model of Virtual Reality technology domain. For example, consider a human that has little or no awareness of the interference [156]. Such a human through a software system aspires to simulate a real-world environment as an immersive 3-Dimensional (3D) environment, primarily through 3D computer-generated graphics. Various human sensory aspects like auditory, haptic (through force), olfactory, vision, human motor, proprioception (body position and movement), and cognitive capacities are experienced through this system. *Immersion* and *Presence* are ideal states of a VR software system but are not necessary prerequisites for engagement and interaction.Despite the rise in computation power of VR hardware, VR software systems do not adequately achieve realism through underlying VR software. VR hardware and VR Software have evolved independently, leading to severe platform fragmentation. Such challenges still continue with FBX and glTF standards. Recently attempts were made to support interoperability between VR software and hardware using OpenXR standard APIs [97].

OpenXR is an open standard for virtual reality (VR) and augmented reality (AR) devices, developed by the Khronos Group, an industry consortium focused on the creation of open standards for 3D graphics. It provides a common, cross-platform API (Application Programming Interface) for accessing and controlling VR and AR devices, allowing developers to create applications that can run on a wide range of devices without the need for custom integration.It is initially designed to simplify the development and deployment of VR and AR applications, by providing a single, unified API that can be used across different devices and platforms. This allows developers to create applications that can run on a wide range of devices, without the need to write custom code for each device or platform. OpenXR is an open standard, which means that it is freely available to the public and can be used and implemented by anyone. It is supported by a wide range of VR and AR device manufacturers, software companies, and other organizations, and is becoming an increasingly popular choice for cross-platform development in the VR and AR industry. OpenXR is still in the process of being developed and is not yet a final, stable standard. However, it is expected to be released in the near future, and is already being used by some developers and organizations for cross-platform VR and AR development.

Overall, the available standard are either in draft or 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. Thus, we address the problem by formulating the following research questions:

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

In the following section, we discuss our study setup, data analysis, and our theory on creating a meta-model for VR software systems to avoid platform fragmentation and portability issues.

5.2 Role Based Model Template for Specifying VR Software

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 [159] [245] suggested superficial information on the working of a typical VR software system. Most secondary studies present VR from an application point of view with no precise details on underlying aspects of VR as a domain. 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 [114] to investigate components of bare-minimum VR software systems.

Socio-Technical Grounded Theory (STGT) is a modern version of traditional sociological Grounded Theory methodology, specifically designed for Software Engineering and other socio-technical domains. It is based on over 15 years of experience and combines socio-technical principles with grounded theory methods to explore the intersection of technology and society. STGT is intended to provide increased clarity and flexibility in its methodological steps and procedures. [114] It is an iterative and incremental research method using available resources with abductive reasoning for theory development. As per the STGT approach, the following particulars represent the boundaries of our work:

- **Domain & Actors:** VR is our domain, and the VR practitioners who take part in VR software development are considered as actors. Here VR Designers, Developers, Acoustic Engineers, VFX Artists, VR Testers, UX Engineers, Release Engineers, etc. are considered as VR practitioners.
- **Phenomenon:** To a meta-model for VR to illustrate a bare-minimum VR software system
- **Data/Tools/Techniques:** Qualitative data collected through informal interviews, VR SDKs and VR Standards.
- **Researcher:** This study is reviewed and managed by VR Experts who act as researchers in this study.

Below are the detailed steps that are required to be applied to conduct the STGT in practice:

- **Define the Socio-Technical Research Context:** This involves understanding the phenomenon being studied, the domain and actors involved, the researcher's role, and the nature of the collected data.
- **Conduct Basic Data Collection:** Collect data through various methods, such as observations, interviews, and document reviews, focusing on the socio-technical aspects of the research context.

- **Analyze Data:** Analyze the collected data using inductive and deductive reasoning, identifying themes, patterns, and concepts related to the research question.
- **Develop Emergent or Structured Theory:** Develop a theory based on the analyzed data, using either an emergent or structured approach, depending on the nature of the research question and the data collected.
- **Iterate and Refine:** Iteratively repeat the previous steps, refining the theory and updating the analysis as new data is collected and analyzed.
- **Report Findings:** Document and report the findings, ensuring that the research process and the developed theory are transparent and accessible to others.

Based on STGT approach, we defined our research context as follows. We present our additional observations in the following sub sections.

Research Context: *The proprietary standards failed to create cross-platform support for VR software driving the VR technology domain into a platform-dependent system.*

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 desired 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?

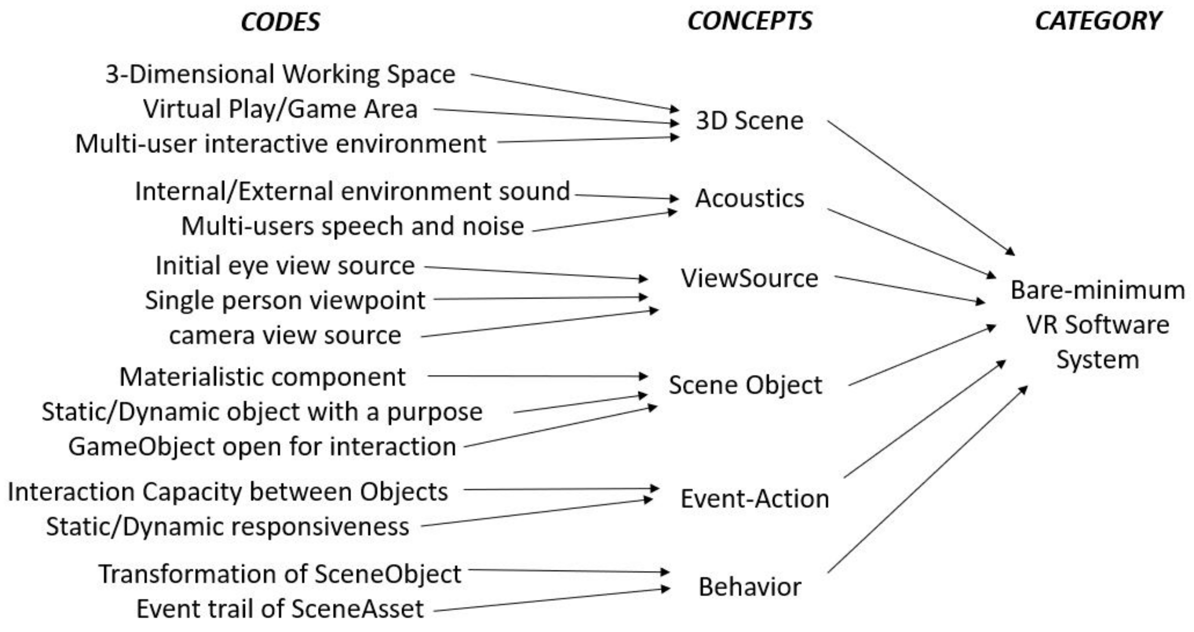
VR SDKs Selection: Our interactions with VR practitioners led us to review widely used VR SDKs like UNITY3D [7], Unreal Game Engine [8], CryEngine [5], AframeJS [4], and Amazon Lumberyard [6]. We examined SDK source code, underlying classes, and white papers. All other non-technical proprietary materials are excluded from examination.

VR Standards Selection: Interactions with VR practitioners led us to explore VR standards in detail. We considered prior standards - VRML [269], X3D [285], WebXR, O3D [119], and prevailing standards - OpenXR [97] 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.

Data Analysis

We used the Open Coding method [255] 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 5.1 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 5.1 Common Codes from Interview Study


CODES
CONCEPTS
CATEGORY
3-Dimensional Working Space
Virtual Play/Game Area
Multi-user interactive environment
3D Scene
Internal/External environment sound
Multi-users speech and noise
Acoustics
Initial eye view source
Single person viewpoint
camera view source
ViewSource
Materialistic component
Static/Dynamic object with a purpose
GameObject open for interaction
Scene Object
Interaction Capacity between Objects
Static/Dynamic responsiveness
Event-Action
Transformation of SceneObject
Event trail of SceneAsset
Behavior
Bare-minimum VR Software System

Observations

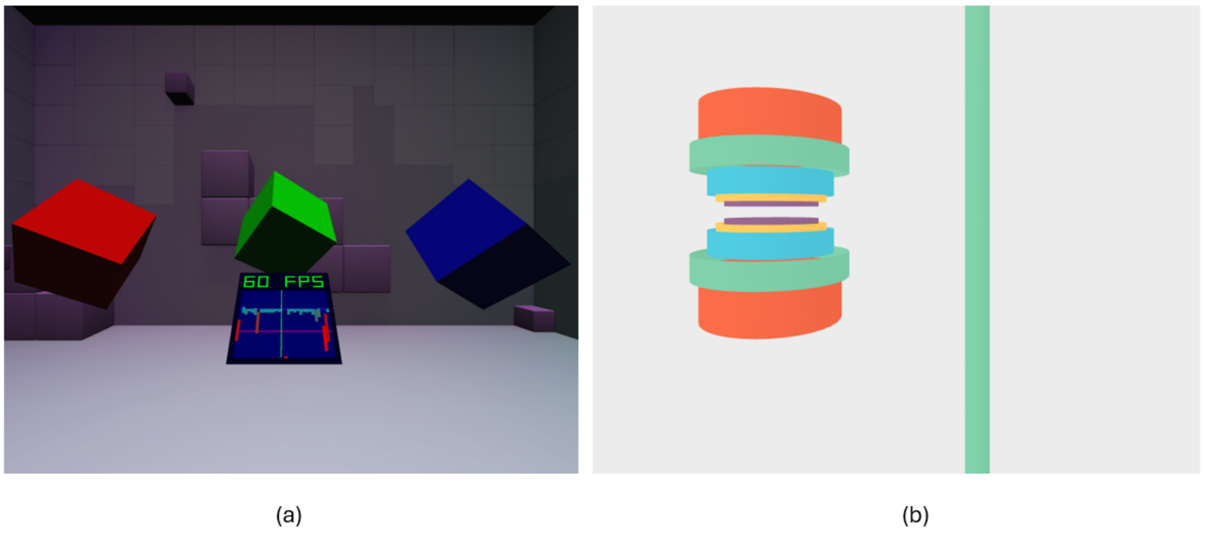
Based on the codes generated using open-coding, we used *Abduction Reasoning* [114] to theorize a meta-model for a bare minimum VR software system. Abductive reasoning helps researchers conduct data analysis through different means such as hunches, clues, metaphors or analogy, symptoms, patterns, and 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 along its state 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.
- ***Timeline*** - All the possible events are categorized into synchronous events and asynchronous events that are based on a trigger i.e. due to an external stimuli. There is possibility of a non-trigger based synchronous and asynchronous events that are within the article and will not be based on a trigger i.e. not due to an external stimuli.

The culmination of the above concepts varies between various VR standards and VR Software Development toolkits(SDKs) based on their levels of abstraction and nomenclature. However, the overall concepts described above are the bare-minimum set to construct a VR software system and they can be observed consistency across various VR standards and VR Software Development toolkits(SDKs) in different forms. To understand how SDKs perceive the same underlying meta-model elements differently, we present examples using WebXR API[1] and AFrameJS[2] VR Scenes. Fig 5.2 contains a WebXR scene[3] as **(a)** that presents 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

[1]https://immersiveweb.dev/
[2]https://aframe.io/docs/1.5.0/introduction/
[3]https://immersive-web.github.io/webxr-samples/input-selection.html?usePolyfill=0

Figure 5.2 Example VR Scene built using WebXR API and AFrameJS



another end, an AframeJS scene [4] **(b)** presents a no-audio 3D scene with a centered view-source, with un-sized 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 prevailing browsers no longer support this standard. 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.

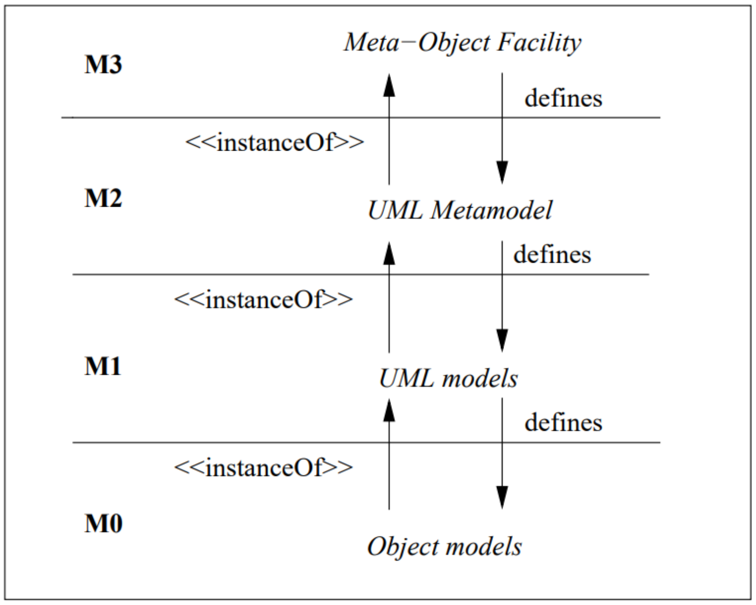
5.3 Virtual Reality Software System Meta Model

The Unified Modeling Language[5] is a programming language with an independent notion for specifying, visualizing, constructing and documenting systems. It is an Object Management Group (OMG) standard language for object-oriented modeling. The UML infrastructure is defined as a four-layer architecture as shown in Fig 5.3

- **Level M3 (meta-metamodel layer)** defines a language for specifying metamodels. The meta object facility is an example of meta-metamodel

[4]https://aframe.io/aframe/examples/animation/warps/
[5]https://www.uml.org/

Figure 5.3 UML four layered meta-model architecture

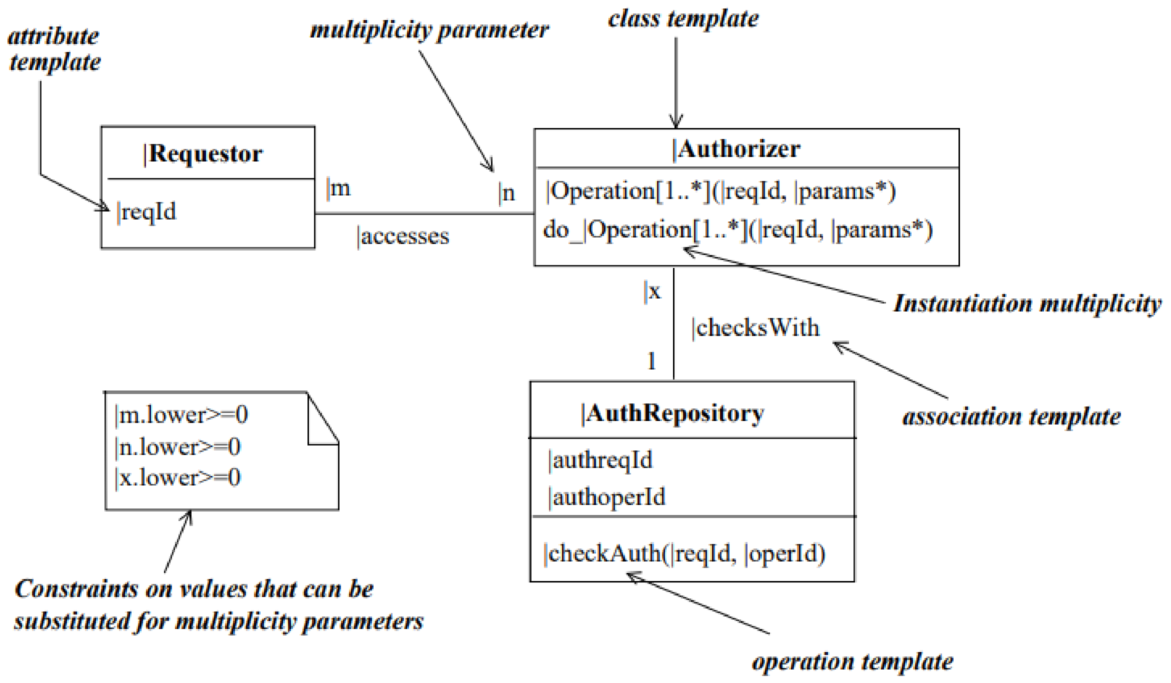


- **Level M2 (metamodel layer)** contains models that specify modeling languages. The UML metamodel and the common warehouse metamodel (CWM) are the examples of metamodels.
- **Level M1 (model layer)** contains models that describe semantic domains. The model layer consists of models expressed in languages specified by M2 meta-models.
- **Level M0 (user data)** consists of object configuration specified by the models at level M1.

The UML was designed primarily as a notation for modeling a single application and its use to model application families in problematic. The Role based meta-modeling language (RBML) is an UML based language extension that supports rigorous specification of patterns that characterize a family of design models proposed Kim et al. [145]. As RBML uses UML syntax, UML tools can be used to create RBML specifications. A RBML specification consists of set of role models that describe pattern properties from different perspectives with additional details. A variant of the RBML that facilitates generation of compliant models from pattern descriptions is used in this work. Template diagrams rather than role models are used to specify families of models. The UML models are obtained from template diagrams by binding the template parameters to actual values.

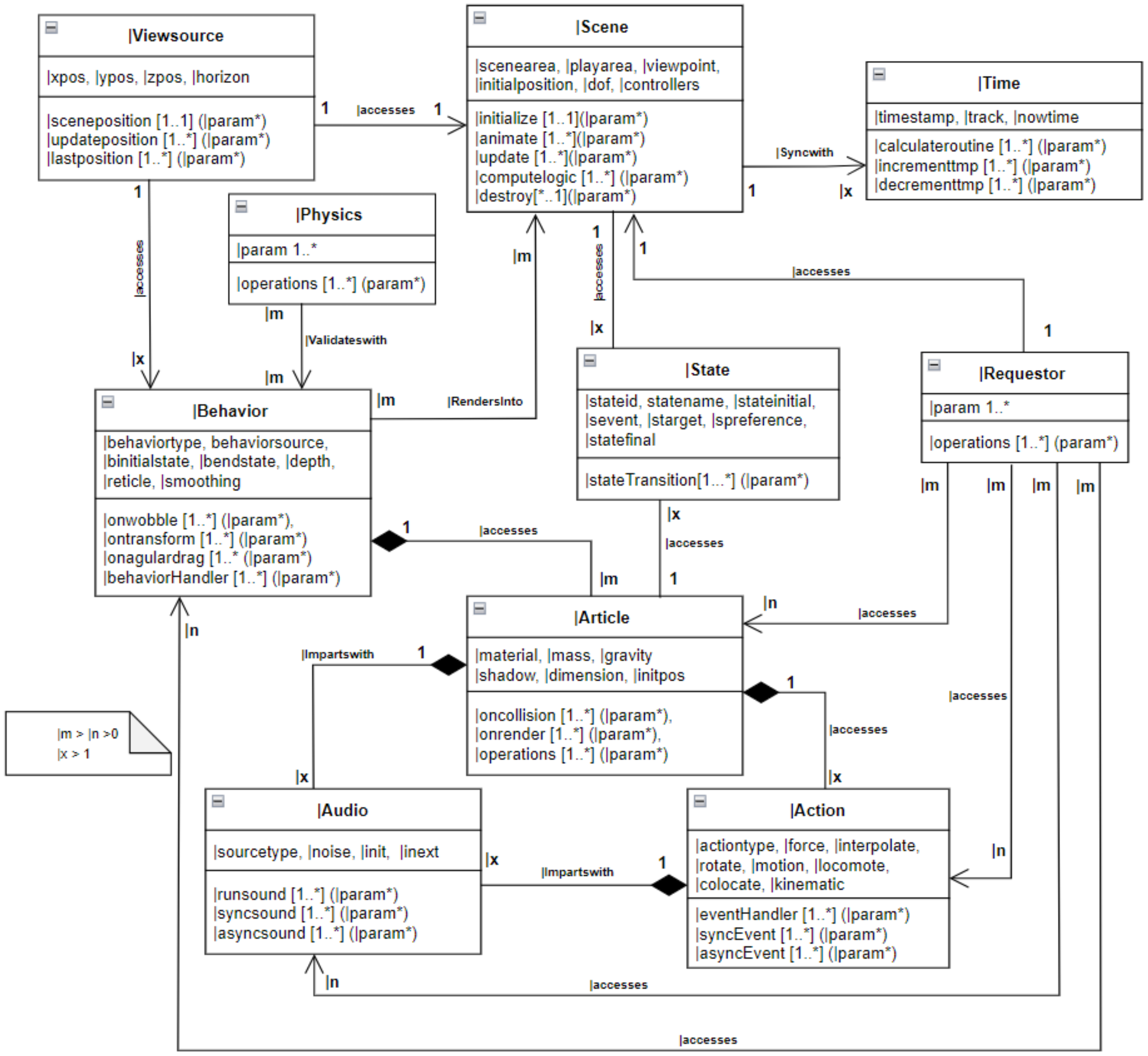
A **Role** represents a specific set of responsibilities, behaviors, or functions within a system or model. It defines expected behaviors and interactions without specifying the concrete implementation. Roles are typically more abstract and focus on the purpose or function an entity fulfills in a given context. In contrast, A **Template** is a predefined structure or pattern that can be reused and instantiated multiple times. It provides a blueprint for creating consistent instances of a particular element or component. Templates are more concrete and focus on the structure and attributes of an entity. In practice, we can use both roles and templates as part of a meta-model, where *roles* help define the abstract behaviors and interactions between entities and *templates* to provide concrete implementations or structures for those roles. For example, a **Developer** role to represent the responsibilities and interactions of a developer in the process, **DeveloperTemplate** to define the specific attributes and structure of a developer entity in your system. This combination allows the model both the dynamic, behavioral aspects (with roles) and the static, structural aspects (with templates) of your system. As the focus is on illustrating static and structural aspects of VR Software System, as part of our work - we use the notion of templates to describe the meta-model of VR technology domain. Fig 5.4 illustrates an example of a class diagram us-

Figure 5.4 RBML Class diagram of model template for requestor authorization workflow



ing RBML specification to describe a meta-model system that authorizes a requestor by identifying the desired operation from the authorizing repository. Here |**Requestor**, |**Authorizer** and |**AuthRepository** are class templates. |**Requestor** carries an attribute called |reqId as part of the class attribute template. |**Authorizer** carries an attribute function called |**Operation** with parameters |**reqID** and |**params***, i.e., any additional domain specification parameters are allowed with a multiplicity of one to many along with instantiation multiplicity through **do**_|**Operation** as part of its attribute template. |**AuthRepository** class template carries |**authreqId** and |**authoperId** as part of the attribute template and |**checkAuth** with parameters |**reqId** and |**operId** as the operation template. The |**Requestor** and |**Authorizer** class templates are associated with the association template called |**accesses** with multiplicity parameters such as |**m** and |**n** where |**m.lower** $>=$ **0** and |**n.lower** $>=$ **0.** |**Authorize**r and |**AuthRepository** class templates are associated with an association template called |**checksWith** with multiplicity parameters such as |**x** and **one** where |**x.lower** >= **0.** The meta-model (M2) from Fig 5.4 will aid technology domain-specific software practitioners to adopt and execute on underlying applications the require authorization workflow as of their model instances (M1) to eventually develop application object models (M0).

Figure 5.5 Role based Model Template of a bare minimum VR Software System


|Viewsource
|xpos, |ypos, |zpos, |horizon
|sceneposition [1..1] (|param*)
|updateposition [1..*] (|param*)
|lastposition [1..*] (|param*)
|Scene
|scenearea, |playarea, |viewpoint, |initialposition, |dof, |controllers
|initialize [1..1](|param*)
|animate [1..*](|param*)
|update [1..*](|param*)
|computelogic [1..*] (|param*)
|destroy[*..1](|param*)
|Time
|timestamp, |track, |nowtime
|calculateroutine [1..*] (|param*)
|incrementtmp [1..*] (|param*)
|decrementtmp [1..*] (|param*)
|Physics
|param 1..*
|operations [1..*] (param*)
|Behavior
|behaviortype, behaviorsource, |binitialstate, |bendstate, |depth, |reticle, |smoothing
|onwobble [1..*] (|param*),
|ontransform [1..*] (|param*)
|onagulardrag [1..* (|param*)
|behaviorHandler [1..*] (|param*)
|State
|stateid, statename, |stateinitial, |sevent, |starget, |spreference, |statefinal
|stateTransition[1...*] (|param*)
|Requestor
|param 1..*
|operations [1..*] (param*)
|Article
|material, |mass, |gravity
|shadow, |dimension, |initpos
|oncollision [1..*] (|param*),
|onrender [1..*] (|param*),
|operations [1..*] (|param*)
|Audio
|sourcetype, |noise, |init, |inext
|runsound [1..*] (|param*)
|syncsound [1..*] (|param*)
|asyncsound [1..*] (|param*)
|Action
|actiontype, |force, |interpolate, |rotate, |motion, |locomote, |colocate, |kinematic
|eventHandler [1..*] (|param*)
|syncEvent [1..*] (|param*)
|asyncEvent [1..*] (|param*)
|accesses
|Syncwith
|Validateswith
|RendersInto
|Impartswith
|m > |n >0
|x > 1
1
|m
|n
|x

Based on Role Based meta-modeling language, we present a role-based model template to illustrate meta-model for **Virtual Reality** as Technology Domain. Figure 5.5 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 5.5, |***Scene***, |***Viewsource***, |***Time***, |***Behavior***, |***Physics***, |***Requestor***, |***Article***, |***Audio***, |**State** and |***Action*** are called as class templates. Each class template consists of two sections, the attribute template and the operation 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)** with their |***State*** and respective |***Behavior*****(s)** through a |***ViewSource*** onto |***Scene***. |***ViewSource*** initiates all other class templates asynchronously with |***Time*** .

Fig 5.6 illustrates the UML four-layered meta-model architecture deduced by considering the model template for the VR technology domain from Fig 5.5.

- **Level M3 (meta-metamodel layer)** defines a language for specifying meta models, i.e., RBML meta-object factory. It helps us illustrate class templates, attribute templates, operation templates, association templates, multiplicity parameters, and Instantiation multiplicity.

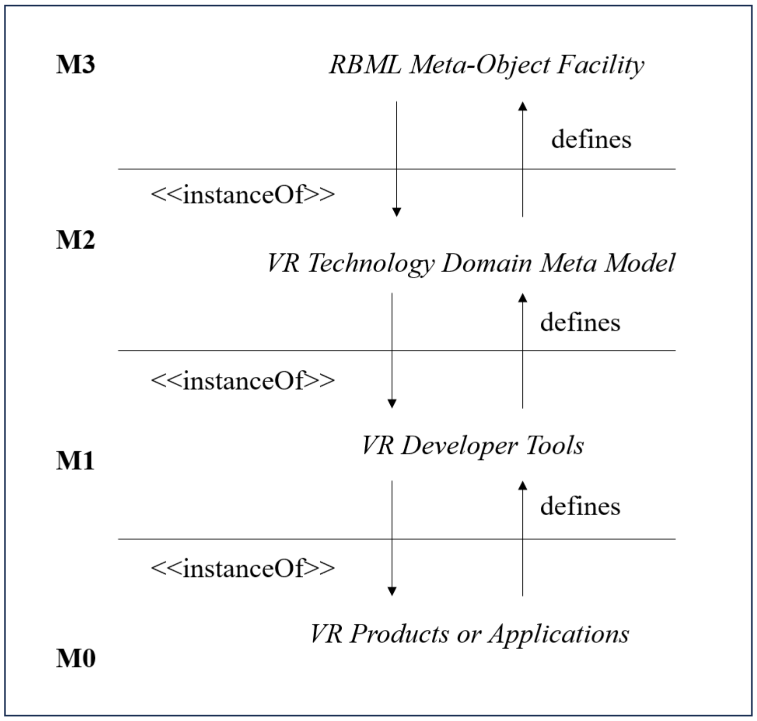
- **Level M2 (metamodel layer)** contains models that specify VR software system metamodel. It describes the class templates, attributes, associations, and multiplicity parameters related to the VR Technology domain.

- **Level M1 (model layer)** Using the metamodel, we create tools associated with the VR technology domain, like Software development Toolkits, domain specification languages, productivity tools for the VR community to ease design, testing, coding, and release, etc. **Example:** OpenSource VR Engine, Model-based VR TestCase Generation tools, Model-based Design Editors.

- **Level M0 (end-user application)** This layer presents the end-user applications developed using the tools developed by the VR community. **Example:** Flight simulator, Virtual Compliance Training of Safely Bio-Critical Lab.

To understand the instantiation of the VR meta-model into the VR application, we bring an example that provides bindings between the meta-model and its related application using a virtual reality soccer training application developed by Rezzil Inc. Fig 5.7 provides us a VR scene where the soccer trainee is required to perform a task called **"Kick"**. Table 5.1 illustrates the bindings for a test case to verify the **action:kick** and the response **audio:splash** associated with **article:ball**. The example bindings can be used as part of a novel VR Test case generator tool. This table provides a correlation between the meta-model attributes and a potential test-case generator application-specific attributes for a VR soccer training application 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.

Figure 5.6 UML four layered meta-model architecture for Virtual Reality Domain



5.3.1 Extending VR Meta-model to Domain-specific and Context-specific applications

The role-based model template for a bare-minimum VR software system can be extended to an application centered on a domain or to a context within a given domain. In the context of software applications, a domain refers to the specific field or area of expertise that the software is designed to serve. It represents the concepts, processes, and rules that define the problem space the software aims to solve. Application domains vary widely, ranging from healthcare and finance to gaming and social media. Domains are often associated with specific industries or business processes and may involve specialized terminology and workflows. Software developers use domain models to represent these domains and create software solutions tailored to their needs. By understanding the domain, developers can design context-specific applications and effectively address the challenges users face in that field. As shown in Fig 5.8, we can visualize the top-down stack of layers using the VR model template for domain-specific and context-specific applications by extending underlying attributes from bare-minimum attributes to domain-specific and context-specific attributes.

For better illustration, let us consider x represents the bare-minimum attributes of the VR model template called V^0 where $V^0(x)$ represents the collection of bare-minimum attributes that belong to the VR model template. Now y represents the customized attributes with an extended role-based model template called V^1 where $V^1(y)$ represents the collection of extended attributes belonging to the extended VR model template. Thus, as per equation 5.1, the collection of bare-minimum attributes from the

Figure 5.7 VR Soccer Training Application by Rezzil Inc.

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	*

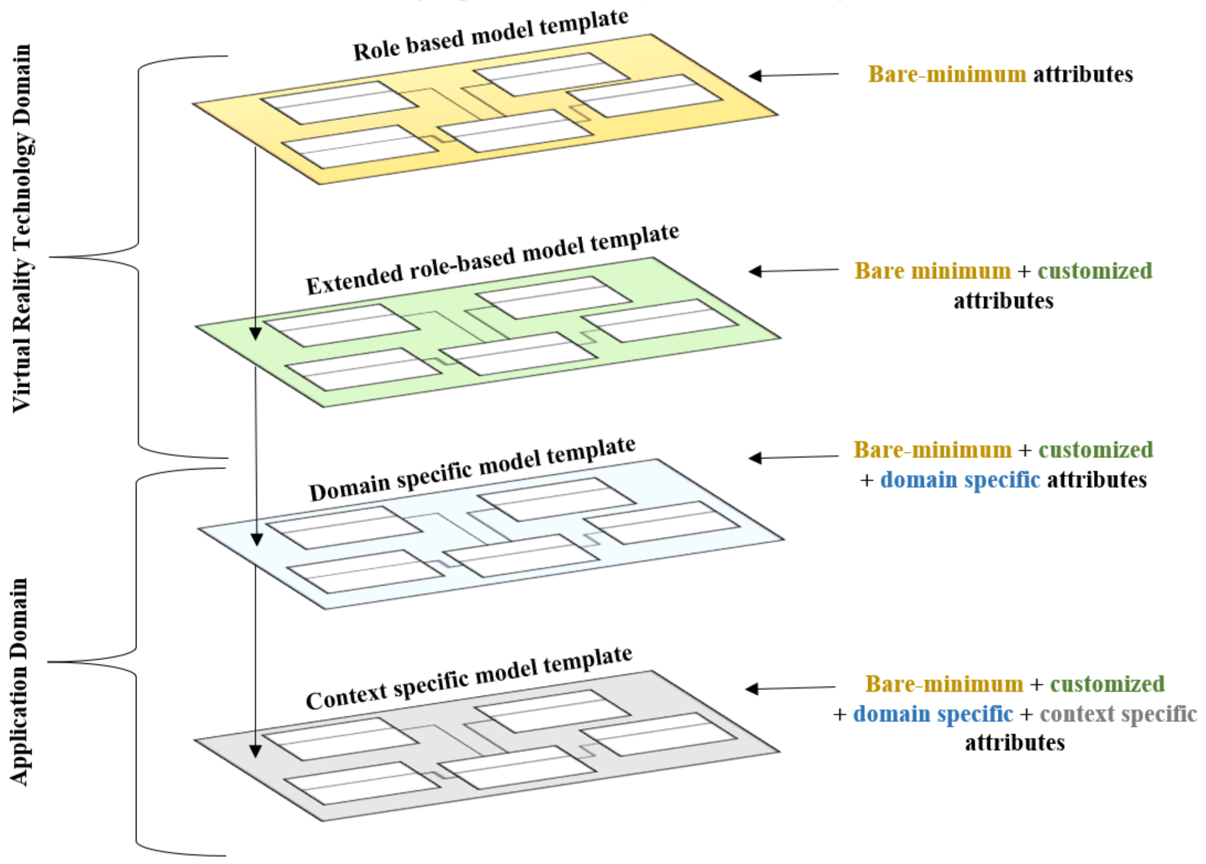
Table 5.1 Example bindings for a test-case of kicking the ball using template class

VR model template and customized attributes together now constitutes the overall extended role-based model template.

$$V^0(x) \cup V^1(y) = V^{01} \tag{5.1}$$

For example, |material is one of the bare-minimum attributes from the |Article class of the VR model template from Fig 5.5 belonging to $V^0(x)$. This is a bare minimum attribute as it is generic across all the VR developer engines. However, the customized attributes like |roughness, not a bare-minimum property of a |material but a unique property of specific VR engines belonging to $V^1(y)$, can be included to extend the bare-minimum model template. In contrast to the Domain-specific model templates V^2, these are further extended by including domain-specific attributes represented by z where $V^2(z)$ is the collection of all such domain-specific attributes belonging to the domain-specific VR model template as shown in the equation 5.3.

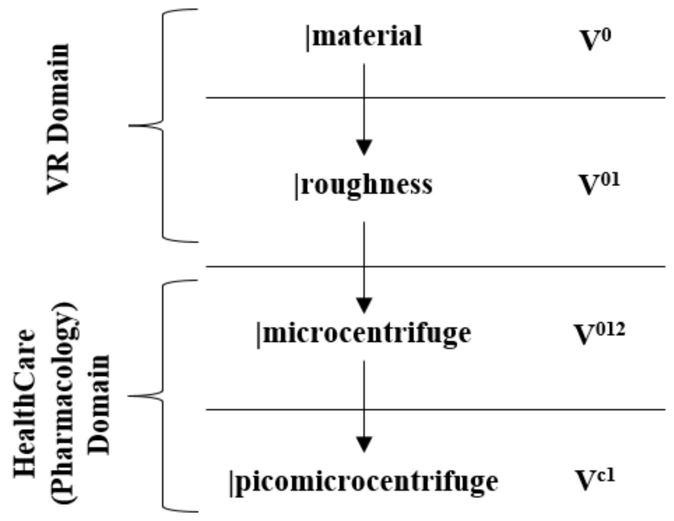
Figure 5.8 Visualizing top-down stack layers on extending VR meta model


Virtual Reality Technology Domain
Application Domain
Role based model template
Extended role-based model template
Domain specific model template
Context specific model template
Bare-minimum attributes
Bare minimum + customized attributes
Bare-minimum + customized + domain specific attributes
Bare-minimum + customized + domain specific + context specific attributes

$$(V^0(x) \cup V^1(y)) \cup V^2(z) = V^{012} \tag{5.2}$$

For example, consider the pharmacology domain in healthcare, where a domain-specific practitioner (pharmacologist) uses various chemicals to develop, identify, and test the drugs to cure, treat, and prevent diseases. The **microcentrifuge** is one of the many apparatus they rely on to spin down and separate chemical particles in small volumes of liquid samples. Consider a pharmacology domain-specific VR scene developed to train pharmacologists; a pre-curated domain-specific article |**microcentrifuge** will be equipped we fixed |**material** properties that are unique to a pharmacology domain. Similarly, the |**microcentrifuge** can be used for different context-specific use cases, such as the molecular separation of cell organelles (like various nuclei or DNA or RNA), phenol extraction, and studying the effects of drugs on various biological systems. Suppose a VR training scene is developed for pharmacologists on such context-specific applications V^c . In that case, VR practitioners must rely on context-specific knowledge and standard operating procedures in such use cases to design a VR application. In such cases, context-specific attributes are to be used as part of VR application development, which is still an extension of the domain-specific model template. As shown in equation 5.3, if V^c is a context-specific

Figure 5.9 Visualizing example of extending VR meta model to HealthCare (Pharmacology) Application Domain



application, it constitutes model attributes of V^{012} which are from domain-specific model template and combination of various contexts i.e. V^{c1}, V^{c2}, V^{c3} ... V^{cn} where n can be of infinite contexts for a given application domain.

$$V^c = V^{012} + V^{c1} + V^{c2} + ... + V^{cn} \;\; where \;\; 0 < n < \infty \tag{5.3}$$

Fig 5.9 illustrates this example by presenting the model attributes in layers. This figure shows that |**material** is a bare-minimum model attribute (V^0), and |**roughness** is a custom model attribute related to the VR technology domain (V^{01}), |**microcentrifuge** is a pharmacology domain-specific model attribute (V^{012}), a sub-domain of the healthcare domain. |**picomicrocentrifuge** is a context-specific domain, a specific model attribute (V^{c1}) that belongs to a specific use-case of pharmacology experiments, which inherits the properties of |**microcentrifuge** but maintains distinctive properties in practice. Thus, it is a unique context-specific model attribute of the pharmacology domain. Thus, a role-based model template for a VR software system can be extended to a customizable, VR technology domain-centered model template. It can be extended further to a domain-specific application and a context-specific application within a given application domain.

5.3.2 Implications of VR Model Template for VR Community

Studies have shown that enterprise VR product development differs from traditional software development [139]. 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 technology domain and help practitioners create novel open-source tools to support and ease overall 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.

Conclusion: A role-based model template as a meta-model for a bare-minimum VR software system is critical to building future developer tools for VR software. 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 develop generic open-source tools to ease VR product development and address platform fragmentation between VR hardware and software. Based on the formulated role-based model template for Virtual Reality products, we aim to develop a novel tool to address the thrust areas identified in the previous chapter. Specifically, in the following chapter, we use this VR technology domain meta-model to develop a requirement specification tool to ease the overall requirement-gathering process and improve the end-to-end VR product delivery.

Chapter 6

Model Based Requirement Specification for Virtual Reality Software

Traditionally VR software product development involved writing programs to generate synthetic worlds using geometric primitives and simulated physics [155]. Such programs are hardware dependent, highly customized in design, not scalable, not portable and have limited usage as they are highly contextual to a particular use-case. For example, software simulation applications to train users on complex tasks in military combat, space exploration, etc. Modern VR software products have evolved from standardized software-development toolkits (SDK) that support portability and re-usability of geometric primitives along with the applicable physics laws. Most of these SDKs like UNITY3D [7], Unreal Game Engine [8], CryEngine [5], AframeJS [4], and Amazon Lumberyard [6] etc. are influenced by computer graphics-based game engines predominantly used for Two-Dimensional (2D) and Three-Dimensional (3D) Game development, and are now repackaged as VR SDKs. Studies have shown that most of the current VR software developers are from the gaming and visual effects industry [15] and primarily rely on Game-Development processes to build VR Products. In our previous work, we conducted a multi-year empirical study [138] to understand the development methods adopted by the VR community. As part of this empirical study, we studied 1600+ VR community through VR Conferences, VR Developer Meetups, VR Online Community Forums, and other VR enthusiasts. We used focused interviews with diverse sample sets using a purposeful sampling approach, online forum thread study using interaction analysis model (IAM), and Online Survey. Our empirical study concludes that VR product development is primarily different from conventional software product development due to difference in Multimodality Challenges, Volatile Hardware, Platform Fragmentation, Diverse Stakeholders, Software Portability issues, Difficulty in achieving Realism and Poor Tool Support. In contrast, Requirements Engineering is one of the key thrust area of research as it has poor tool support [138].

Does rigorous requirement engineering matter for VR software development? - A typical VR scene revolves around various entities like scene terrain, assets involved in the scene, action responses between the assets, custom behaviors, timeline, design, control flow of the overall scene, acoustics, physics, light source, reflexes of the participants, and many more. Gathering the requirements of all these entities is a laborious task. The requirement analyst should always be cautious about the control flow and data flow of VR scenes correlated with the rise in the scene's length and the extent of the entity's usage. In contrast, our prior empirical study [138] shows that RE methods for conventional software product development do not sufficiently work for VR as they do not consider the aspects of personalized human experiences and extent of realism in the diverse entities involved in VR products. Let us consider an

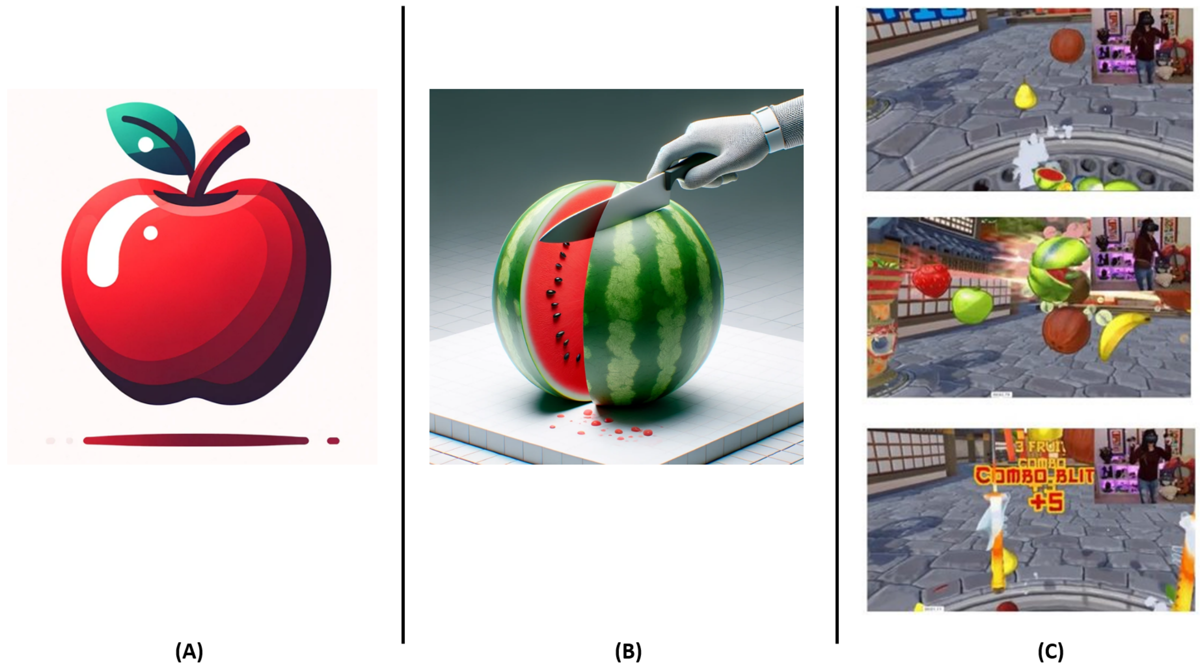


Figure 6.1 Understanding rigour of Requirement gathering for 2D vs 3D Animation Vs 3D Environment
Note: These images was generated with the assistance of AI

example to understand the rigor behind requirements gathering in VR software. Fig 6.1 illustrates three different types of virtual assets they are the 2-Dimensional non-animated static image of an Apple labeled as (A) and (B) is a 2-Dimensional dynamic animated graphic, which is an illustration of cutting a watermelon, (C) is a 3-Dimensional virtual dynamic environment of a fruit crush game where a player is supposed to thrash fruits in the digital world to gain points. To develop (A), we need properties like Color, Texture, and Dimension of the asset. In the case of (B), we may need additional properties along with previous properties like Cast Shadow, Point-of-View position, minimal interaction, and partial dynamic in user response. However, in the case of (C) - it required capturing many more properties like layout information, scale, material, gravity, mass, audio, ray-casting, Object OnCollision, Object Rigidness, light sources, motion, rotation, occlusion, angular drag, Object Kinematic properties, Interpolation of an object, Vobble effect of Object, VR participant locomotion, environment depth, scene rendering, synchronous-asynchronous VR participant task-action-responses, control flow and data flow of overall events. Comparatively, the sophistication levels of requirements are too high in (C) than in (A) or (B) for low-fidelity games like Fruit Ninja. As a result, the requirements Engineering process is complex due to the extent of realism involved in VR software products.

Recent studies [190] [191] show that most design authoring tools for VR software that consider requirements as input where input is expected in a specified format. This can cause a loss of information flow into the design leading to confusion. Authoring tools need to be customizable to allow different parameters as inputs to facilitate multiple interaction flows. However, these tools overlook may desirable aspects of VR interaction design. It was also observed that a given authoring tool may or may not capture all possible requirements elicited by the requirement analyst. The VR prototype engineer must rely on multiple authoring tools to arrive at a minimum viable product for VR development teams. Even in the case of our multi-year empirical study on understanding VR product development being different from conventional software product development, current VR community predominantly elicit and specify VR product requirements in plain English using an unstructured template. They were also found to use methods like Task-Tree and Event-Action with less precision [138]. While capturing requirements in such simple ways is inefficient, it is desirable if clear guidelines, templates, and traceability mechanisms are established to track the changes. Practices that require manually intensive efforts degrade the quality of the VR Scene and eventually increase the delivery cost due to loss of information during the requirements analysis process. *All the above observations motivated us to develop a requirement specification tool for VR community to ease VR product development.*

6.1 Authoring Tools vs Requirement Specification Tools

VR Authoring Tools: - An authoring tool is a software application or platform that enables users to create content in a cohesive and interactive format, including text, code, graphics, audio, and video. These tools simplify content creation, making it accessible even to those without advanced technical skills. In the context of virtual reality technology domain, VR authoring tools are explicitly created to develop interactive and immersive virtual reality (VR) content. They enable users to develop VR experiences without requiring extensive programming knowledge. They offer an intuitive user interface, integration of multimedia assets, support for a wide range of VR devices, support collaborative features across various VR developer engines, and integration support to external software like learning management systems to host VR content. Sometimes, VR authoring tools cater to different expertise levels, offering pre-made interactions and scenes for non-specialists or customizable content for specialists. Unfortunately, most VR authoring tools do not incorporate authoring tools as an integral feature in the VR developer tools. Due to serious platform fragmentation issues within the VR technology domain, individual VR practitioners develop custom plugins to streamline the VR content to modify and distribute across distinct VR developer engines. Examples of VR authoring tools include ***AirVu Authoring Tool, LearnBrite, Storyflow, and CenarioVR***. These tools cater to various industries, such as education, healthcare, manufacturing, and customer service, providing customizable solutions predominantly for developing immersive VR training modules. It is pretty rare to observe a VR authoring tool that conventional VR practitioners can use as a general-purpose tool. This is primarily because the artifacts

Table 6.1 Comparision between Authoring Tools and Requirement Specification Tools

Feature	Authoring Tools	Requirement Specification Tools
Focus	Implement the requirements into Design or Mockups or Workflows	Specify, tracking and maintain requirements
Target audience	Designers, Developers, and subject matter experts	Project managers, Technical stakeholders and Customers
Key functions	Structured workflows, attribute assignment, requirement hierarchy	Traceability, change management, version control, specification
Use cases	Early stage of design and development	Throughout the development process
VR Tool Examples	AirVu Authoring Tool, LearnBrite, Storyflow, CenarioVR	Task Tree Templates, Conventional SRS Templates

generated from such VR authoring tools may not work across different VR developer engines and do not support multiple VR devices.

VR Requirement Specification Tool: Requirement specification tools help manage, organize, and document software requirements throughout the development of software products. Their primary goal is to ensure that all stakeholders involved in a software project have a clear understanding of the requirements and that the final product aligns with the initial vision. Such tools are required to support the creation of requirements, provide traceability, collaboration, change management, the ability to prioritize requirements, version control of requirements, etc. In the context of virtual reality technology domain context, there isn't a dedicated requirement specification tool that can suffice such VR technology domain-specific needs. Due to the lack of VR-centered requirement specification tools, the VR community is heavily relying on conventional approaches like Software Requirement Specification Templates, Task tree based Requirement specification templates etc.

As shown in Table 6.1, we illustrate the differences between authoring and requirement specification tools. In brief, authoring tools are for authoring and organizing content based on requirements, while requirement specification tools are for managing and tracking requirements throughout development. Also, the authoring tools focus on clarity and conciseness, while requirement specification tools provide a more comprehensive approach to requirements management. As there is a need for such comprehension in the VR technology domain, we developed a model-based requirement specification tool. In the following subsection, we provide more details about our novel requirement specification tool that can help requirement analysts specify requirements with higher precision and clarity.

6.2 VReqST - To specify VR Requirements

The **VReqST**, i.e., *"Virtual Reality Requirement Specification Tool,"* is developed for requirement analysts of the VR community to specify requirements using a role based model template of VR Software System [140]. VReqST will aid requirement analysts to describe Scene properties, Article properties, Actions - Responses between these articles in the given scene along with their state change, custom behaviors (like algorithms, logic interpretation, etc.) that are to be executed in a defined timeline. Our first iteration for VReqST was oriented towards a formal-specification language with pre-defined code constructs. We exchanged our early ideas with requirement analysts from the VR Community through various platforms. Based on the feedback from VR community, we simplified it into a model template tool instead of evolving it into a formal-specification language. In contrast, VReqST behaves as an informal form of a specification language. In the following sub-sections, we present the details about different model template(s), their underlying validator(s), the architecture, tool overview and workflow of VReqST.

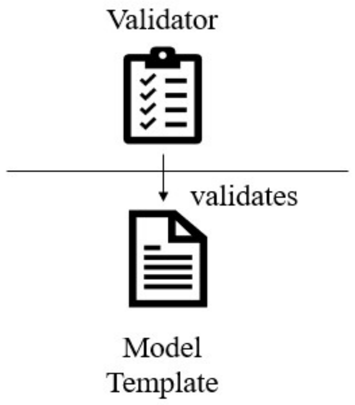


Figure 6.2 Model Template and Validator

6.2.1 Model Template and its Validator

As mentioned in previous Section, model templates are extensions of model concepts defined as part of the role-based model template for VR Software Systems [140]. 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. Table 6.2 provides the mapping of model concept and their model attributes. In case of VReqST, we represent each model concept as a model template file. They are represented in JSON (JavaScript Object Notation) format. The model template file contains model attributes, a minimum set required to capture requirements for VR Software products as shown in Table 6.2. For example, the model template file of the model concept called **"Article"** contains one of the model attributes called *"IsKinematic"* - a physics property of 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 kinematic physics property, whereas *'0'* signifies that respective article does not hold kinematic physics property.

All such validation rules for model attributes for a given model template are defined as part of the underlying validator. Each model template file (scene, article, action-response, and timeline) has a distinct validator file to validate the datatype and scope of the model attributes, as shown in Fig 6.2. These validator files are also represented in JSON format. The model template files and their corresponding validator files are available as part of our documentation [135]. Table 6.2 provides details of the default naming convention for the model template file and validator files for respective model concept. The requirement analysts can rely on VReqST documentation to specify requirements in the model template file while eliciting requirements with their stakeholders. These model template files will aid requirement analysts on probing the requirements more accurately.

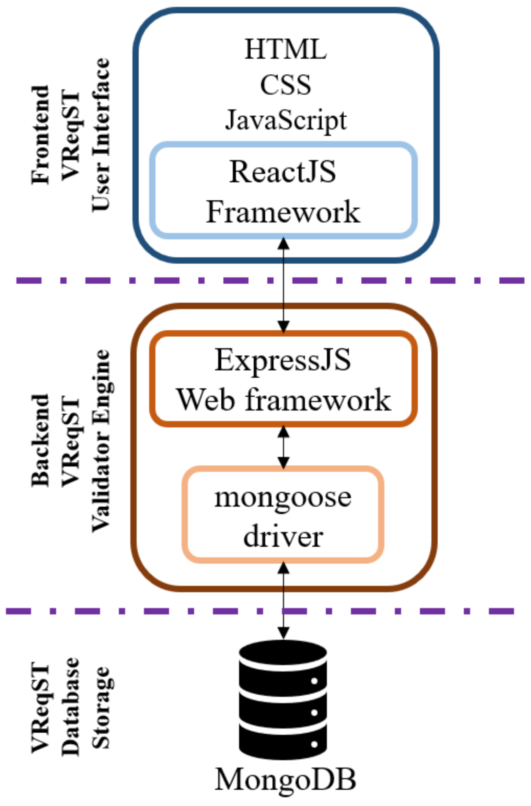


Figure 6.3 VReqST - Architecture

Table 6.2 VReqST - Model Concepts with corresponding Model Attributes

Model Concepts	**Model Template File**	**Validator File**	**Model Attributes**
Scene	scene.json	svalid.json	_scenename,_sid,_slabel,_playarea,#pid,#length_playarea, #breadth_playarea,#height_playarea,#comment, #x_scenecenter,#y_scenecenter,#z_scenecenter, _camera,IsSceneObject,trackingorigin,_initialcamerapos, _viewport,_clippingplane,_horizon,_dof,_skybox, _controllers,_gravity,_interaction,_nestedscene,_audio, _timeline,_Opttxt1,usertype,uplayarea,initialupos, uplayareacenter
Article	article.json	artvalid.json	_objectname,_sid,_slabel,_IsHidden,_enumcount,_Is3DObject, HasChild,shape,dimension,IsText,IsText3D,CastShadow, ReceiveShadow,ContributeGlobalIlumination,IsIlluminate, Transform_initialpos,Transform_initialrotation, Transform_objectscale ,repeattransfrom,XRGrabInteractable, XRInteractionMaskLayer,TrackPosition,TrackRotation, Throw_Detach,forcegravity,velocity,angularvelocity, Smoothing_duration,attachtransform,hasmass,dragfriction, angulardrag,Isgravityenable,IsKinematic,CanInterpolate, CollisionPolling,aud_hasaudio,aud_type,aud_src,aud_volume, aud_PlayInloop,aud_IsSurround,aud_Dopplerlevel,aud_spread, aud_mindist,aud_maxdist,_Opttxt1,@context_img_source
Action-Response	actres.json	arvalid.json	actresid,sourceObj,targetObj,IsCollision,response,comment, Syncronous,repeatactionfor
Timeline	timeline.json	tvalid.json	animate_trigSync,tsyncid,tsOntriggertrue,SyncObjList,tSyncNote, animate_nontrigSync,ntsyncid,ntsOntriggerfalse,ntSyncObjList, ntSyncNote,animate_trigAsync,tasyncid,taOntriggertrue, AsyncObjList,tAsyncNote,animate_nontrigAsync,ntasyncid, ntaOntriggerfalse, ntAsyncObjList,ntAsyncNote,routine, routeid,starttime,endtime,order

6.2.2 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. Fig 6.3 illustrates the architecture of VReqST tool. The source code, tool demo, documentation, and sample requirement specification examples are available as part of our resources [135]. Requirement Analysts of VR products are the target users of VReqST. This tool is developed as a web-based tool to aid requirement analysts to author requirements on-demand and refer them to subsequent team with less hassle. This will eventually ease VR designers and developers to develop overall VR software product and trace back to requirements swiftly.

As shown in Fig 6.4, there are two sections in the VReqST tool, i.e., Authoring Wizard and Validator Engine. The requirement analysts are only exposed to the authoring wizard while authoring VR software product specifications, whereas the validator engine is executed in the background. As shown in Fig 6.4, the requirement analyst are required to start with scene-related details, i.e., the scene model template is the first stage of the VR requirement specification. Once the required model attributes of the scene model template are composed, the specification is validated using the scene validator. Upon successful

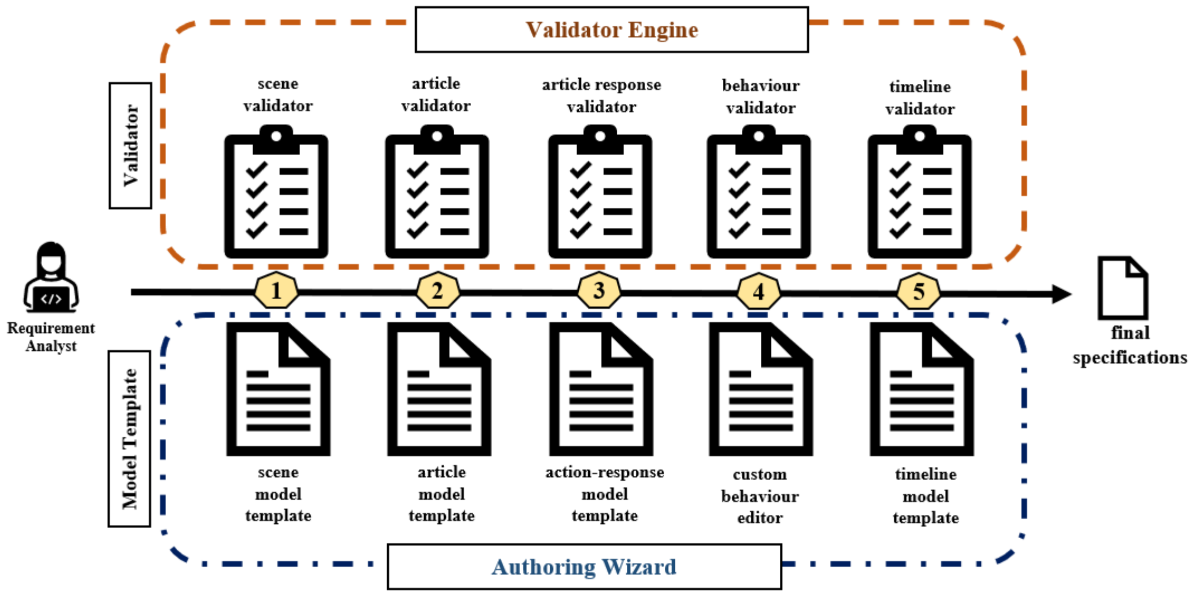


Figure 6.4 Overview of VReqST - Author Journey

validation, the requirement analyst can compose an article model template, a second stage of the VR requirement specification. All the dependent scene model attributes are carried forward to the article model template. The article model attributes are validated using the article validator. Upon successful validation, scene and article-related model attributes are carried forward to the following stage, i.e., action-response, the third stage of VR requirement specifications. The requirement analyst must define all possible action responses between articles that are listed as part of the article model template in second stage. Along with all possible action responses, the state of an article and its initial, final, and transition states are to be defined.

After completing the action-response model template, all the scene, article, and action-response attributes are carried into the customer behavior editor. This is the fourth stage of authoring VR requirement specifications. The custom behavior editor in VReqST will help requirement analysts define behaviors using logic and conditional constructs. For example, in the case of VR games - scoring criteria can be codified as part of this stage. In the case of VR simulation scenes, custom algorithms like co-location or locomotion can be codified. The codified code constructs are validated based on the behavior validator. After authoring behaviors, the requirement analysts are required to present the overall specifications in a timeline. This is the final stage of VR requirement specification. The attributes defined in all the previous stages can be presented as asynchronous and synchronous events. A timeline validator validates the attributes associated with the timeline model template. After completing all the stages, a final specification file combining all the model attributes will be published. This final specification is now available for designers and developers to initiate VR software product development. All the stages in VReqST are linear and cannot be skipped, i.e., the specification will start with a scene, article, action response, behavior, and timeline. We may not be able to specify articles directly without specifying scene information. We can navigate back to the respective stage if validation of a given stage is complete. We shall go through the detailed validation workflow of VReqST in following subsection.

6.2.3 VReqST Validation Workflow

Fig 6.5 illustrates an overall validation workflow of specifications of the model template using the respective validator. We detail this section into two parts: validation of model template files and validation of custom behavior editor

Model Template Validation: Fig 6.5.a provides detailed steps for validating the model template file. There are 4 model templates in VReqST. This applies to the scene, article, action response, and timeline model template files; below are the steps in validating the model template file.

- **Step 1:** The requirement analyst accesses the authoring wizard and chooses a fillable model template (scene, article, action response, or timeline). This fillable model template contains all the model attributes. The requirement analyst must choose the validate option upon filling in all the required details for model attributes in a given model template file.

- **Step 2:** The validate button will convert the filled model template file into an abstract syntax tree with all the model attributes. The validator file contains the validation rules and validates the model attributes in the abstract syntax tree.

- **Step 3:** If the validation is successful, the final model template is processed by the specification processor, where all the previous step's model attributes are carried forward and linked to support traceability. The finalized specifications are published to the requirement analyst. If the validation fails, model attributes are to be revised per the validator logic in the authoring wizard.

Custom Behavior Editor Validation: Fig 6.5.b provides detailed steps for validating the custom behavior editor. As shown in Fig 6.4, Stages 1, 2, 3, and 5 are fillable model template files validated against respective validator files. However, stage 4 in the authoring wizard contains a custom behavior editor that allows requirement analysts to write custom behaviors on articles and their action responses. These behaviors are business rules with expected output for the corresponding output. These business rules are use-case-based and vary from product to product. Following are the steps involved in validating the behaviors.

- **Step 1:** The requirement analyst uses the WRITE wizard to describe the behaviors using inbuilt code constructs like 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 per our documentation [135]. The scene, article, and action response model attributes are made available as part of the WRITE wizard to specify behaviors.

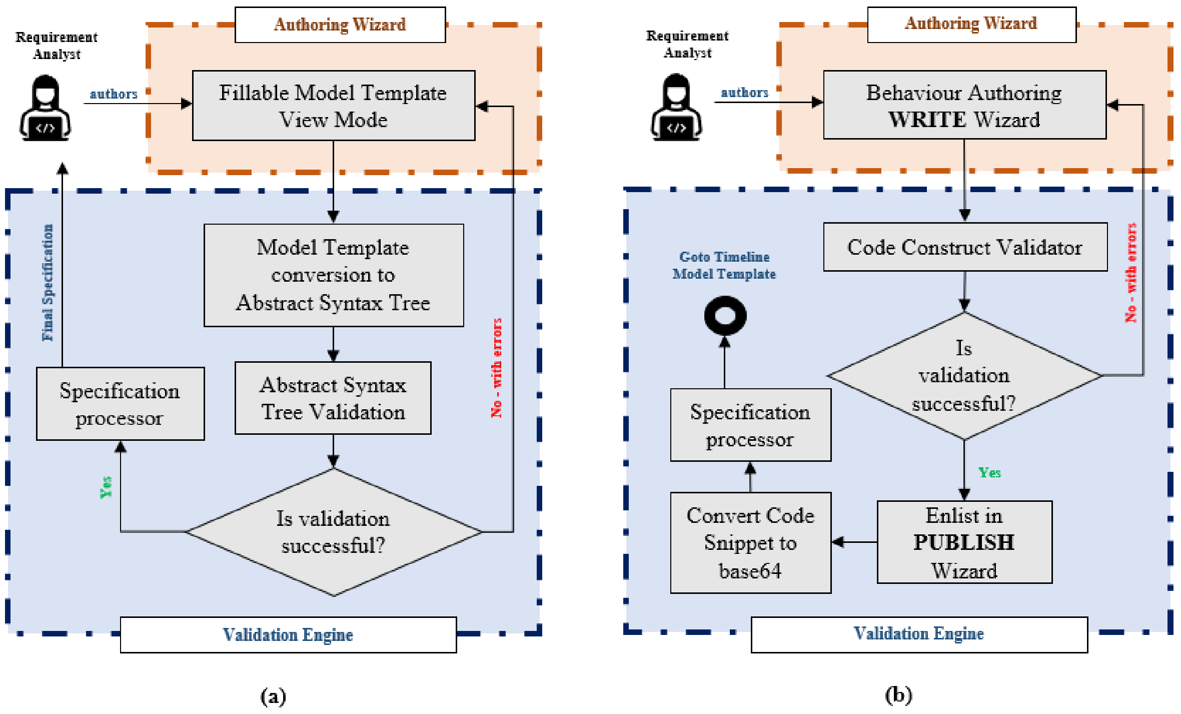


Figure 6.5 Validation Workflow of Model Template and Custom Behaviour Editor - VReqST Tool

- **Step 2:** If the behavior logic is specified per the prescribed code construct, the behavior logic is valid. The valid behavior logic is listed in the READ and PUBLISH wizard of the custom behavior editor. The READ wizard will help requirement analysts review the written behavior logic. Out of all the specified behaviors, the requirement analyst can pick and choose the desired behaviors to be included in the final specification. The PUBLISH wizard will help achieve this task. If the behavior logic is invalid, it is required to be specified again as per the prescribed code constructs defined as part of our documentation [135].

- **Step 3:** The published behaviors are converted into base-64 as multi-line code snippets cannot be stored in JSON object files. All the chosen behaviors from the PUBLISH wizard are ordered under a single behavior file and are stored in the database.

- **Step 4:** All the previous step's model attributes are carried forward and linked with the published behavior file to support traceability. This step is carried forwarded into the timeline model template, which is the final step of the VReqST tool.

6.2.4 Fillable Model Template and Validator Files

Following are the fillable model template file definition and their corresponding validator file definition. Below is the Fillable Scene Model template File.

```
{
   "_scenename":"",
   "_sid":"",
   "_slabel":"",
      "_playarea":{
      "#pid":"",
      "#length_playarea":"",
      "#breadth_playarea":"",
      "#height_playarea":"",
    "#comment":"",
    "#x_scenecenter":"",
      "#y_scenecenter":"",
      "#z_scenecenter":""

    },
   "_camera":
  {
     "IsSceneObject":"",
     "trackingorigin":""
   },
   "_initialcamerapos":{
      "#x_initialcamerapos":"0",
      "#y_initialcamerapos":"5",
      "#z_initialcamerapos":"0"
   },
   "_viewport":{
      "#x_viewport":"",
      "#y_viewport":"",
      "#w_viewport":"",
    "#h_viewport":""
   },
  "_clippingplane":{
  "near_cp":"",
  "far_cp":""
  },
   "_horizon":"",
   "_dof":"",
   "_skybox":"",
   "_controllers":
  {
    "type":"",
    "raycast":"",
    "raydistance":"",
    "raythinkness":"",
    "raycolor":"",
    "raytype":""
    },
   "_gravity":{
     "value":""
   },
   "_interaction":"",
   "_nestedscene":{
      "#value":"",
      "#scenecount":"",
      "#sid_order":""
   },
   "_audio":"",
   "_timeline":"",
   "_Opttxt1":"",
   "@context_mock":"",
   "usertype":[
   {
   "type": "",
   "uplayarea":{
   "#length_uplayarea":"",
     "#breadth_uplayarea":"",
   "#height_uplayarea":""
  },
  "initialupos":{
      "#x_initialupos":"",
      "#y_initialupos":"",
      "#z_initialupos":""
    },
    "uplayareacenter":{
    "#x_uplayareacenter":"",
      "#y_uplayareacenter":"",
      "#z_uplayareacenter":""
    }
   },
    {
   "type": "",
   "uplayarea":{
   "#length_uplayarea":"",
     "#breadth_uplayarea":"",
   "#height_uplayarea":""
  },
  "initialupos":{
      "#x_initialupos":"",
      "#y_initialupos":"",
      "#z_initialupos":""
    },
    "uplayareacenter":{
    "#x_uplayareacenter":"",
      "#y_uplayareacenter":"",
      "#z_uplayareacenter":""
    }
    }
  ]
}
```

Listing 6.1 Fillable Scene Model Template

Below is the Validator File for the fillable Scene Model template File.

```
    {
    "_scenename":{
       "req":"mandatory",
       "typeof":"string",
       "repeat":"notallow",
       "%comment%":"Mention your VR Scene name here"
    },
    "_sid":{
       "req":"mandatory",
       "typeof":"string",
       "repeat":"notallow",
       "%comment%":"A fillable Unique Identifier of the scene"
    },
    "_slabel":{
       "req":"optional",
       "typeof":"string",
       "repeat":"notallow",
       "%comment%":"A fillable Optional text field for scene description in 200
   words"
    },
   "#pid":{
      "req":"mandatory",
      "root":"_playarea",
      "typeof":"string",
      "%comment%":"Unique identifier for playarea in the VR Scene"
   },
   "#length_playarea":{
      "req":"mandatory",
      "root":"_playarea",
      "typeof":"string",
      "repeat":"notallow",
      "%comment%":"Length dimension of the playarea and it is mandatory value tag"
   },
   "#breadth_playarea":{
      "req":"mandatory",
      "root":"_playarea",
      "typeof":"string",
      "repeat":"notallow",
      "%comment%":"Breadth dimension of the playarea and it is mandatory value tag"
   },
   "#height_playarea":{
      "req":"mandatory",
      "root":"_playarea",
      "typeof":"string",
      "repeat":"notallow",
      "%comment%":"Height dimension of the playarea and it is mandatory value tag"
   },

   "#x_scenecenter":{
      "req":"mandatory",
      "root":"_playarea",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"x-axis value of scene center of the playarea and it is mandatory
     value tag"
   },
   "#y_scenecenter":{
      "req":"mandatory",
      "root":"_playarea",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"y-axis value of scene center of the playarea and it is mandatory
     value tag"
   },
   "#z_scenecenter":{
      "req":"mandatory",
      "root":"_playarea",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"z-axis value of scene center of the playarea and it is mandatory
     value tag"
   },


        "IsSceneObject":{
      "root":"_camera",
       "req":"mandatory",
       "typeof":"boolean",
       "repeat":"notallow",
       "%comment%":"true for yes, false for no. This is to set initial camera in
    the scene. Additional attributes include IsSceneObject which holds boolean
    value and trackingorigin holds string value i.e. either floor (physical real-
    world ground) or +/- height from the physical real-world ground. Example: 5+
    floor"
    },
      "trackingorigin":{
      "root":"_camera",
       "req":"mandatory",
      "typeof":"string",
      "repeat":"notallow",
      "%comment%":"true for yes, false for no. This is to set initial camera in the scene. Additional attributes include IsSceneObject which holds boolean value and trackingorigin holds string value i.e. either floor (physical real-world ground) or +/- height from the physical real-world ground. Example: 5+ floor"
    },
    "#x_initialcamerapos":{
      "root":"_initialcamerapos",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"allow",
      "%comment%":"x-axis value of initial camera position of the playarea and it is mandatory value tag"
    },
    "#y_initialcamerapos":{
      "root":"_initialcamerapos",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"allow",
      "%comment%":"y-axis value of initial camera position of the playarea and it is mandatory value tag"
    },
    "#z_initialcamerapos":{
      "root":"_initialcamerapos",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"allow",
      "%comment%":"z-axis value of initial camera position of the playarea and it is mandatory value tag"
    },

    "#x_viewport":{
      "root":"_viewport",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"allow",
      "%comment%":"x-axis value of viewport of the playarea and it is mandatory
   value tag"
    },
    "#y_viewport":{
      "root":"_viewport",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"allow",
      "%comment%":"y-axis value of viewport of the playarea and it is mandatory
   value tag"
    },
    "#w_viewport":{
      "root":"_viewport",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"allow",
      "%comment%":"width value of viewport of the playarea and it is mandatory
   value tag"
    },
    "#h_viewport":{
      "root":"_viewport",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"allow",
      "%comment%":"height value of viewport of the playarea and it is mandatory
   value tag"
    },

   "near_cp":{
      "root":"_clippingplane",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"Used to represent near clipping plane with value ranging from
   0.0 to 1000000"
   },

   "far_cp":{
      "root":"_clippingplane",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"Used to represent far clipping plane with value ranging from 0.0
    to 1000000"
   },

    "_horizon":{
       "req":"mandatory",
       "typeof":"boolean",
       "repeat":"notallow",
       "%comment%":"true for yes, false for no. This is to set horizon sun to map
   the terrian with real world"
    },
    "_dof":{
       "req":"mandatory",
       "typeof":"number",
       "repeat":"notallow",
       "%comment%":"accepts two values 3 and 6. Throw errors for the rest of the
   values"
    },
    "_skybox":{
       "req":"mandatory",
       "typeof":"number",
       "repeat":"notallow",
       "%comment%":"accepts two values 3 and 6. Throw errors for the rest of the
   values"
    },

   "type":{
      "root":"_controllers",
      "req":"mandatory",
      "typeof":"string",
      "repeat":"notallow",
      "%comment%":"To decide the type of controller like Hand, fingure, gesture and
    other types of controllers"
   },
   "raycast":{
      "root":"_controllers",
      "req":"mandatory",
      "typeof":"boolean",
      "repeat":"notallow",
      "%comment%":"It is a Physics function that projects a Ray into the scene,
   returning a boolean value if a target was successfully hit. It is a true or
   false flag"
   },
   "raydistance":{
      "root":"_controllers",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"It is the distance represents the magnitude of the vector from
   the ray’s origin to the impact point. Value 10 is "
   },
   "raythinkness":{
      "root":"_controllers",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"It is the thickness of ray with minimum value as 1 and max value
    as 10"
   },
   "raycolor":{
      "root":"_controllers",
      "req":"mandatory",
      "typeof":"string",
      "repeat":"notallow",
      "%comment%":"It is the color or ray casted in the Scene. Use color name and
   use transparent for no color"
   },
   "raytype":{
      "root":"_controllers",
      "req":"mandatory",
      "typeof":"string",
      "repeat":"notallow",
      "%comment%":"It is a type of ray which can be a straightline, curve line,
 wave etc."
   },
   "value":{
      "root":"_gravity",
      "req":"mandatory",
      "typeof":"number",
      "%comment%":"value of earth gravity. Default is 10."
   },
    "_interaction":{
       "req":"mandatory",
       "typeof":"boolean",
       "repeat":"notallow",
       "%comment%":"This describes the interaction capacity in the scene. True for
 interactions on and False for no interactions"
    },
   "#value":{
      "root":"_nestedscene",
      "req":"mandatory",
      "typeof":"boolean",
      "repeat":"notallow",
      "%comment%":"True if there is a scene within a scene."
   },
   "#scenecount":{
      "root":"_nestedscene",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"Number of scenes as nested scenes"
   },
   "#sid_order":{
      "root":"_nestedscene",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"the order of the nested scenes"
   },
    "_audio":{
       "req":"mandatory",
       "typeof":"boolean",
       "repeat":"notallow",
       "%comment%":"true for spatial audio, false for no spatial audio"
    },
    "_timeline":{
       "req":"mandatory",
       "typeof":"boolean",
       "repeat":"notallow",
       "%comment%":"true for timed scene with dynamic events, false for a static
    scene with static events"
    },
    "_Opttxt1":{
       "req":"optional",
       "typeof":"string",
       "repeat":"notallow",
       "%comment%":"For user optional text"
    },
    "@context_mock":{
       "req":"optional",
       "typeof":"string",
       "repeat":"notallow",
       "%comment%":"For external reference URLs like figma screens or other
    resource links"
    },
     "usertype":{
       "req":"mandatory",
       "typeof":"object",
       "repeat":"notallow",
       "%comment%":"Defines the scene as Single user or Multi-user scene along with
     additional values like uplayarea to define the length, breadth and height of
    the user along with initialupos value in x,y,z coordinates"
    },
    "type":{
  "root": "usertype",
   "req":"mandatory",
    "typeof":"string",
    "repeat":"allow",
    "%comment%":"Is the scene single user type or a multi user type. Use single or
    multi for value tag"
    },
"#length_uplayarea":{
 "root": "uplayarea",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"The length of a user playarea"
    },
    "#breadth_uplayarea":{
 "root": "uplayarea",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"breadth of a user playarea"
    },
    "#height_uplayarea":{
      "root": "uplayarea",
       "proot":"usertype",
       "req":"mandatory",
        "typeof":"number",
        "repeat":"allow",
        "%comment%":"height of a user playarea"
        },
    "#x_initialupos":{
 "root": "initialupos",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"x-axis  of a user in this user play area"
    },
    "#y_initialupos":{
 "root": "initialupos",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"y-axis of a user in this user play area"
    },
    "#z_initialupos":{
 "root": "initialupos",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"z-axis of a user in this user play area"
    },
      "#x_uplayareacenter":{
 "root": "uplayareacenter",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"x-axis initial position of a user in this user play area"
    },
    "#y_uplayareacenter":{
 "root": "uplayareacenter",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"x-axis initial position of a user in this user play area"
    },
    "#z_uplayareacenter":{
 "root": "uplayareacenter",
  "proot":"usertype",
  "req":"mandatory",
    "typeof":"number",
    "repeat":"allow",
    "%comment%":"z-axis initial position of a user in this user play area"
    }

```

```
}
```

Listing 6.2 Scene Validator

Below is the Fillable Article Model template File.

```
{
  "articles": [
    {
      "_objectname": "",
      "_sid": "",
      "_slabel": "",
      "_IsHidden": "",
      "_enumcount": "",
      "_Is3DObject": "",
      "HasChild": "",
      "shape": "",
      "dimension": {
        "dradii": "",
        "dvolumn": "",
        "dlength": "",
        "dbreadth": "",
        "dheight": ""
      },
      "IsText": "",
      "IsText3D": "",
      "lighting": {
        "CastShadow": "",
        "ReceiveShadow": "",
        "ContributeGlobalIlumination": ""
      },
      "IsIlluminate": "",
      "Transform_initialpos": {
        "#x_initialpos": "",
        "#y_initialpos": "",
        "#z_initialpos": ""
      },
      "Transform_initialrotation": {
        "#x_initialrotation": "",
        "#y_initialrotation": "",
        "#z_initialrotation": ""
      },
      "Transform_objectscale": {
        "#x_objectscale": "",
        "#y_objectscale": "",
        "#z_objectscale": ""
      },
      "repeattransfrom": {
        "distfactorx": "",
        "distfactory": "",
        "distfactorz": ""
      },
      "Interaction": {
        "XRGrabInteractable": "",
        "XRInteractionMaskLayer": "",
        "TrackPosition": "",
        "TrackRotation": "",
        "Throw_Detach": "",
        "forcegravity": "",
        "velocity": "",
        "angularvelocity": ""
      },
      "Smoothing": "",
      "Smoothing_duration": "",
      "attachtransform": {
        "#rotate_x": "",
        "#rotate_y": "",
        "#rotate_z": "",
        "#pos_x": "",
        "#pos_y": "",
        "#pos_z": ""
      },
      "XRRigidObject": {
        "value": "",
        "mass": "",
        "dragfriction": "",
        "angulardrag": "",
        "Isgravityenable": "",
        "IsKinematic": "",
        "CanInterpolate": "",
        "CollisionPolling": ""
      },
      "aud_hasaudio": "",
      "aud_type": "",
      "aud_src": "",
      "aud_volume": "",
      "aud_PlayInloop": "",
      "aud_IsSurround": "",
      "aud_Dopplerlevel": "",
      "aud_spread": "",
      "aud_mindist": "",
      "aud_maxdist": "",
      "_Opttxt1": "",
      "@context_img_source": "",
      "state": [
        {
          "stateid": "",
          "statename": "",
          "stateinitial": "",
          "statetransition": [
            {
              "sevent": "",
              "starget": "",
              "spref": ""
            },
            {
              "sevent": "",
              "starget": "",
              "spref": ""
            }
          ],
          "statefinal": ""
        },
        {
          "stateid": "",
          "statename": "",
          "stateinitial": "",
          "statetransition": [
            {
              "sevent": "",
              "starget": "",
              "spref": ""
            },
            {
              "sevent": "",
              "starget": "",
              "spref": ""
            }
          ],
          "statefinal": ""
        }
      ]
    }
  ]
}
```

Listing 6.3 Article Model template

Below is the Validator File for the fillable Article Model template File.

```
{
    "articles": {
        "req": "mandatory",
        "typeof": "object",
        "repeat": "notallow",
        "%comment%": "Array list of Articles that its respective properties"
    },
    "_objectname": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Name of the Object"
    },
    "_sid": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "unique identifier to identify the object"
    },
    "_slabel": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "50 character label to describe the object"
    },
    "_IsHidden": {
        "req": "mandatory",
        "typeof": "number",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if the object is hidden or unhidden"
    },
    "_enumcount": {
        "req": "mandatory",
        "typeof": "number",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Count of objects in the scene"
    },
    "_Is3DObject": {
        "req": "mandatory",
        "typeof": "number",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if the object is 3D or not"
    },
    "HasChild": {
        "req": "mandatory",
        "typeof": "number",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if the object has linked child object"
    },
    "shape": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Shape of the object, mention the geometery type"
    },
    "dradii": {
        "req": "mandatory",
        "typeof": "number",
        "repeat": "allow",
        "proot": "articles",
        "root": "dimension",
        "%comment%": "radius of the object if it is a sphere else NULL"
    },
    "dvolumn": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "dimension",
        "%comment%": "volumn of the object to determine the size"
    },
    "dlength": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "dimension",
        "%comment%": "Length of the object if it has a length property"
    },
    "dbreadth": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "dimension",
        "%comment%": "Breadth of the object if it has a breadth property"
    },
    "dheigth": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "dimension",
        "%comment%": "Height of the object if it has a height property"
    },
    "IsText": {
        "req": "mandatory",
        "typeof": "boolean",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if it is a Text object"
    },
    "IsText3D": {
        "req": "mandatory",
        "typeof": "boolean",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if it is a 3D Text object"
    },
    "CastShadow": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "lighting",
        "%comment%": "Flag to determine if the object can cast shadow"
    },
    "ReceiveShadow": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "lighting",
        "%comment%": "Flag to determine if the object can receive a shadow"
    },
    "ContributeGlobalIlumination": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "lighting",
        "%comment%": "Flag to determine if the object contributes to overall scene
    illumination or self glowing object"
    },
    "IsIlluminate": {
        "req": "mandatory",
        "typeof": "boolean",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if the object can illuminate self"
    },
    "#x_initialpos": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_initalpos",
        "%comment%": "x-axis initial position of the object in the scene terrian"
    },
    "#y_initialpos": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_initalpos",
        "%comment%": "y-axis initial position of the object in the scene terrian"
    },
    "#z_initialpos": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_initalpos",
        "%comment%": "z-axis initial position of the object in the scene terrian"
    },
    "#x_initialrotation": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_initalrotation",
        "%comment%": "x-axis initial rotation stance of the object in the scene
terrian"
    },
    "#y_initialrotation": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_initalrotation",
        "%comment%": "y-axis initial rotation stance of the object in the scene
terrian"
    },
    "#z_initialrotation": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_initalrotation",
        "%comment%": "z-axis initial rotation stance of the object in the scene
terrian"
    },
    "#x_objectscale": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_objectscale",
        "%comment%": "x-axis tranformable object scale of the object in the scene
    terrian"
    },
    "#y_objectscale": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_objectscale",
        "%comment%": "y-axis tranformable object scale of the object in the scene
    terrian"
    },
    "#z_objectscale": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Transform_objectscale",
        "%comment%": "z-axis tranformable object scale of the object in the scene
    terrian"
    },
    "#distfactorx": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "repeattransform",
        "%comment%": "Distance Factor in x-axis of the object"
    },
    "#distfactory": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "repeattransform",
        "%comment%": "Distance Factor in y-axis of the object"
    },
    "#distfactorz": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "repeattransform",
        "%comment%":"Distance Factor in z-axis of the object"
    },

    "XRGrabInteractable": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object is grabbable and has
    interaction"
    },
    "XRInteractionMaskLayer": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object interaction has masked layer"
    },
    "TrackPosition": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object can have a tracker position"
    },
    "TrackRotation": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object can have a tracker rotation"
    },
    "Throw_Detach": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object can be thrown and detachable"
    },
    "forcegravity": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object can have a force gravity
    applied"
    },
    "velocity": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object can hold natural velocity"
    },
    "angularvelocity": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "proot": "articles",
        "root": "Interaction",
        "%comment%": "Flag to determine if the object can hold an angular velocity"
    },
    "Smoothing": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if the object can hold smoothing"
    },
    "Smoothing_duration": {
        "req": "mandatory",
        "typeof": "string",
        "repeat": "allow",
        "root": "articles",
        "%comment%": "Flag to determine if the object smoothing duration"
    },
    "#rotate_x": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "attachtransform",
        "repeat": "allow",
        "%comment%": "Flag to determine x-axis of object rotation position"
    },
    "#rotate_y": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "attachtransform",
        "repeat": "allow",
        "%comment%": "Flag to determine y-axis of object rotation position"
    },
    "#rotate_z": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "attachtransform",
        "repeat": "allow",
        "%comment%": "Flag to determine z-axis of object rotation position"
    },
    "#pos_x": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "attachtransform",
        "repeat": "allow",
        "%comment%": "Flag to determine x-axis of object transform position"
    },
    "#pos_y": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "attachtransform",
        "repeat": "allow",
        "%comment%": "Flag to determine y-axis of object transform position"
    },
    "#pos_z": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "attachtransform",
        "repeat": "allow",
        "%comment%": "Flag to determine y-axis of object transform position"
    },
    "value": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Level of rigid with the scale of 1 to 10"
    },
    "mass": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Flag to mention the mass of the object"
    },
    "dragfriction": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Flag to determine if the object can hold drag friction"
    },
    "angulardrag": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Flag to determine if the object can be angular drag"
    },
    "Isgravityenable": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Flag to determine if the object abides by natural gravity"
    },
    "IsKinematic": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Flag to determine if the object is mobile or kinematic"
    },
    "CanInterpolate": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Flag to determie if the object can interpolate"
    },
    "CollisionPolling": {
        "req": "mandatory",
        "typeof": "string",
        "proot": "articles",
        "root": "XRRigidObject",
        "repeat": "allow",
        "%comment%": "Flag to determine if the object can CollisionPolling"
    },
    "aud_hasaudio": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Flag to determine if the object has audio setup"
    },
    "aud_type": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Type of Audio file"
    },
    "aud_src": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Audio Source file and its path"
    },
    "aud_volume": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Measure of audio volume"
    },
    "aud_PlayInloop": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Flag to determine if the audio is playable in loop"
    },
    "aud_IsSurround": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Flag to determine if the audio of the object is surroundable"
    },
    "aud_Dopplerlevel": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Audio doppler level of the object if available"
    },
    "aud_spread": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Audio spread measure of the object"
    },
    "aud_mindist": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Minimum distance covered via audio"
    },
    "aud_maxdist": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Maximum distance covered via audio"
    },
    "_Opttxt1": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "optional notes placeholder to add additional notes about the
   object"
    },
    "@context_img_source": {
        "req": "mandatory",
        "typeof": "string",
        "root": "articles",
        "repeat": "allow",
        "%comment%": "Sample image source file location for designer reference"
    },

  "state":{
  "req":"mandatory",
  "root":"ObjAction",
  "typeof":"object",
  "repeat":"allow",
  "%comment%":"This is a master node for state array object to defined states of a
    given action and its response over objects"
  },

  "stateid":{
  "req":"mandatory",
  "root":"ObjAction",
  "typeof":"number",
  "repeat":"allow",
  "%comment%":"Define your own unique identifier to identify a state"
    },


  "statename":{
  "req":"mandatory",
  "proot":"state",
  "root":"ObjAction",
  "typeof":"string",
  "repeat":"allow",
  "%comment%":"Provide a state name for a given stateid"

  },

  "stateinitial":{
  "req":"mandatory",
  "proot":"state",
  "root":"ObjAction",
  "typeof":"string",
  "repeat":"allow",
  "%comment%":"Describe your Initial Statement in less than 10 words"
    },

  "statetransition":{

  "req":"mandatory",
  "proot":"state",
  "root":"ObjAction",
  "typeof":"object",
  "repeat":"allow",
  "%comment%":"to describe the underlying events and target with a preference value
    "
  },

  "sevent":{
  "req":"mandatory",
  "proot":"statetransition",
  "root":"ObjAction",
  "typeof":"string",
  "repeat":"allow",
  "%comment%":"to describe the underlying events and its target along with a
    preference value"
  },

  "starget":{
  "req":"mandatory",
  "proot":"statetransition",
  "root":"ObjAction",
  "typeof":"string",
  "repeat":"allow",
  "%comment%":"to describe the underlying target for a given event along with a
    preference value"
  },

  "spref":{
  "req":"mandatory",
  "proot":"statetransition",
  "root":"ObjAction",
  "typeof":"number",
  "repeat":"allow",
  "%comment%":"to describe the perference value for a given sevent and starget. It
    starts with value 0"
  },

  "statefinal":{

  "req":"mandatory",
  "root":"state",
  "proot":"ObjAction",
  "typeof":"string",
  "repeat":"allow",
  "%comment%":"Describe your final Statement in less than 10 words"
  }


}
```

Listing 6.4 Article Validator

Below is the Fillable Action Response Model template File.

```
{
  "ObjAction": [
  {
  "actresid":"",
  "sourceObj":"",
    "targetObj":"",
  "IsCollision":"",
    "response":"",
    "comment":"",
    "Syncronous":"",
  "repeatactionfor":""

  }
 ]
}
```

Listing 6.5 Action Response Model Template

Below is the Validator File for the fillable Action Response Model template File.

```
{

"ObjAction":{
    "req":"mandatory",
    "typeof":"object",
     "repeat":"notallow",
     "%comment%":"Collection of Objects that are set for action and corresponding
    response"
    },

"actresid":{
    "req":"mandatory",
     "root":"ObjAction",
    "typeof":"string",
     "repeat":"allow",
     "%comment%":"Unique Identifier for a given action and response"
    },

"sourceObj":{
    "req":"mandatory",
  "root":"ObjAction",
    "typeof":"string",
     "repeat":"allow",
     "%comment%":"Source object where the action is applied or originated"
    },

"targetObj":{
    "req":"mandatory",
  "root":"ObjAction",
    "typeof":"string",
     "repeat":"allow",
     "%comment%":"Target object where the action’s response is made visible"
    },


"IsCollision":{
    "req":"mandatory",
  "root":"ObjAction",
    "typeof":"boolean",
     "repeat":"allow",
     "%comment%":"Flag to determine if the action is based on collision"
    },

"response":{
    "req":"mandatory",
  "root":"ObjAction",
    "typeof":"string",
     "repeat":"allow",
     "%comment%":"Defines the type of response, i,e, natural law based response or
   a customized response"
    },

  "comment":{
    "req":"mandatory",
    "root":"ObjAction",
    "typeof":"string",
     "repeat":"allow",
     "%comment%":"Illustrate in words the course of response and its journey"
    },

"Syncronous":{
    "req":"mandatory",
  "root":"ObjAction",
    "typeof":"boolean",
     "repeat":"allow",
     "%comment%":"Defines if the response is Syncronous to action or has any delay
    or Asyncronous to other objects"
    },

  "repeatactionfor":{
    "req":"mandatory",
    "root":"ObjAction",
    "typeof":"string",
     "repeat":"allow",
     "%comment%":"Flag to use or repeat the same action for other objects. Metion
    the targetobjects ids or names"
    }




}
```

Listing 6.6 Action Response Validator

Below is the Fillable Timeline Model template File.

```
{

     "animate_trigSync":{
                       "tsyncid":"",
            "tsOntrigger":"true",
            "SyncObjList":[ ],
            "tSyncNote":""
        },
      "animate_nontrigSync":{
            "ntsyncid":"",
                       "ntsOntrigger":"false", "ntSyncObjList":[ ], "ntSyncNote
    ":""
```

```
        },
      "animate_trigAsync":{
            "tasyncid":"",
                        "taOntrigger":"true", "tAsyncObjList":[ ], "tAsyncNote":""
        },
      "animate_nontrigAsync":{
            "ntasyncid":"",
                        "ntaOntrigger":"false", "ntAsyncObjList":[ ], "ntAsyncNote
    ":""
        },


     "routine": [     {  "routeid":"",    "starttime":"",   "endtime":"",  "order
    ":[] }]



}
```

Listing 6.7 Timeline Model Template

Below is the Validator File for the fillable Timeline Model template File.

```
{
   "tsyncid":{
     "root": "animate_trigSync",
      "req":"mandatory",
      "typeof":"number",
      "repeat":"notallow",
      "%comment%":"Unique Identifier for animating scene synchronously based on a
    trigger"
   },
  "tsOntrigger":{
     "root": "animate_trigSync",
      "req":"optional",
      "typeof":"boolean",
     "repeat":"notallow",
      "%comment%":"True or false flag if this is a triggered animation based an
    action"
   },
   "SyncObjList":{
     "root": "animate_trigSync",
      "req":"optional",
      "typeof":"object",
      "repeat":"notallow",
      "%comment%":"List of objects are animated in a timeline that are triggered
    Synchronously"
   },
  "tSyncNote":{
     "root": "animate_trigSync",
      "req":"optional",
      "typeof":"string",
     "repeat":"notallow",
      "%comment%":"place holder for additional notes for Sync object"
   },

    "ntsyncid":{
     "root": "animate_nontrigSync",
      "req":"mandatory",
      "typeof":"number",
       "repeat":"notallow",
      "%comment%":"Unique Identifier for animating scene synchronously based on a
    non trigger"
   },
  "ntsOntrigger":{
     "root": "animate_nontrigSync",
      "req":"optional",
      "typeof":"boolean",
     "repeat":"notallow",
      "%comment%":"True or false flag if this is a non triggered animation based an
     action"
   },
   "ntSyncObjList":{
     "root": "animate_nontrigSync",
      "req":"optional",
      "typeof":"object",
      "repeat":"notallow",
      "%comment%":"List of objects are animated in a timeline that are non
    triggered ASynchronously"
   },
  "ntSyncNote":{
     "root": "animate_nontrigSync",
      "req":"optional",
      "typeof":"string",
     "repeat":"notallow",
      "%comment%":"place holder for additional notes for Sync object"
   },



     "tasyncid":{
     "root": "animate_trigAsync",
      "req":"mandatory",
      "typeof":"number",
        "repeat":"notallow",
      "%comment%":"Unique Identifier for animating scene asynchronously based on a
   trigger"
   },
  "taOntrigger":{
     "root": "animate_trigAsync",
      "req":"optional",
      "typeof":"boolean",
     "repeat":"notallow",
      "%comment%":"True or false flag if this is a triggered animation based an
   action"
   },
   "tAsyncObjList":{
     "root": "animate_trigAsync",
      "req":"optional",
      "typeof":"object",
      "repeat":"notallow",
      "%comment%":"List of objects are animated in a timeline that are triggered
   ASynchronously"
   },
  "tAsyncNote":{
    "root": "animate_trigAsync",
     "req":"optional",
     "typeof":"string",
    "repeat":"notallow",
     "%comment%":"place holder for additional notes for ASync object"
  },

   "ntasyncid":{
    "root": "animate_nontrigAsync",
     "req":"mandatory",
     "typeof":"number",
      "repeat":"notallow",
     "%comment%":"Unique Identifier for animating scene asynchronously based on a
   non trigger"
  },
  "ntaOntrigger":{
    "root": "animate_nontrigAsync",
     "req":"optional",
     "typeof":"boolean",
    "repeat":"notallow",
     "%comment%":"True or false flag if this is a non triggered animation based an
    action"
  },
   "ntAsyncObjList":{
    "root": "animate_nontrigAsync",
     "req":"optional",
     "typeof":"object",
     "repeat":"notallow",
     "%comment%":"List of objects are animated in a timeline that are non
   triggered ASynchronously"
  },
  "ntAsyncNote":{
    "root": "animate_nontrigAsync",
     "req":"optional",
     "typeof":"string",
    "repeat":"notallow",
     "%comment%":"place holder for additional notes for ASync object"
  },

  "routine":{
     "repeat":"notallow",
     "req":"mandatory",
    "typeof":"object",
      "%comment%":"To mention a repeated routine of timeline actions"
  },


 "routeid":{
   "root": "routine",
   "req":"mandatory",
   "typeof":"number",
    "repeat":"allow",
    "%comment%":"Unique identifier of a routine"

 },
  "starttime":{
   "root": "routine",
   "req":"optional",
   "typeof":"string",
     "repeat":"allow",
    "%comment%":"Start Timestamps of routine"

 },
  "endtime":{
   "root": "routine",
   "req":"optional",
   "typeof":"string",
     "repeat":"allow",
    "%comment%":"End Timestamps of routine"

 },
   "order":{
   "root": "routine",
   "req":"optional",
   "typeof":"object",
      "repeat":"allow",
    "%comment%":"Order of items in a routine, mention the routine ids in a
   sequence to understand the order of events."

  }


}
```

Listing 6.8 Timeline Validator

6.2.5 Understanding Behavior and State in VReqST

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. To author this behavior in Stage 4, we use the code snippet below using a **CASE** condition, and everything in **bold** is the pre-defined code construct from *behave.json*. The text in *italics* are articles from *article.json*, and the rest is in free-form text, which is not validated. The code snippet construct in bold is validated using behave.json.

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 requirement analysts can specify behaviors using in-built code constructs. These code constructs do not follow existing programming standards but are custom-made for VReqST. They follow an in-built syntax that is simplified and easy for a non-developer. After considering the feedback from VR community and as requirement analysts are are full-scale developers, we made them low-code.

Similarly, **"State"** transition can be specified by defining state parameters like initial state and final based on transition parameters like event and target. For example, to describe the state transition of sun-rise in VReqST, the requirement is to specify that while the sun rises, the sun rays should cast a shadow and transition snow to water in a specified time counter.

```
{
 "state": [
    {
    "id": "sunrise2013",
    "name": "Sun Rise in New York",
    "initial": "dark horizon",
    "transition": [
    {
      "object": ["Sun","trees"],
      "event": "rising",
      "target": "sun ray cast shadow on all assests from east to west"
     },
    {
      "object": ["Sun","Snow"],
      "event": "rising",
      "target": "turn the snow into water gradually after 35 seconds"
    }
    ],
    "final": "Display sunrise with cast shadow on trees and snow converted to water
     after 50 seconds"
     }
   ]
}
```

Listing 6.9 Sample State Template

6.2.6 VReqST Features

Following are a few significant features of the VReqST tool that are provisioned for requirement analysts to embrace based on the sophistication of their VR software product:

- **Bare minimum model template and attributes:** VReqST provides bare minimum model attributes related to specific model concepts by default. These default model attributes are illustrated in Table 6.2. The underlying validator file for the respective model template file constitutes validation rules associated with these bare-minimum attributes. The requirement analysts can use these default model template attributes to author simple VR software products.

- **Implement new model template and attributes:** The requirement analysts can include new model attributes without altering the bare minimum model attributes for a given model concept. For every newly implemented model attribute in the model template file, a validation logic should exist in its validator file. For example, if a new attribute called **”color”** (inspired by OpenXR standard specification [97] - **XrColor4f** structure) is included *scene.json*, 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 *svalid.json*. Thus, supporting new model attributes extends the scope of capturing requirement specifications for VR. The steps to update these files are available as part of our documentation [135].

- **Managing Specifications:** The step-by-step process of managing projects in VReqST and detailed screen-flow are available in supporting documentation [135]. Requirement Analysts can rely on this document to manage their VR requirement specification projects. Requirement analyst can author requirement specifications for multiple VR products through a project management page. Each project follows the specification workflow mentioned as part of Fig 6.4 and Fig 6.5. Requirement analysts can write, save, edit, re-edit, and delete the authored requirement specifications of their respective VR products.

6.3 Example Specifications

We used VReqST to specify requirements for two VR applications to understand the capabilities and shortcomings. We specified detailed requirements for a VR bowling alley game and a VR Virtual art gallery that uses a limitless locomotion algorithm called PragPal [178]. We developed these VR Scenes using the UNITY Game Engine Version - 2022.2.16 and works on Oculus Meta Quest 2 head-mounted device.

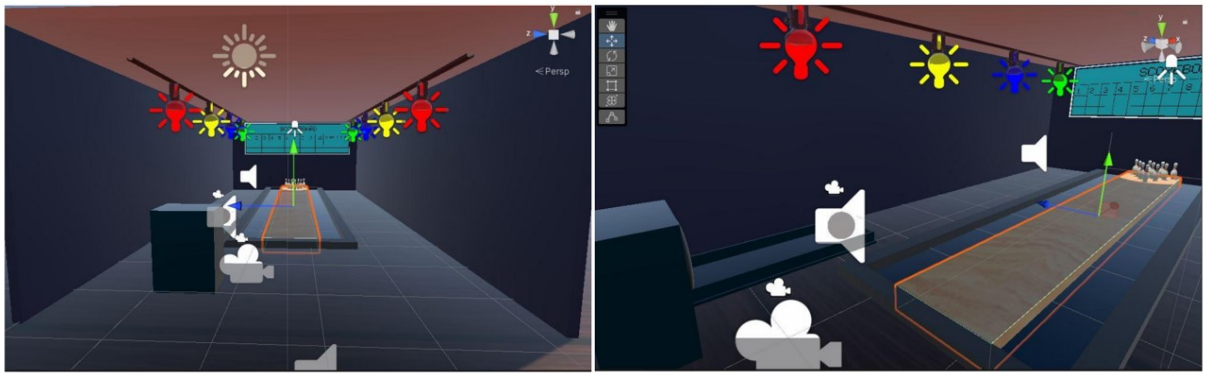


Figure 6.6 VR Bowling Alley Game in a developer mode (UNITY Engine) - Point-of-View 1 & 2

VR Bowling Alley Game: A VR bowling alley game is a virtual facility where bowling is played. It is a multiplayer game that follows all the bowling alley rules. The Game scene is played under a 30 ft (length) x 30 ft (breadth) x 30 ft (height) virtual play area. The game contains a bowling ball, ten pins, a pinsetter for setting the pins in a frame, an alley lane, and a gutter that acts as a boundary for the alley lane. The game is controlled by a Control Desk that registers the parties, registers bowlers, and manages the party assignment to the game lanes. There are ten games for each party player. The scores are displayed on the scoreboard. The player from a party who scores the highest in all ten games wins. We authored and shared the VR bowling alley game specifications with the VR developer to design and develop the game. The final working game [134] and the sample specifications are available as part of our resources [136].

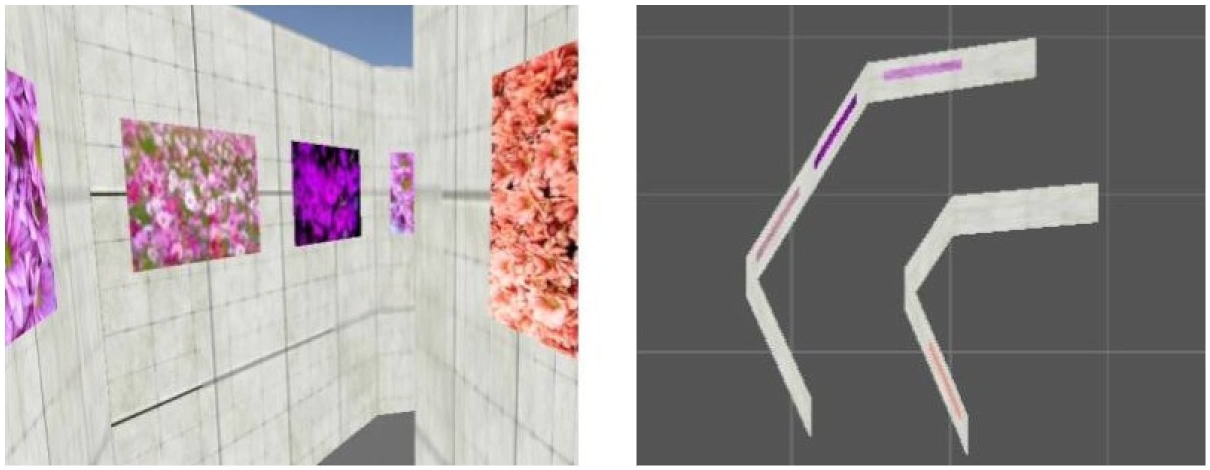


Figure 6.7 VR Art Gallery using limitless walking (UNITY Engine) - Point-of-View 1 & 2

VR Virtual Art Gallery: The VR virtual art gallery is an endless corridor comprising two walls running parallel to each other. The end users walk through the gallery to explore the art exhibits on these walls and progress in the forward direction. Under the influence of an underlying locomotion algorithm, the users' path is virtually limitless in a limited physical space, i.e., within a fixed physical play area, the user is set to locomote limitless in a virtual play area. We authored the requirement specifications of this virtual art gallery and the underlying limitless locomotion algorithm using the VReqST to study the capabilities of specifying algorithms using the behavior step. The final working game [178] and the sample specifications are available as part of our resources [136]. This validation helped us conduct a Unit Test of VReqST and demonstrate its capabilities of specifying custom features and algorithms for a given VR scene. Considering these early observations, we planned to conduct a detailed study in practice to examine the completeness and correctness of requirements specified through VReqST as a future work. In the following chapter, we present validation of VReqST tool in practice.

Specifications by Novice Users: At IIIT Hyderabad, we offered a CS300 level Software Engineering (SE) course during Spring 2023 focused on building software systems using SE practices. The course had 130+ Undergraduate, Graduate, and Industry student learners. We randomly created 28 teams with a team size of 4 to 5 students for all the course related tasks and activities. Requirement engineering concepts were announced as part of the course to the students before announcing the assignment. We conducted a team-based assignment as an experiment [141] in a classroom setup to use the VReqST [135] tool to elicit and specify requirements using a VR application as an example domain.

As a prerequisite, we let students use some VR applications at our University VR lab and recommended watching a few VR movies. The experiment consists of two phases, as described below. The duration of each phase is 2 weeks. As part of phase one, each team must internally elicit one VR application and write detailed requirements using a Software Requirement Specification (SRS) template [76] supplied to them. As part of phase two, the SRS documents from the previous phase are exchanged randomly among teams. Each team is now required to use the VReqST tool [135] to elicit and specify the requirements of VR applications from the allocated SRS document, i.e., each team receives one random SRS document except the one they authored. We created a well-defined evaluation rubric [76] to grade the requirements correctness of VR application by comparing SRS submission and its related VReqST specification. We have set up a face-time between the teams who authored the SRS submission and its correlated VReqST submission to grade the requirement correctness in the presence of a VR domain expert. All these specification submissions are made available as part of our resources [136].

Conclusion: In this chapter, we illustrate the overall workflow of a requirement specification tool called VReqST to author specifications for building VR software products. We developed this tool based on role based model template of VR Software system and have presented few sample specifications for reference. We present our detailed validation study as part of following chapter.

Chapter 7

Validation of VReqST Tool in Practice

How do we validate a requirement specification method? Software requirements are not static and evolve throughout the software development lifecycle. As the project progresses, new requirements may arise due to changing user needs, technological advancements, and stakeholder feedback. As a result, software development teams must be agile and adaptable in managing and accommodating these evolving requirements. Conversely, requirement specifications are vital in ensuring that software products meet the desired functionalities, performance, and quality standards during this evolution process. Requirement specifications serve as a guiding framework for the software development team during requirement evolution. These specifications comprehend the software's intended behavior and enable the software practitioners to make informed design decisions and develop solutions that align with the specified requirements. By adhering to these specifications, software development teams can create reliable, efficient products that meet their users' expectations. Consistency, Completeness and Correctness of Requirements **(R)** Specification **(S)** are only be validated and verified under the context a given Domain **(D)** [92]. As shown in below equation 7.1, given the assumption that the machine will perform as instructed by the specification, and that our model of the domain faithfully predicts how the real world will behave [92].

$$S \cup D \in R \tag{7.1}$$

These observations holds true even for VR software systems and its VR practitioner community. However, the critical difference is the evolution of VR requirements is very difficult to track and supervise unless there is a structured approach to comprehending VR as a technology domain. VReqST aims to address this gap. VReqST is developed to specify, maintain, track, and manage the evolution of VR software product requirements more precisely. Considering this as our motivation, we reached the VR community to understand how VReqST can disrupt this evolution and aid the VR community in easing overall VR product development. In the following sub-sections, we present our attempts to validate the adoption and impact of VReqST in practice.

7.1 VR Community Adoption & Feedback

We reached the VR Community from the Industry through various channels like Linkedin, VR/AR Meetups, Online VR Summits, Discord XR Connect, Khronos Group – Deloitte Digital and UNITY Unite Expo (2022 & 2023) to promote VReqST for validation, adoption, and feedback. We clearly defined our evaluation study's objectives, ensuring they are Specific, Measurable, Achievable, Relevant, and Time-bound. This ensures we receive a clear direction for our validation and helps incrementally improve the tool for wider adoption. We reached more than 500 practitioners from the VR Community, of which only 101 participants have shown to deploy the tool in-house for validation. Of these participants, 53 have actively participated in iterations and helped us revise and improve the tool over time. These participants includes Business Analysts, VR Product Managers, VR Program Managers. VR Quality Engineers and VR Developers. This multi-year empirical study was conducted between October 2022 and January 2024. Following are the steps followed by the practitioners who have shown interest in implementing and validating VReqST in practice.

- Firstly, we shared VReqST official documentation and steps to install the tool so that they can groom and identify the intent along the tool's usage into their regular processes [135].
- We have put in all our efforts to deploy VReqST to integrate seamlessly with their existing processes and scale effectively with their existing requirement specification practices.
- We helped the participants understand the practicality and viability of implementing the VReqST tool. Assess the potential benefits against the costs, resources, and time required for integration with their conventional approaches.
- We requested to involve diverse teams, domain experts, and stakeholders from their internal organization to gather varied perspectives and insights. We leveraged collective expertise to comprehend the tool's potential impact across various aspects of their VR product development. These diverse viewpoints will help us identify the tool's opportunities, threats, and unforeseen implications in practice.
- We received enhancements to VReqST and incorporated them in a new version of the tool. We received enhancements in three iterations and we included the revisions and released a new version in all the three iterations for further adoption.
- In the end, we conducted a short survey to understand the impact of VReqST in their day-to-day business when compared to their conventional practices.

In the following subsection, we present our detailed study setup, feedback received in three iterations along with the impact survey details.

7.1.1 VReqST Validation Study Setup

The following steps are involved in our validation study setup with VR community practitioners in respective iterations.

Iteration 1

- In early October 2022, we provided the source code of our first version of VReqST tool[1] to deploy.
- We requested to author at least 2-3 new requirement specifications of their existing projects using VReqST and share them with their VR developers.
- As part of the early analysis, we requested the requirement analysts to review the capabilities of the VReqST tool compared with their conventional requirement specification processes.
- The total of 53 unique VR practitioners have deployed the tool during this iteration.

Iteration 2

- We re-connected with the VR practitioners and sought their feedback in mid April 2023. We implemented the feedback from iteration 1 and shared the revised VReqST tool[2] for redeployment in May 2023.
- We requested the practitioners to re-author the old specifications using the revised tool and also the new requirements to experience the revisions and ease-of-use with new enhancements.
- During this period, we collected feedback from these VR practitioners on iteration 2 version of the VReqST tool and documented for further improvement.
- The total of 48 out of 53 unique VR practitioners have provided feedback on the iteration 2 version of the VReqST tool.
- However, all the 53 unique VR practitioners have deployed the iteration 2 edition of VReqST tool.

Iteration 3

- The received significantly valuable feedback during our iteration 2 and enhanced the tool during July 2023 to October 2023. We completed our iteration 3 version of VReqST tool[3] by end of October 2023 and delivered it to the engaged VR practitioners during early November 2023.
- All the 53 unique VR practitioners have deployed the iteration 3 edition of VReqST tool. This was the final iteration of our VReqST enhancements that significantly improved the workflow and ease-of-use of the VReqST in practice.

[1]https://github.com/Shambhaviiiit/VReqST
[2]https://github.com/Amogha027/VReqST
[3]https://github.com/Amogha027/VReqST-2

In the following sub-section, we present detailed observations shared by VR practitioners during each iterations along with the implementation information related to those enhancements in detail.

7.1.2 VReqST Impact and Experience Survey

After a thorough review of iterations and implementing feedback on VReqST, we reached out to the VR practitioners to share their experiences on using VReqST and the impact it has made on their day-to-day activities. The following are the questions posed to VR practitioners to help us understand their experiences and impact of VReqST on their VR product development journey in contrast to their regular processes.

1. Is the VReqST tool compatible with the existing tools that you use for your overall VR software product development?
2. Is the overall user-interface of VReqST tool's user interface and design is intuitive enough and user-friendly for your team members?
3. Does the VReqST tool allow you to specify requirements for necessary features in your VR software to meet your business needs?
4. With whom and how frequently did you use VReqST tool while specifying or explaining the requirements to your target users (developers or stakeholders)?
5. Are you able completely specify and track the requirements associated with your feature in a given VR scene?
6. Does the tool allow you to correctly specify the requirement as per the conventional standards of the VR technology domain?
7. Is the VReqST tool's performance and stability under different conditions and workloads acceptable for your day-to-day business?
8. Do you see the VReqST tool can be extendable with your current product needs and adapt to future needs considering updates to VR technology domain?

Overall, our validation study and Impact & Experience Survey of the VReqST tool focuses on the following ***quality attributes*** listed below in particular stack order. We evaluated the benefits of VReqST using these attributes. We share detailed observations from the VR practitioners using these quality attributes for better quantification.

- **Specify**: Ability to specify the requirements using the existing model templates.
- **Customization**: Ability to customize and alter the model template and underlying validator to meet custom business requirements.

- **Scalability**: Ability to scale VReqST to include new model attributes for applications that are domain specific and context specific.
- **Traceability**: Ability to trace the requirements across various stages of specifications.
- **Usability**: Ability to use the overall tool and manage track changes to the requirements.

7.1.3 Observations - Validation Study

While releasing the incremental version of VReqST during three iterations, we gathered feedback through informal interviews with the VR practitioners on VReqST. We summarize the overall observations gathered during all the iterations using ***'Abductive Reasoning'*** approach and illustrated below. We categorized these observations into two parts - **Merits** and **Enhancements**

Merits

- **Meets the Intention:** The VReqST tool can meet its required intention, i.e., specifying VR requirements, customizing the VR model template and its validator, and scaling the VR model template to new model attributes.
- **Supports Traceability**: The VReqST tool supports traceability. We can trace defined articles and their action responses and state information across the trail of overall specifications. This helps a VR developer and quality engineer to trace, define, and test the flow of events while developing the VR scene.
- **Improves Clarity:** The VReqST tool helps improve requirement gaps and present clarity as the specifications are precise. It also reduces the time required to comprehend the overall implication of a specification as the model template provides details from Scene to timeline.
- **Enhances productivity:** The VReqST tool avoids back-and-forth communication between developers and product managers on requirement clarification, eventually helping accomplish developer tasks more effectively and increasing output by saving time and effort.
- **Increases adaptability:** By managing requirements of various projects through projects view, helps requirement analysts to adapt to new workflows and achieve better results.
- **Enhances collaboration:** As the specifications can be shared across various projects, multiple stakeholders can collaborate on projects simultaneously, fostering teamwork and communication.
- **UI/UX:** The practitioners has reported 20+ recommendations to the overall UI/UX design. We included the updates in an iterative release. As the VReqST tool is still a prototype, we can extend industry-scale UI/UX features in the user interface that can impact user experience. End-to-end usability testing on VReqST helped us improve the usability aspects of the tool.

- **Resuability:** The specifications authored in VReqST are either localized to a specific project or can be extended to a given new scene. The tool has ability to save, share and re-use the authored specifications, especially for articles and their action responses as a repository. Such a repository bank can help other requirement analysts to adopt and reuse based on their business needs.

Enhancements

- **Simplify Authoring:** Currently, the requirement analysts (RA) are required to specify requirements using the JavaScript Object Notation (JSON) model template files. Instead, if a form-based model template to JSON conversion is handled at all stages of VReqST, we may see higher adoption of VReqST from VR practitioners with non-programming backgrounds.
- **Security and Privacy Aspects:** Security and Privacy requirements are a few key areas of overall VR Software. The VReqST tool in its current form does not provide any model attributes linked with security and privacy in line with bare minimum model attributes. This may lead to setbacks in the practice of adopting VReqST.
- **Complexity:** Authoring complex behavior requires much conceptual effort compared to conventional requirement specifications that are authored in plain English. The behavior editor needs to be either gamified or allowed to be in a low-code mode to improve the overall adoption of the behavior editor.

In contrast to the consolidated observations, we present detailed feedback that we gathered during each iteration below, along with the revisions performed on the VReqST tool. We shared the revised tool during each iteration and have incorporated these changes accordingly.

Iteration 1

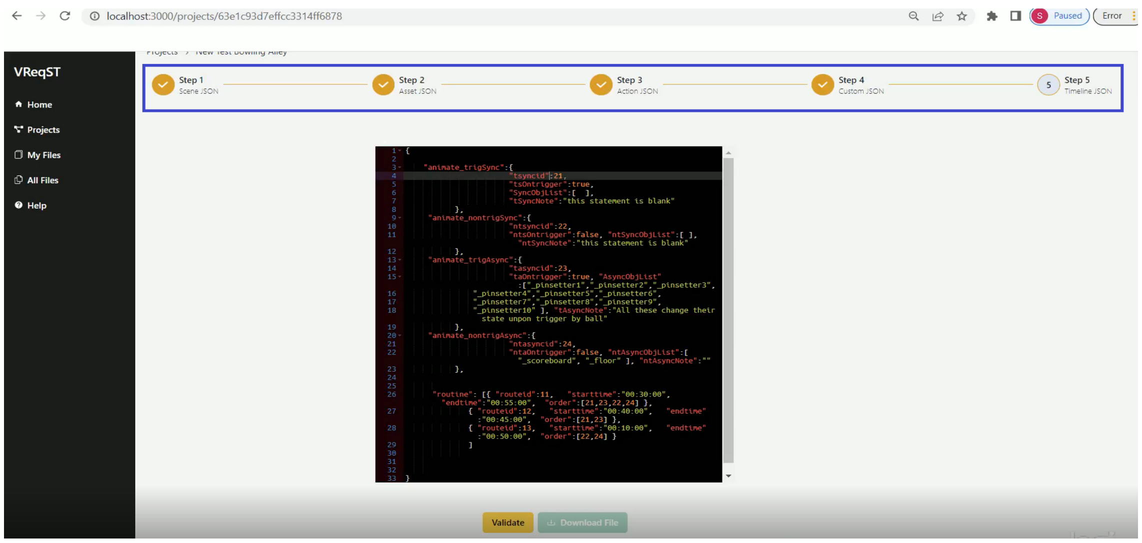
- **Multistage Authoring** - Our initial VReqST tool didn't support a multi-stage authoring of our requirements specifications for available model template files like Scene, Article, action responses, Custom behaviors, and Timeline in a particular order. As shown in Fig 7.1, we revised the user interface of the Authoring wizard to include the model template validation in a particular order. The highlighted pane using a blue outline clearly illustrates the revision. Each stage requires successful validation to enter into a new specification stage, i.e., unless scene model template validation is unsuccessful, one cannot navigate to the article model template. Upon completing all stages, the requirement analysts can navigate all the stages to go through the individual specifications in detail. This revision helps requirement analysts review and specify requirements in stages and trace them to the initial specification.
- **Project Management** - In our initial VReqST edition, the requirement specification was static and linear, i.e., a requirement analyst must specify and download the finalized specifications. However, in the revised version, we included user-management and project-management modules

to manage the specifications group for a particular project. This will help requirement analysts save, maintain, and revise the specifications for individual projects. As part of this revision, we can now create a new project, share it with others and edit/revise in future based on the changes to our business need. Fig 7.2 illustrates these additions.

- **In-page Validation Messages** - We revised our ad-hoc message regarding validation status on demand on the same page rather than performing validation at the end of all stages. With the inclusion of multi-stage authoring, we implemented successful and failure messaging as in-page status messages, as shown in Fig 7.3.

- We have converted the underlying MongoDB instance from a private cloud cluster to a public cloud cluster so that the VR practitioners can download the underlying DB with the core installation instance and re-point it to their private cloud cluster instances. This will help the VR community practitioners to store their requirement specifications as private.

- Other minor revisions include font changes and text re-sizing across different pages, including additional fields on the registration page, and a few minor implementation changes to make it simpler to deploy.



```
{

    "animate_trigSync":{
                    "tsyncid":21,
                    "tsOntrigger":true,
                    "SyncObjList":[  ],
                    "tSyncNote":"this statement is blank"
            },
     "animate_nontrigSync":{
                    "ntsyncid":22,
                    "ntsOntrigger":false, "ntSyncObjList":[ ],
                      "ntSyncNote":"this statement is blank"
            },
     "animate_trigAsync":{
                    "tasyncid":23,
                    "taOntrigger":true, "AsyncObjList"
                      :["_pinsetter1","_pinsetter2","_pinsetter3",
             "_pinsetter4","_pinsetter5","_pinsetter6",
             "_pinsetter7","_pinsetter8","_pinsetter9",
             "_pinsetter10" ], "tAsyncNote":"All these change their
               state unpon trigger by ball"
            },
     "animate_nontrigAsync":{
                    "ntasyncid":24,
                    "ntaOntrigger":false, "ntAsyncObjList":[
                      "_scoreboard", "_floor" ], "ntAsyncNote":""
            },


      "routine": [{ "routeid":11,   "starttime":"00:30:00",
        "endtime":"00:55:00",  "order":[21,23,22,24] },
             { "routeid":12,   "starttime":"00:40:00",   "endtime"
               :"00:45:00",  "order":[21,23] },
             { "routeid":13,   "starttime":"00:10:00",   "endtime"
               :"00:50:00",  "order":[22,24] }
             ]



}
```

Figure 7.1 Multi-stage view of Requirement Authoring

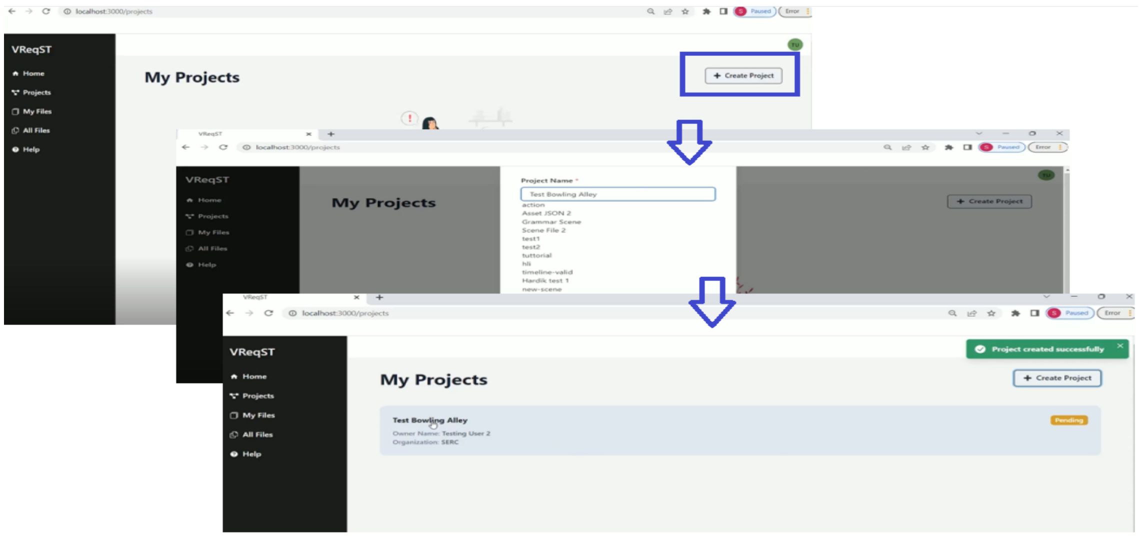


Figure 7.2 Managing Requirement Specification as Projects

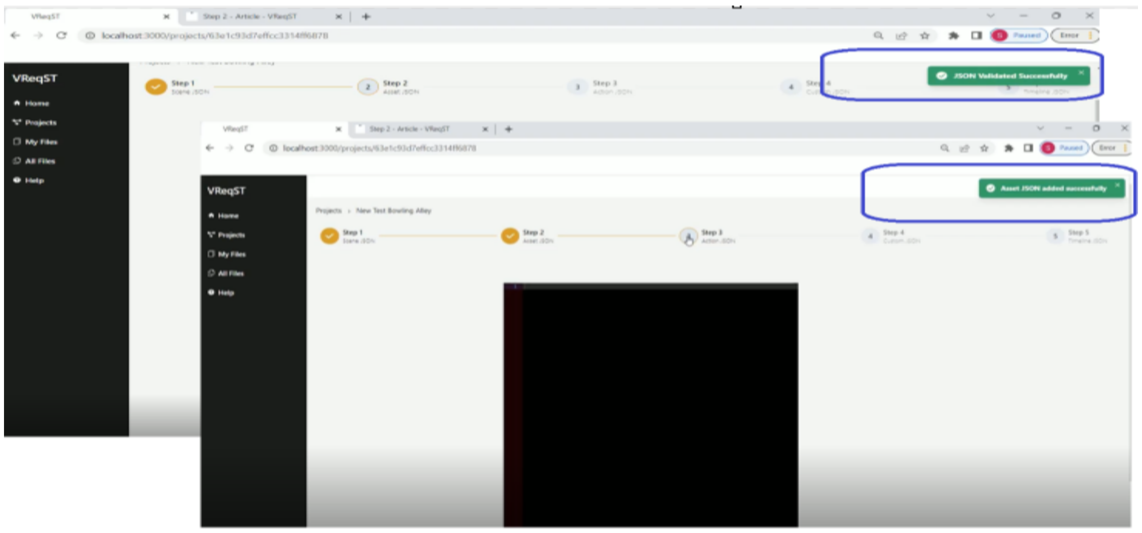
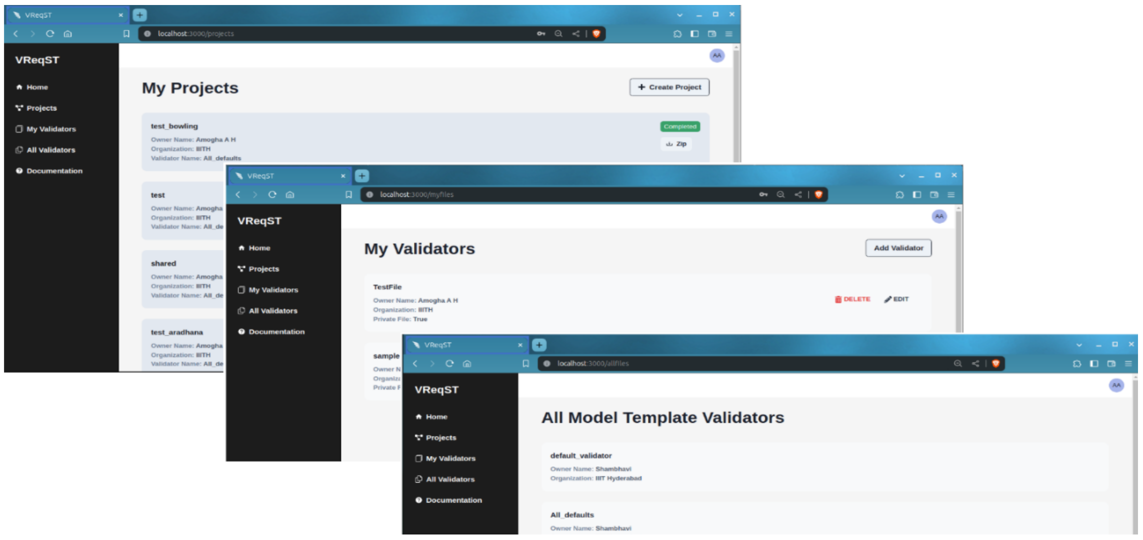


Figure 7.3 In-page validation message for a given model template

Iteration 2

- **Customizing Validator Files** - Until the iteration 1 version of VReqST, the requirement analyst had to rely on the core model template validator for each validation stage, i.e., scene, article, action-response, custom, and timeline. The model template validator must be updated directly from the backend if they wish to customize. However, as part of this revision, we provisioned a customizing model template validator through UI. We also provide the ability to choose the custom validator while authoring the specification during project creation. Thus, as part of this



My Projects

test_bowling
Owner Name: Amogha A H
Organization: IIITH
Validator Name: All_defaults

test
Owner Name: Amogha
Organization: IIITH
Validator Name: All_de

shared
Owner Name: Amogha
Organization: IIITH
Validator Name: All_de

test_aradhana
Owner Name: Amogha
Organization: IIITH
Validator Name: All_de

My Validators

TestFile
Owner Name: Amogha A H
Organization: IIITH
Private File: True

sample
Owner N
Organiz
Private F

All Model Template Validators

default_validator
Owner Name: Shambhavi
Organization: IIIT Hyderabad

All_defaults
Owner Name: Shambhavi

Figure 7.4 Customizing Validator Files through UI

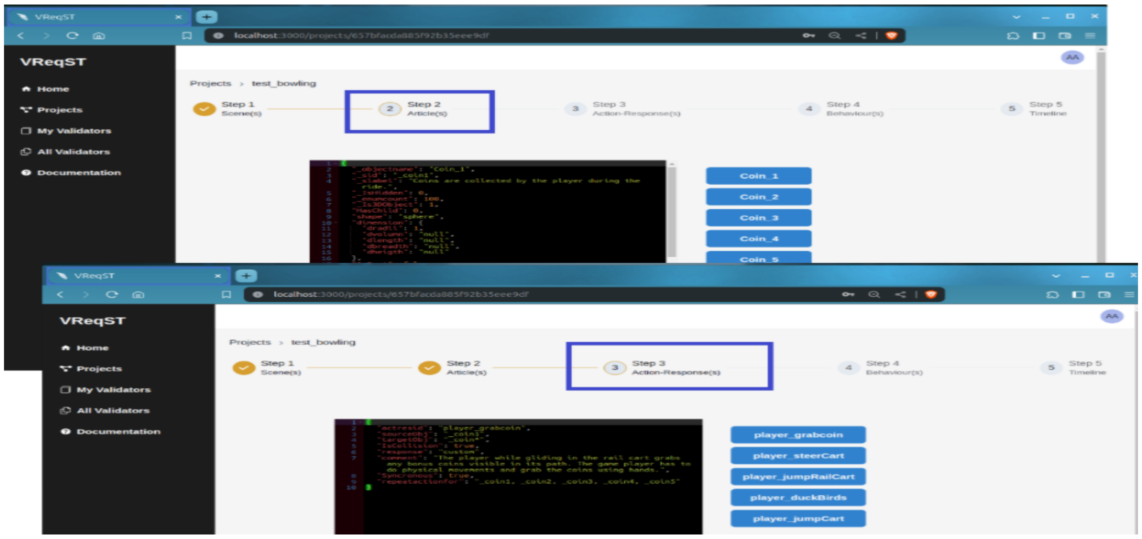
iteration, the requirement analyst can upload a custom validator for each VR requirement specification stage and choose this custom validator while creating a requirements project. Fig 7.4 illustrates the new view of My projects, My validators, i.e., a list of custom validators created by oneself, and all model template validators created by the rest of the other requirement analysts who use the same tool.

- **Authoring States of Article** - Until the previous iteration, the state for a given article is not included as part of the default article model template. As part of this iteration, the underlying article model template now supports the ”state” of an article where the requirement analyst will now have the ability to define the initial state, final state, and state transition of a given article based on the event performed w.r.t a given timestamp.

- Other minor revisions include additional text on welcome page, Include documentation support as Help, ability to review past and ongoing project details are included as part of this iteration.

Iteration 3

- **Article and Action/Response Template revisions** - Until iteration 2, the requirement analysts must specify all the articles in a single article model template. As part of this iteration, we revised the article model template view stage by providing a new button to specify one article after another. As shown in Fig 7.5, the highlighted article stage in blue contains articles like *coin_1*, *coin_2*, etc, listed beside the model template editor. Upon successfully specifying each article in Step 2, the requirement analyst can click on the respective article to revise the specification before moving to the next stage, i.e., the action-response stage. As shown in Fig 7.5, Step 3, highlighted in blue, i.e., action-response, also has a similar update. Instead of authoring all action responses between two articles in a single action-response model template file, we can now author



```
{
  "_objectname": "Coin_1",
  "_sid": "_coin1",
  "_slabel": "Coins are collected by the player during the
    ride.",
  "_IsHidden": 0,
  "_enumcount": 100,
  "_Is3DObject": 1,
  "HasChild": 0,
  "shape": "sphere",
  "dimension": {
    "dradii": 1,
    "dvolumn": "null",
    "dlength": "null",
    "dbreadth": "null",
    "dheigth": "null"
  },
```

```
{
  "actresid": "player_grabcoin",
  "sourceObj": "_coin1",
  "targetObj": "_coin*",
  "IsCollision": true,
  "response": "custom",
  "comment": "The player while gliding in the rail cart grabs
    any bonus coins visible in its path. The game player has to
    do physical movements and grab the coins using hands.",
  "Syncronous": true,
  "repeatactionfor": "_coin1, _coin2, _coin3, _coin4, _coin5"
}
```

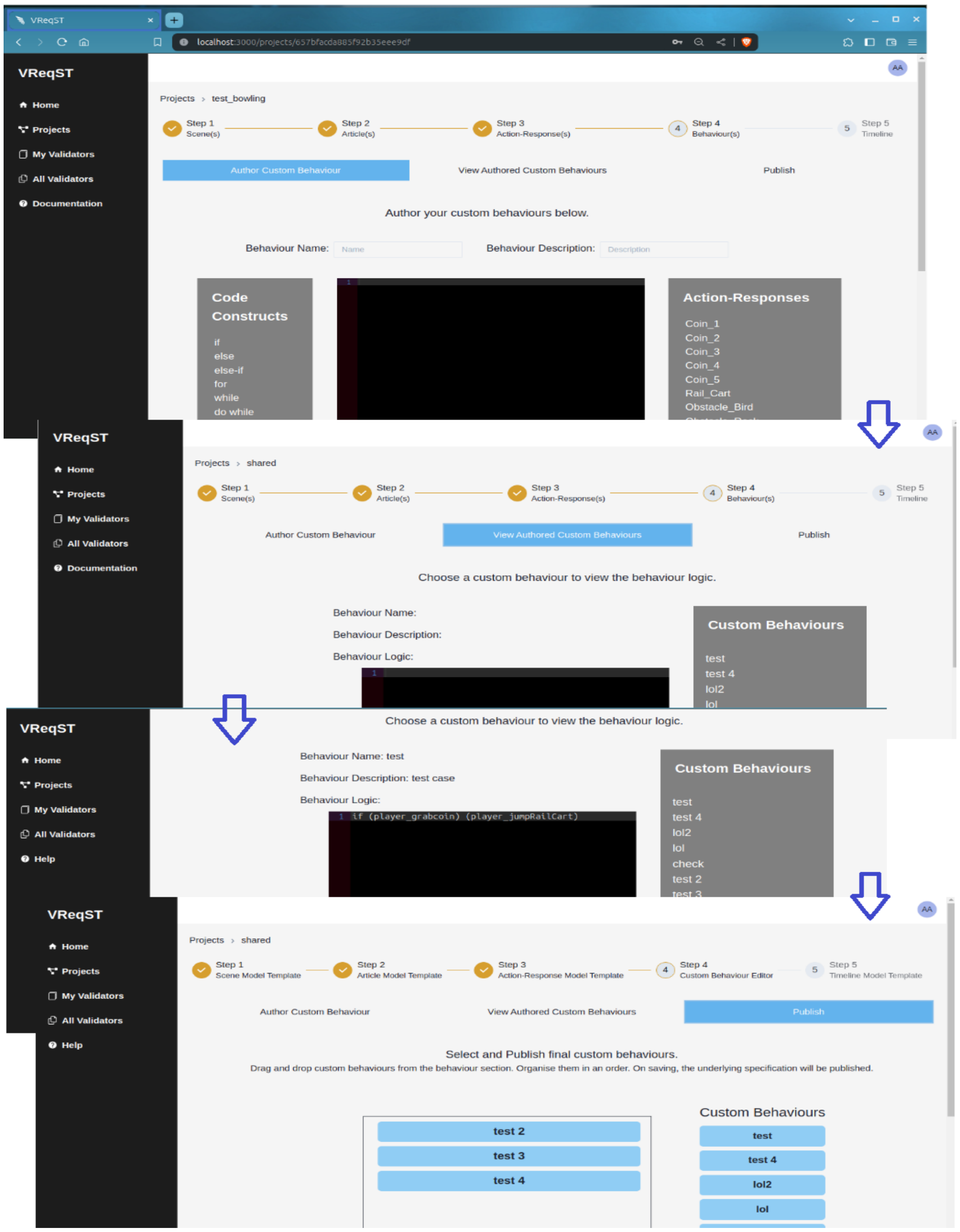
Figure 7.5 Revisions to Article and Action-Response Model Template

them separately and view them individually. For example, from Fig 7.5 under Step 3, i.e., action response, we may find examples of specifications of action-response between player and coin as *player_grabcoin, player_steercart, player_jumprailcart,* etc.

- **Behavior Editor Revisions** - As part of this iteration, we revised the entire workflow of Step 4, i.e., the *Behavior editor* formerly known as *"Custom JSON Step."* This step has three modes: (1) *'author behaviors'*, view authored behaviors, and publish the specifications from the list of authored behaviors. As shown in Fig 7.6, the author behavior mode contains three sections, code constructs (if, if-else, for, while, etc.), and a list of articles from the previous step. The requirement analyst will drag and drop the code constructs and articles to define the behavior conditions. (2) *'view author behavior mode'* contains a list of options to view the list of behaviors from the 'author behaviors' mode. (3) *'publish'* mode contains a list of opens to drag and a list of behaviors to be included in the final validation list of behavior step. These three modes can be visualized in Fig 7.6.

- Other minor revisions include text re-sizing across different pages, including additional fields on the validator file picker page, and a few minor implementation changes to make it simpler to deploy.

After considering thorough feedback from the VR Community, we updated the VReqST tool in three iterations and released it as an open-source tool. In contrast to this feedback, we surveyed the participants to understand the impact and overall experiences of using VReqST in practice. We present more details in this regard in the following subsection.



Author your custom behaviours below.

Behaviour Name: Name Behaviour Description: Description

Choose a custom behaviour to view the behaviour logic.

Behaviour Name:

Behaviour Description:

Behaviour Logic:

Choose a custom behaviour to view the behaviour logic.

Behaviour Name: test

Behaviour Description: test case

Behaviour Logic:

```
if (player_grabcoin) (player_jumpRailCart)
```

Select and Publish final custom behaviours.

Drag and drop custom behaviours from the behaviour section. Organise them in an order. On saving, the underlying specification will be published.

Figure 7.6 Revisions to Behavior Editor to Review, Author and Publish custom behaviors

7.1.4 Observations - Impact and Experience Survey

Between December 2023 and January 2024, We reached the 53 VR practitioners who participated in the VReqST iterative Validation study to participate in an Impact and Experience Survey[4] to provide their overall experience of their team on using VReqST and their experiences in utilizing the specifications in practice. 26 VR practitioners among 53 have responded to the survey. Following are the detailed observations from the survey.

Demography of Participants: Most organizations that participated in this Impact and Experience survey have been building enterprise VR products for about six years on average, with two years being the minimum tenure of a given organization on a lower end and 20+ years for the maximum tenure of a given organization on a higher end on building VR products from countries US, UK, India, China, Europe and Australia. Among the 26 participants, we have Engineering Manager(s), Engineering Directors with various capacities, Product Managers, Technical Architects, and Software Engineers who work in XR, an Emerging Technology domain have participated. The participants of the survey primarily build VR products in the following domains: Health Care, Defense, Simulation, Digital Twins, Interior design, Inner environment design, Industrial layout, Corporate Education, Compliance Training, Entertainment, Gaming, Drug discovery, Medical protocol design, Gaming Content, Cinema, Retail marketing, Marketing Campaigns, Authoring Tools, Metaverse, Fashion, Ads, Wearables, Financial Analytics, Tourism, Consulting, Enterprise Banking and Recreational tours. These practitioners are from Deloitte Digital, Plug XR, CornerstoneOndemand, SAP-XR, Samsung Studios, SkillSoft, UNITY Developer Group, ThoughWorks Dev Group, Khronous Dev Community, and many other VR open source practitioners helped us provide their valuable feedback.

Compatibility: Fig 7.7 illustrates that most participants observe that the VReqST tool is compatible with their existing processes and can be easily integrated without any hassle. Primarily, small organizations with an organization size of less than 200 employees have found VReqST to be highly compatible with their regular VR product development processes.

Ease of Use (UI/UX): Fig 7.8 illustrates that most participants observe that the VReqST tool is intuitive, considering its user interface and overall user experience seem manageable in practice. Small organizations with an organization size of fewer than 200 employees have found VReqST to be highly consistent with its UI design and ease of use in practice.

Frequency of Use: Fig 7.9 illustrates that most participants who started using VReqST frequently or occasionally for authoring requirement specifications for their VR products have mentioned that VReqST can specify requirements in detail irrespective of the organization's size. In almost all cases, the VReqST is highly capable & easy to specify requirements, or it is manageable compared to their con-

[4]https://forms.office.com/r/LVsaBEyHs5

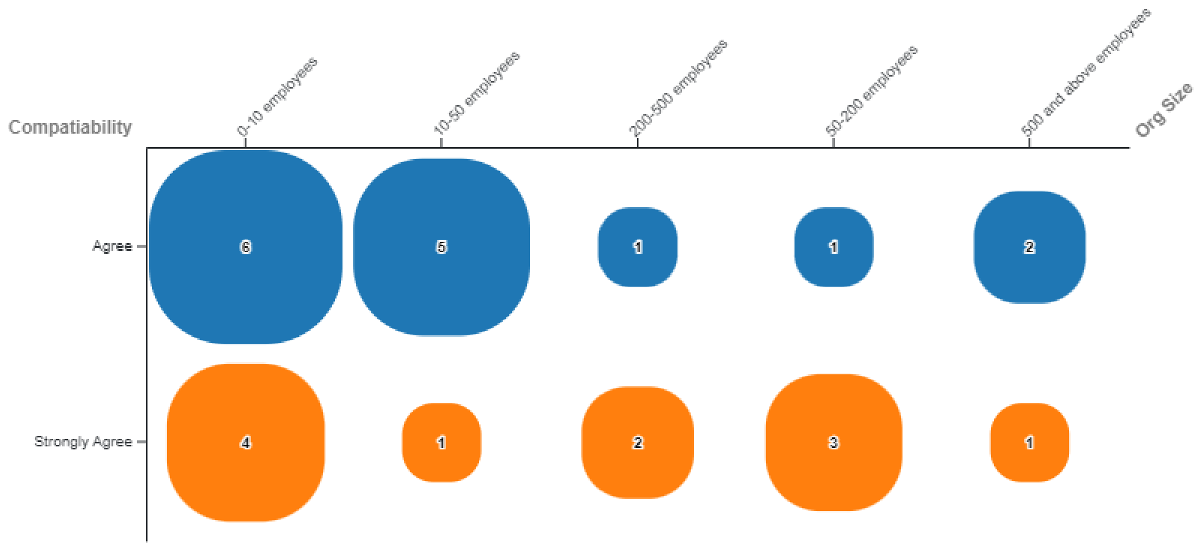


Figure 7.7 Compatibility of VReqST with existing processes group by Size of Organization

ventional approaches.

Feature Tracking: Fig 7.10 illustrates that most participants who started using VReqST frequently or occasionally for authoring requirement specifications for their VR products have mentioned that VReqST can easily track and trace requirements irrespective of the organization's size. In almost all cases, the VReqST is highly capable & easy to track and trace requirements to overall features across the requirement specifications of overall VR products when compared to their conventional approaches.

Extendability: Fig 7.11 illustrates that most participants who started using VReqST for authoring requirement specifications for their VR products have observed that VReqST is highly extendable and adaptable in practice. It can be extended to accommodate new model attributes, as the underlying model validator can be customized for domain-specific or context-specific applications. Most small and mid-range organizations working on context-specific VR products observed VR as a better alternative to specifying requirements to meet their business goals.

Additional Observations: Along with the survey questionnaire, we asked the survey participants to share their additional insights on adopting and using VReqST. The following subsection illustrates detailed responses in participants' words, which we have coded into specific themes for better illustration.

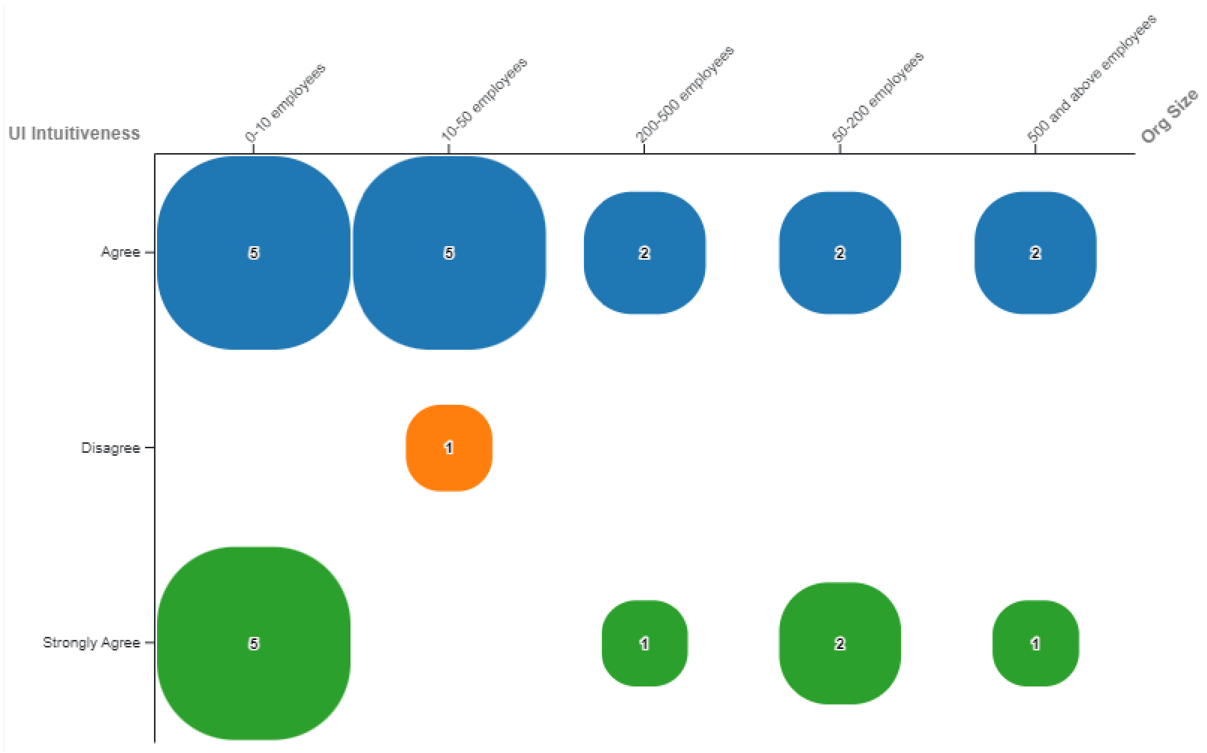


Figure 7.8 Ease of UI/UX of VReqST in practice group by Organization Size

7.1.5 VReqST extensions in VR product development

With the VR community practitioners' exposure to VReqST, the following ideas are suggested as future extensions to the VReqST tool through the survey. Using Open Coding approach, an grounded theory approach to generate themes out of contextual comments captured through a survey. *Code Generation, Test Case Generation, Risk and Dependencies, Traceability, Maintenance and Portability, New Requirement Types* are the generated themes. We detailed them as follows along with the citation of comments from the respective participant.

Code Generation - VReqST-based requirement specifications can be extrapolated further on generating model-driven code generation tools for scenes and 3-dimensional articles. Irrespective of the VR technology stack, VReqST specifications can be used to develop new tools that can generate alternate versions of code snippets for behaviors that could be generated on-demand during the scene play. It is one of the critical future aspects of extending VReqST into automated scene generation and behavior generation on run time.

- *In most cases, VReqST specifications can be used into automation pipeline. This is yet to be explored. We are looking at expanding the underlying template to support security requirements.* **[P4]**

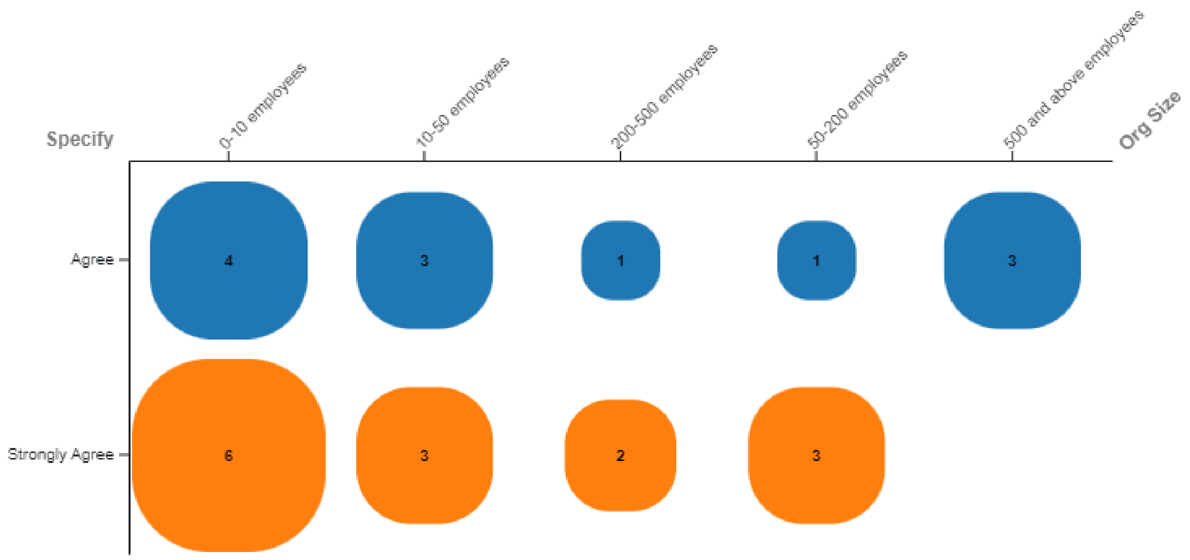


Figure 7.9 Ability to Specify VReqST specifications by frequency of usage across organizations

- *It helps Consultants to gather requirements. Needs more automation to channel the JSON spec to codegen and testing.***[P12]**
- *Very detail. This tool will have no value if this is not pulled into a automation pipeline. There is need to build supporting tools like co-pilot to build code with VReqST Specs.***[P15]**
- *Provides ability to record requires specs with higher precision. Need this to be part of automated delivery pipeline for more adoption.* **[P17]**
- *We just entered XR market with a garment wearable product. We are looking at integrating the JSON spec to asset generation using AI models. Thanks for keeping this opensource.* **[P20]**

Test case Generation- As VReqST curates the requirements in detail, automated test case generation of scene dynamics, article properties, action responses, behaviors, and timeline of events can be effortlessly explored to comprehend the quality of a delivered VR scene. A new novel customizable software testing framework can be developed out of VReqST as a external wrapper to generate on-demand test cases during run-time to improve the performance of the VR scene.

- *We started as pilot project with two Requirement analysts. We found VReqST to be high extendable. The generated specifications can be used as input source for our Test case generation system to conduct unit and integration testing. We are planning to extend our specifications for automated design validations. Thank you for collaboration.* **[P1]**
- *We are building games for about 20 years now with highly templatized terrain data and internal asset repository. We might have to do backward re-engineering of existing assets to this spec for automated scene, test case generation and interaction-flow testing. This tool definitely helps newbies, Good Luck with the tool.***[P23]**

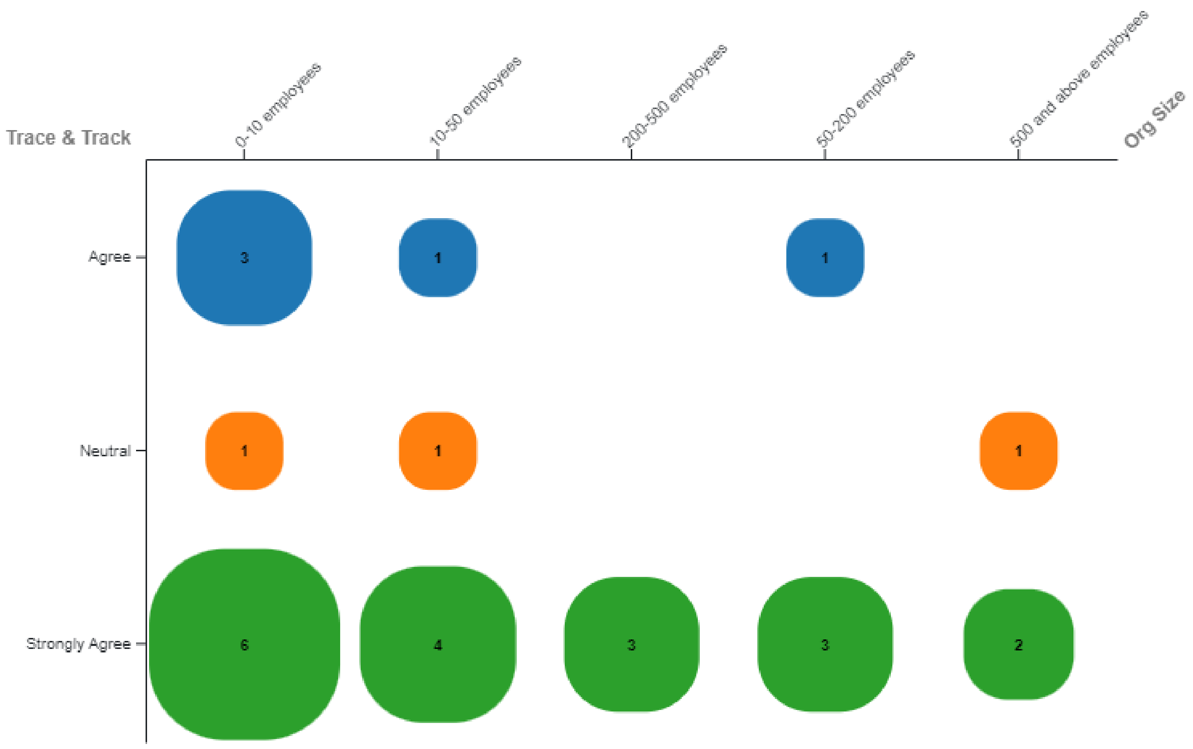


Figure 7.10 Ability to track VReqST requirements by frequency of usage across different class of organizations

Risk and Dependencies - With detailed VReqST requirements, the requirement analyst can determine the risk of prioritizing or extending new features. The requirement analyst can easily understand the overall dependencies of articles and their action responses over each other. These dependencies will help the requirement analyst estimate the consequence of the continuous evolution of a feature in VR and contribute towards a quality VR product.

- *Although it requires some effort, but VReqST output is a good input source for our low-code platform. Most of the asset and behavior generation can be automated to some extent. The specifications are easy to maintain and versioning. Most of our designers rely on the JSON text to go through the specific for revisions.* **[P2]**
- *Easy to integrate into our establish processes. Easy to author small requirements and later trace it to a large specification. Helps is maintaining requirement repository for dedicated projects in regards to assets and terrain definitions. Helps re-draw scope and risk on failing to meet the requirement.* **[P9]**
- *Plugable to internal VR authoring tools. Easy to determine the risk requirement failure in implementation. However, it is difficult to read requirement in JSON format. Need a better way to read the VReqST specification in detail.***[P10]**

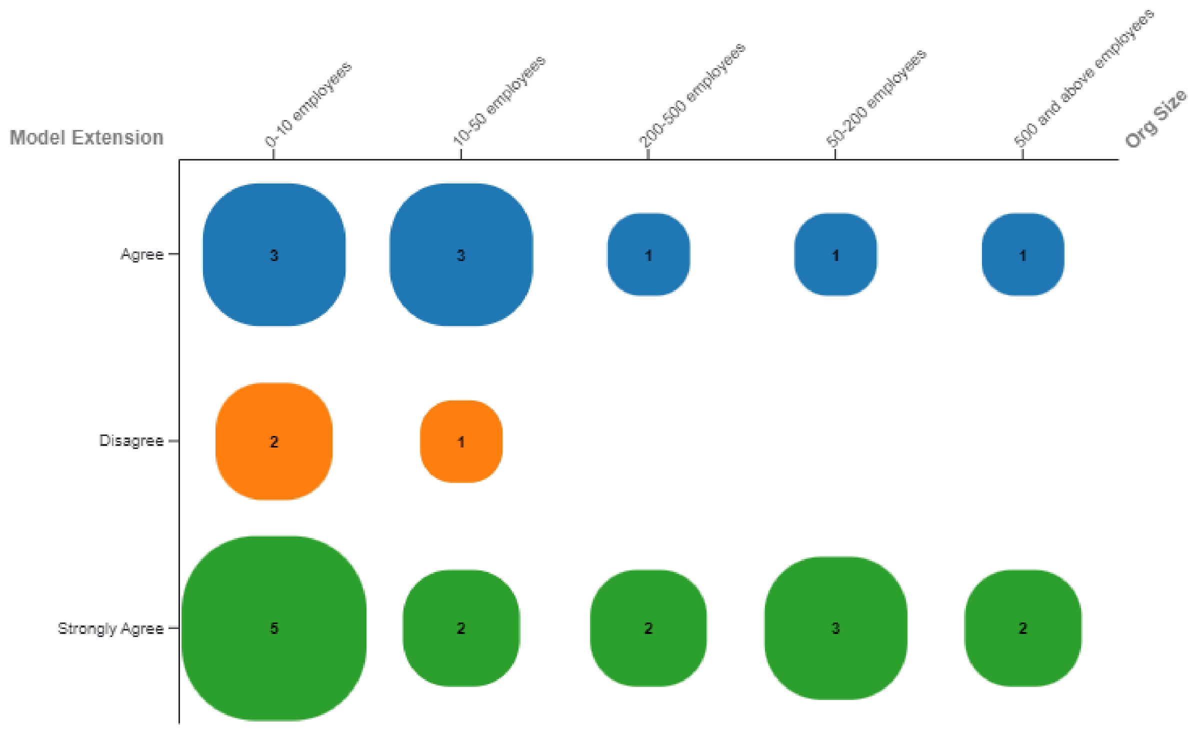


Figure 7.11 Ability to extend VReqST to Domain specific applications group by ability to specify requirements based on conventional standards across different class of organizations

- *VReqST may not be much useful as we have curated fixed pre-defined visualization templates. However, it helps when we have to redesign a new visualization template. reusability and traceability of reporting feature is easy to notice from specification to template. We may have to less adaption but may become vital in specific business cases. We suggest to extend a AI mode to predict/propose visualization templates for recorded VReqST specifications.* **[P18]**

Traceability - As the VReqST requirement specifications establish a relationship between VR concepts like scenes, articles, action responses, and timelines - the changes made in one concept will reflect across other concepts. These specifications support design traceability to understand the rationale behind design decisions. New tools can be developed along side VReqST to handle code traceability to aid the purpose of code segments and helps in debugging. Such new tools can support test traceability to help testers to identify gaps in test-case coverage.

- *Need to put in more effort to define our product specific validator. We cannot use the tool with our own business/context specific validator. Easy to version and trace the specifications. For RAs it is difficult to ready JSON specifications as they are not technically skilled.* **[P5]**
- *Traceability of feature is key. Easy to author micro requirements.* **[P24]**

Maintenance and Portability - With substantial requirements, the challenges with VR platform portability, i.e., porting VR scenes developed using a particular technology stack to another, can be mitigated to

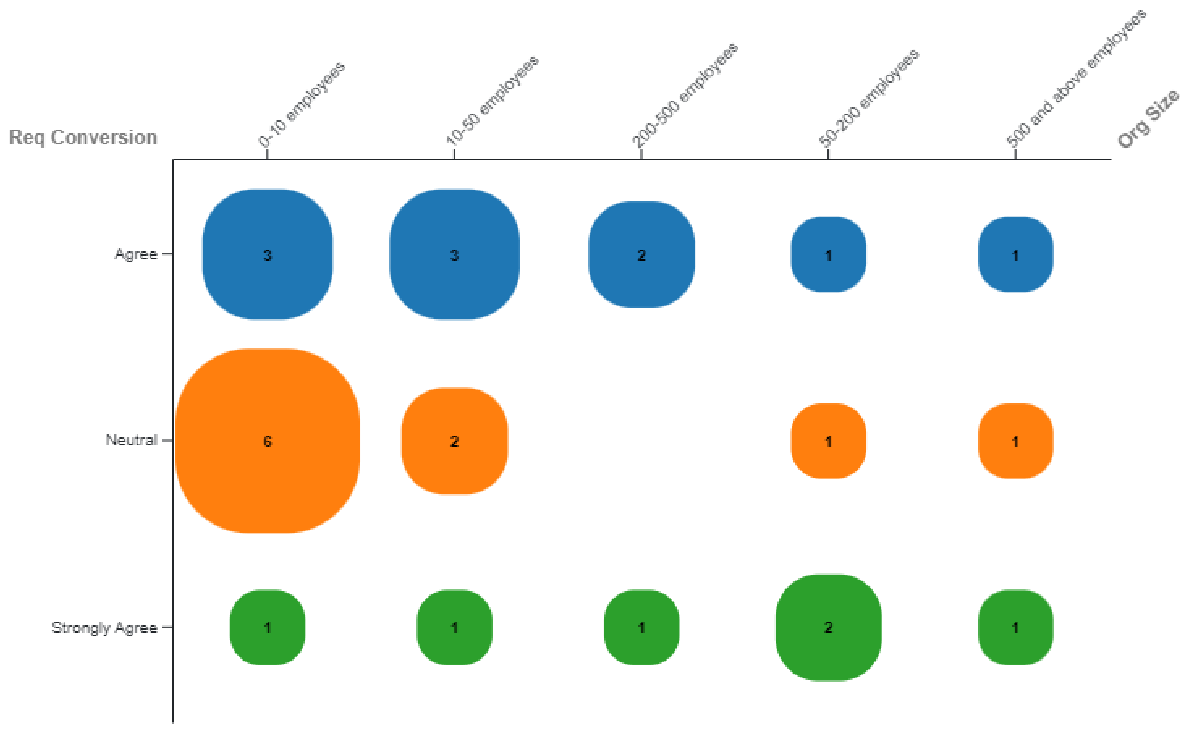


Figure 7.12 Ability to convert conventional specifications to VReqST Specification

some extent. As VReqST can distinguish platform-independent requirements from platform-dependent, the VR developers can now maintain platform-independent source code independently from the regular code branch. Figure 7.12 provides us the ability to port conventional requirement specification into VReqST specification. Such a practice may improve the maintainability and versioning of VR source code.

- *We are still in early stages of adoption. Our analysts are not tech savvy to use this tool. But our developers like it as it helps them understand requirements with higher accuracy. We still need to explore its capabilities on portability.***[P3]**
- *When we started, we had to author our own validator. It is easy to integrate but requires a lot of early effort. The first start move is difficult. Require a strong community support to reuse and share the artifacts specifications. There is a need for a asset specification repo in github etc.***[P6]**
- *We most use it for AR ad campaigns. Easy to use and manage specifications.* **[P25]**
- *We can also author requirements for AR applications by updating the underlying validator. The specifications are portable between VR only and MR applications even if they are authored for AR applications. Need to put it extra effort to develop underlying template for our own business need.***[P8]**
- *It requires a lot of effort to understand and update the domain specific validator. Require more community support.***[P11]**

- *The requirements are too detail. Difficult to illustrate large terrains with such high specifics. However, easy to author micro requirements.* **[P14]** *Our use-cases are high domain specific. We need to put alot of efforts to develop the underlying validator. For now, we are better with our conventional approach to capture requirements in templates.* **[P22]**

New Types of Requirements - The current VReqST version does not facilitate security, privacy, and usability requirements. The VR community recommends facilitating additional features to VReqST, including customizable security, privacy, and usability requirements, as part of the new model template. These requirement types are non-generic and domain-centered. They may vary from application to application. As VReqST is customizable, the security, privacy, and usability requirements can be included beyond the bare minimum model template. It can be managed by domain-specific requirement analysts to accommodate such new requirement types by composing a custom model template on par with bare-minimum model template.

- *The specificity of requirements is maintained through this tool. It requires training as our BAs are not enough technical to author specifications in JSON. Request to provide a form based template that can convert to JSON in backend.***[P7]**
- *Request to include privacy related specification validator as default. It helps cover compliance use-cases easily.***[P13]**
- *Fashion requires clear specifics. VReqST fills that gap. It is slightly painful to write our custom validator. But easy to maintain and re-trace the specifications. Need strong collaboration among XR community partners to share open source asset specs to re-shareability.* **[P16]**
- *Currently our plain english req specs are being converted to Vreqst specs. It helps us create large terrain templates for future touring projects. Behavior editor is coolest feature. Easy to author and track linkages between the interactions.* **[P19]**

7.2 Requirement Specifications for Depression Detection Application using VR: a Proof of Concept

Background: Between January 2023 and October 2023, researchers from Software Engineering Research Center[5] and Cognitive Science Lab[6] from IIIT Hyderabad worked towards developing multiple Virtual Reality Scenes for the detection or intervention of depression using VR technology. These VR scenes are used as a medium to evoke emotions that are deemed to result in positive, negative, and neutral emotions. These three emotions are evoked using positive, negative, and neutral environments. Fig 7.13, 7.14, and 7.15 illustrate the positive scene, negative scene, and neutral scene with top view and side views. The requirement specifications of these VR scenes are first authored using VReqST, they are revised and then finalized for design and development. These VR Scenes are developed in

[5]https://serc.iiit.ac.in
[6]https://sites.google.com/view/ps-pacgroup/home

UNITY Game Engine [7] and are visualized using HTC Vive Pro 2. This headset has a dual RGB low persistence LCD screen with a resolution of 2448 × 2448 pixels per eye (4896 x 2448 pixels combined) and a refresh rate of 90/120 Hz. It has a field of view of 120 degrees and requires two compatible base stations and two compatible motion controllers. We used a PC with RAM 16GB and graphics Card 970.

Experiment: The three developed VR scenes are designed to evoke the participants' positive, negative, and neutral emotions while locomoting in the virtual environment. While the participant is under locomotion under the influence of a VR scene, the participant's gait pattern is captured using a VelGmat [275], i.e., a Velostat-based gait mat that captures gait pressure. The gait pattern under the influence of positive, negative, and neutral scenes will reveal the levels of depression in a given participant as the scene evokes emotions. The participant is required to walk around in the environment once clockwise around the right shelf and once anti-clockwise around the left shelf, and in each round, while returning, pick up the specified object and place it into the respective crate.

VR Scene Requirements: All three environments have a 10 x 2 feet aisle with six feet-high shelves on both sides, two feet apart. There are two similar aisles on the right and left that allow the players to circle the right and left shelves clockwise and anticlockwise, respectively. The walking area is separated from the respective terrain by a glass wall. The right aisle has an opaque wall with images of positive and negative environments. Both shelves are stacked with four objects on three different levels. The aisles on the sides have crates in the corners.

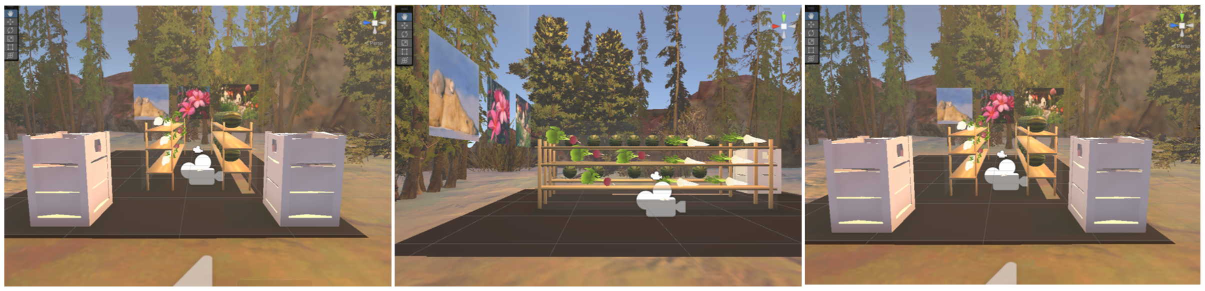


Figure 7.13 Scene flow of Positive Scenes in Mental Health VR Application

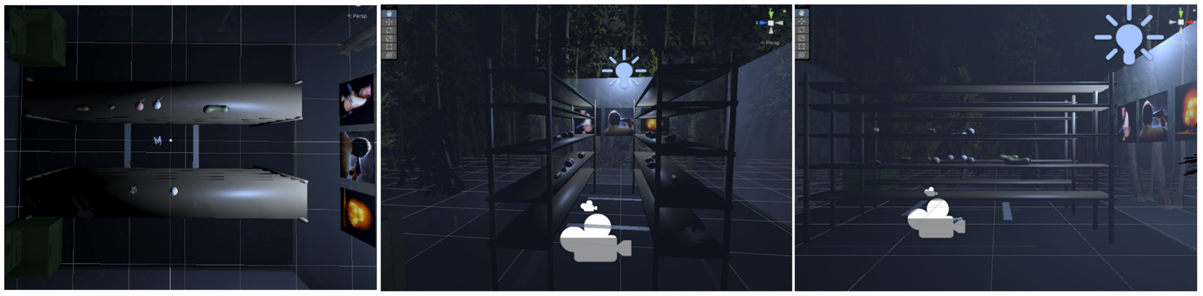


Figure 7.14 Scene flow of Negative Scenes in Mental Health VR Application

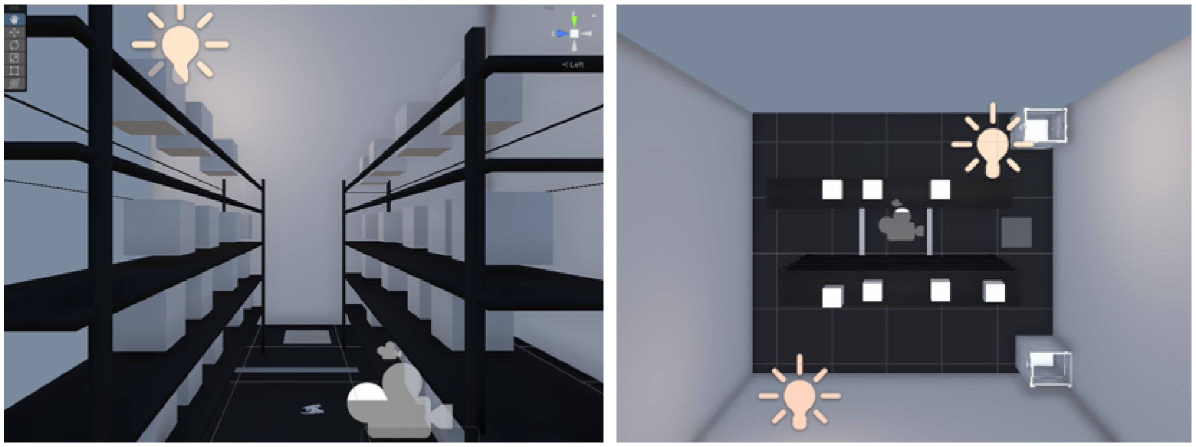


Figure 7.15 Scene flow of Neutral Scenes in Mental Health VR Application

- **Positive Environment** - Fig 7.13 illustrates the top and side view of the positive scene. The goal of the positive scene is to evoke happiness in the participant. It is primarily achieved using bright and warm lighting source with saturated colors with a pleasant background music. The environment is filled with forest terrain with trees around a central plan with two racks separated by a walkable area. The racks contain few objects. The two large crates on two ends of the center pane are on one side of the scene, whereas the other end of the center pane contains images floating on a virtual wall. The objects in the scene are primarily under color-grade shades of light blue and light pink for better differentiation. The images on the display wall are selected from the IAPS database[7] with high valance ratings (above 50). The International Affective Picture System (IAPS) is a database of pictures designed to provide a standardized set of pictures for studying emotion and attention. It is widely used in psychological research. The tree assets used in the forest were imported from the Unity Demo URP terrain.

- **Negative Environment** - Fig 7.14 illustrates the top and side view of the negative scene. The goal of the negative scene is to evoke sadness in the participant. It is primarily achieved using dark and low-intensity lighting source with dull colors along with a disturbed background music. The environment is filled with forest terrain with trees around a central plan with two racks separated by a walkable area. The two large crates on two ends of the center pane are on one side of the scene. The racks are bulkier and contain too many objects, thus making the space feel more congested. One end of the center pane contains images floating on a virtual wall. The objects in the scene are primarily under color-grade shades of dark brown and dark green, making them difficult to differentiate. The images on the display wall depict decay and suffering, selected from the IAPS database with very low valance ratings (less than 50). The International Affective Picture System (IAPS) is a database of pictures designed to provide a standardized set of pictures

[7]https://www.imageemotion.org/

for studying emotion and attention. It is widely used in psychological research. The tree assets used in the forest were imported from the Unity Demo URP terrain.

- **Neutral Environment** - Fig 7.15 illustrates the top and side view of the neutral scene. This controlled environment is designed to collect participants' gait data without emotional elicitation in the VR. The Scene contains black and white colors on all the objects, with no Images on the wall or music.

The VReqST specification of the proof-of-concept application are made available as part of our resources [136]. The resources contain the model template files and their attributes related to positive, negative and neutral scene.

Conclusion: We evaluated VReqST - the requirement specification tool in practice by working with practitioners from the VR Community. The practitioners from Deloitte Digital, Plug XR, CornerstoneOndemand, SAP-XR, Samsung Studio, SkillSoft, UNITY Developer Group, ThoughWorks, Khronous Dev Community, and many other VR open source practitioners helped us provide their valuable feedback. We revised the VReqST tool in three iterations and released it as an open-source tool for further adoption. As the VR meta-model backs VReqST, it can significantly improve overall VR product development through a Model-Driven development approach for VR products. We further detail this as our future work in following chapter.

Chapter 8

Deploying and Configuring VReqST in Practice

VReqST - the requirement specification tool can be implemented as a single host instance residing on a single server for local user consumption in closed networks and as a Software-as-Service instance on the cloud for sharing across the teams in an open network. This section describes the steps required to deploy and configure the VReqST tool in practice.

8.1 Hardware Requirements

Following are the bare-minimum hardware requirements for deploying VReqST.

- **Client Machine (end user):** A minimum of a Pentium 4 processor with at-least 128MB of RAM is needed, while 256MB is recommended for optimal performance. A minimum of 100MB of free disk space is necessary, with 200MB or more recommended for efficient operation. Ensure that you have enough storage capacity to accommodate your desired web browser and any associated files, such as bookmarks and cache. Any standard display should suffice, as long as it allows you to view web pages comfortably. Keyboard and mouse are essential for interacting with websites and inputting text. Touchpads or touchscreens can also serve this purpose. Additionally, a network connection is crucial for browsing the internet. This can include wired connections via Ethernet cables or wireless connections using Wi-Fi or cellular networks.

- **Server Machine (VReqST Host Server):** A multi-core processor with a high clock speed is ideal for handling heavy traffic and resource-intensive applications. Look for a processor with at least 8 cores and a frequency higher than 3 GHz. Ensure you have enough RAM to accommodate simultaneous user requests and prevent your website from slowing down or crashing. If your website receives a lot of traffic, consider a dedicated server with at least 16 or 32GB of RAM. Choose a storage option that fits your capacity and performance needs. While traditional hard disk drives (HDDs) are cheaper and offer a lot of storage space, they have relatively slow reading and writing speeds. Solid-state drives (SSDs) are faster and more expensive but provide better performance. 1 TB of ROM is recommended.

8.2 Software Requirements

Following are the bare-minimum software requirements for deploying VReqST.

- **Client Machine (end user):** Any basic operating system, like Windows 10 or above, MacOS, or any Linux variant, is required on a client's machine. A web browser to browse the VReqST application and its related documentation site. TCP/IP protocol suite to communicate with network and gain access to internet.
- **Server Machine (VReqST Host Server):** In addition to the client machine software requirements, deploy git from https://git-scm.com to access VReqST source code from Github version control. Deploy Node.js version 14 from https://node.js.org and install it on the server machine.

8.3 Deployment Steps

Following are the detailed steps to deploy VReqST tool on the server machine.

- Open git terminal and use the command and install

```
$ git clone https://github.com/Amogha027/VReqST-2
```

- Now navigate to VReqST-2, VReqST-main and VReqST folder and run the following command on the terminal

```
$ cd VReqST-2/VReqST-main/VReqST
```

- This will deploy the VReqST on the server machine.
- The backend_server and validation_server instances are already hosted on render.com[1]. This will build and executes our project code into production dynamically.
- Alternatively, to host the application on hosted instance, update the environment URLs of **validation_server** and **backend** in the following path of node.js application through command terminal - *'frontend/client/src/server_urls.jsx'* as follows

```
    $ export const validation_server = "http://localhost:5001"
    $ export const backend = "http://localhost:5002"
```

- To start the backend_server on port 5002, navigate to VReqST folder on command terminal and run following.

[1]https://render.com/

```
$ cd backend
```

- To start validation_server on Port 5001, navigate to VReqST folder on command terminal and run following

```
$ cd validation_server
$ npm install
$ nodemon index.js
```

- To start the application on port 3000, navigate to VReqST folder on command terminal and run following

```
$ cd frontend
$ npm install
$ npm run client-install
$ npm run dev
```

- Upon running the application, the VReqST will be launched on the default browser with a local site http://localhost:3000. You may allow the site access to intranet by providing a desired domain-name and grant access to targeted user-group.

8.4 Configuring Database

: Following are the detailed steps required to configure MongoDB database instance on-demand remotely.

- Access https://mongodb.com and create an account. Create a new project and add a new database to the project.
- Define 4 collection with following naming convention - customrules, jsons, projects and users.
- This will create a new dedicated isntance of the database that is required to be linked to your VReqST.
- You may access the collection data under databases listed on https://mongodb.com
- In case of any additional details, please refer to the help section on the VReqST website that will redirect to the official documentation[2] and a demo video[3]

[2]https://saianirudh-karri.gitbook.io/vreqst
[3]https://www.youtube.com/watch?v=G9-aW0vW8uA

Chapter 9

Future Work: Model Driven Development for Virtual Reality Software

Model-driven development (MDD) is a software engineering methodology that emphasizes the creation and use of domain models. These are conceptual models representing various aspects of a software system. They serve as blueprints for software development, guiding the process from design to code generation. MDD involves abstracting technical aspects such as logic, data models, and user interfaces into visual representations that can be easily manipulated. The ultimate goal of MDD is to enhance productivity, improve software quality and facilitate collaboration among developers of varying skill levels. In the context of Virtual Reality Software Products, our role-based model template can be useful in formulating a model-driven development approach for Virtual Reality Software Products using domain-specific Large Language models and Generative AI.

9.1 Few Concepts

Domain-specific LLMs (Language Learning Models) are designed to specialize in a specific industry domain or use case, focusing on capturing the nuances and terminology unique to that domain. By following below steps and techniques, developers can create powerful and effective Domain-Specific Large Language Models tailored to specific industries or fields, providing more accurate, reliable, and efficient outputs within those domains.

- **Prompting:** A well-crafting question or prompts that guide the model in generating outputs tailored to a specific domain. Providing specific instructions to the model can produce more relevant and accurate responses within a particular field.

- **Fine-Tuning:** Fine-tuning a generic LLM on a targeted dataset specific to the domain is crucial. This process involves training the model on domain-specific data to enhance its performance within that specialized area. Fine-tuning allows the model to understand industry terminologies and nuances better

- **Retrieval Augmented Generation (RAG):** RAG combines the strengths of information retrieval and LLMs. By connecting the model to external knowledge sources like databases, it can fetch relevant information to generate responses. This technique is valuable for providing explainable and accurate answers, especially in fields where up-to-date information is crucial

- Domain Expert Review: Building a domain-specific LLM requires collaboration with domain experts to curate training datasets, annotate data, and evaluate the model's performance accurately. This collaboration ensures that the model captures the specific nuances and requirements of the domain effectively.

- **Evaluation:** To measure the performance of a domain-specific LLM, comparing it to ground truth data is essential. Ground truth data, which is annotated datasets used for evaluation, helps assess the model's generalization and performance, allowing for adjustments to enhance its effectiveness

Generative AI refers to artificial intelligence models capable of creating new content, such as text, images, audio, or synthetic data, based on a given prompt or input. These models are trained on vast amounts of data and employ advanced algorithms, such as transformers, to generate novel outputs that often appear indistinguishable from human-generated content. Examples of generative AI include ChatGPT[1], a chatbot that can generate text, and DALL-E[2], a tool that creates images from text descriptions.

Generative AI has numerous applications across various industries, including entertainment, education, and business optimization. However, it also poses challenges, such as the potential for misuse in creating deepfakes or generating content that infringes on copyright laws. Despite these concerns, the field of generative AI continues to evolve rapidly, with ongoing research aimed at improving the accuracy, reliability, and safety of these systems.

9.2 MDD Pipleline for VR

Our Role-based model template for comprehending the Virtual Reality Technology domain is a backbone for formulating Model-driven development for developing Virtual Reality Software Products. Using our Role-based model template for VR, we developed an open-source requirement specification tool called VReqST. This tool can apprehend bare-minimum concepts related to the VR technology domain. It can generate precise and clear specifications that can become an input to our MDD pipeline for Virtual Reality.

Fig 9.1 describes a development pipeline for Virtual Reality software. During the requirement phase, VReqST can be used to specify requirements which are then used as input to a 3-dimensional Large Learning Model (LLM) that is specific to VR domain. The input will be in a JSON format. In the design phase, the 3-D LLM is trained using 3-D object data sources that offer data in the form of point clouds, voxels, or meshes to generate multiple desired articles, mock design templates for terrain, 3-dimensional environment, and respective action-response flow.

[1]https://chat.openai.com/auth/login
[2]https://openai.com/dall-e-2

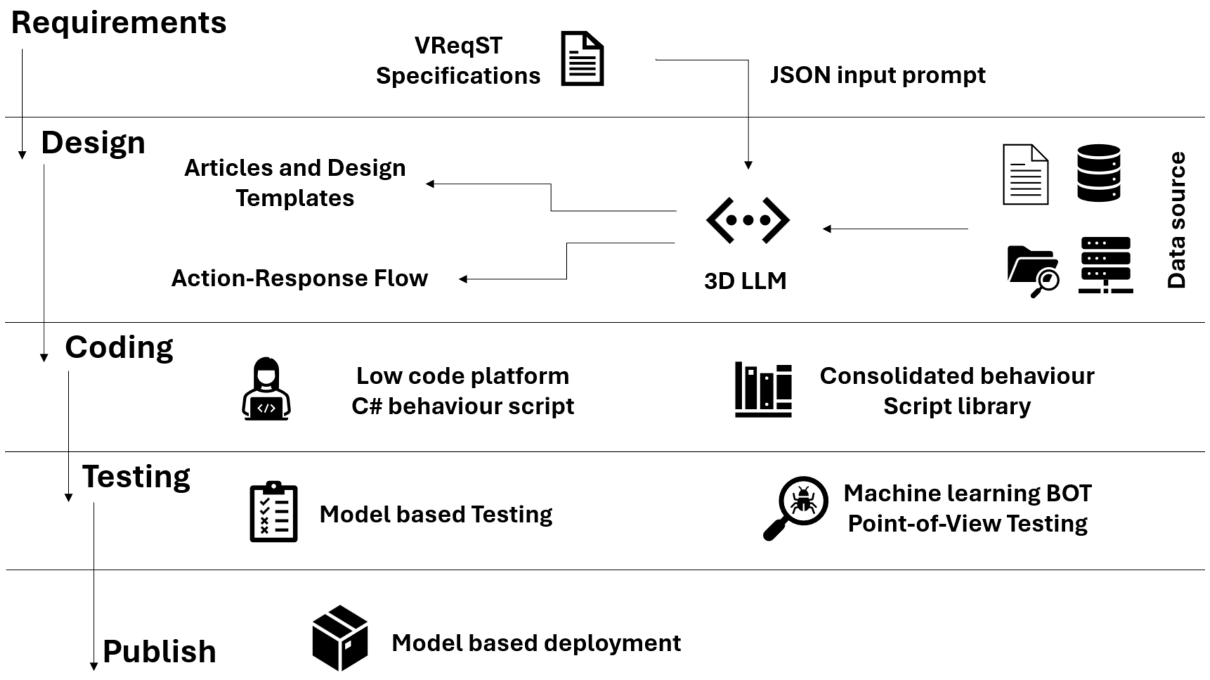


Figure 9.1 Model Driven Development using LLM and GenAI for Virtual Reality Software Products

The output of the 3-D LLM can be used as input to a low-code platform that generates the behavior script for specific VR game engines and creates a consolidated behavior script library for future reference. A model-based testing protocol can be formulated for these behavior scripts based on the generated code snippets. The overall 3-D run-through can be evaluated using a machine learning BOT through point-of-view testing. The tested artifact can now be deployed using a model-based deployment approach, where changes reflected in the model (scene artifacts, article artifacts, action-response artifacts, behavior artifacts, and timeline artifacts) can be deployed asynchronously in an incremental approach to production.

However, despite various automated processes involved in the proposed overall MDD pipeline for Virtual Reality software development, it is partially a human-in-a-loop process. Human validation is required in a few stages to validate the exactness and quality of the resultant VR Scene. This hypothesis requires a lot of conviction to execute at a large scale.

Related Publications

Below is the list of publications related to my Ph.D. thesis group by year and type of publication. They support the evidence of my research and helped deduce conclusions to the overall requirement engineering practices for improving Virtual Reality software product development.

Journal Publication

1. Sai Anirudh Karre, Raghu Reddy. 2024. ***Model Based Requirement Specification for Virtual Reality Software Products***, Frontiers Journal on Virtual Reality, May 2024 [Submitted]
2. Sai Anirudh Karre, Y. Raghu Reddy, and Raghav Mittal. 2024. ***RE Methods for Virtual Reality Software Product Development: A Mapping Study.*** ACM Trans. Softw. Eng. Methodol. 33, 4, Article 88 (May 2024), 31 pages. https://doi.org/10.1145/3649595
3. Sai Anirudh Karre, Neeraj Mathur, and Y. Raghu Reddy. 2020. ***Understanding usability evaluation setup for VR products in industry: a review study.*** SIGAPP Applied Computing Review Journal, Volume 19, Issue 4 (December 2019), 17–27. https://doi.org/10.1145/3381307.3381309

Conference Publication

1. Sai Anirudh Karre, Karthik Vaidhyanathan, and Y. Raghu Reddy. 2023. ***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 (CompEd 2023). Association for Computing Machinery, New York, NY, USA, 195. https://doi.org/10.1145/3617650.3624936
2. Sai Anirudh Karre, Vivek Pareek, Raghav Mittal, and Raghu Reddy. 2023. ***A Role Based Model Template for Specifying Virtual Reality Software.*** In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22). Association for Computing Machinery, New York, NY, USA, Article 225, 1–5. https://doi.org/10.1145/3551349.3560514
3. Sai Anirudh Karre, Raghav Mittal, and Raghu Reddy. 2023. ***Requirements Elicitation for Virtual Reality Products - A Mapping Study.*** In Proceedings of the 16th Innovations in Software Engineering Conference (ISEC '23). Association for Computing Machinery, New York, NY, USA, Article 12, 1–11. https://doi.org/10.1145/3578527.3578536
4. Mohit Kuri, Sai Anirudh Karre, and Y. Raghu Reddy. 2021. ***Understanding Software Quality Metrics for Virtual Reality Products - A Mapping Study***. In 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) (ISEC 2021).

Association for Computing Machinery, New York, NY, USA, Article 8, 1–11. https://doi.org/10.1145/3452383.3452391

5. Sai Anirudh Karre, Neeraj Mathur, and Y. Raghu Reddy. 2019. ***Usability evaluation of VR products in industry: a systematic literature review.*** In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19). Association for Computing Machinery, New York, NY, USA, 1845–1851. https://doi.org/10.1145/3297280.3297462
6. 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'19). Association for Computing Machinery, New York, NY, USA, Article 3, 1–11. https://doi.org/10.1145/3299771.3299772

I worked with many student researchers during my tenure as a Ph.D. student between 2017 and 2024 at the Software Engineering Research Center - IIIT Hyderabad. I mentored a few of them as part of their master's research program. The following is the list of publications and patents in which I was involved, collaborated with, and contributed to the research in software engineering, virtual reality, and human-computer interaction.

Patents

1. Y Raghu Reddy, Raghav Mittal, Sai Anirudh Karre, **System and method for generating a limitless path in virtual reality environment for continuous locomotion,** US11687154B2, IIIT Hyderabad, 2022, https://patents.google.com/patent/US11687154B2/en **[GRANTED]**
2. Neeraj Mathur, Sai Anirudh Karre, Y Raghu Babu Reddy, ***System and method for evaluating and facilitating customized guidelines using usability code pattern analysis***, US11144430B2, IIIT Hyderabad, 2019, https://patents.google.com/patent/US11144430B2/en **[GRANTED]**

Conference Publications

1. Dhiraj SM Varanasi, Sai Anirudh Karre, and Raghu Reddy. 2024. *Software Development Waste amidst COVID-19 Pandemic: An Industry Study.* In Proceedings of the 17th Innovations in Software Engineering Conference (ISEC '24). Association for Computing Machinery, New York, NY, USA, Article 13, 1–5. https://doi.org/10.1145/3641399.3641417
2. Sai Anirudh Karre, Mathur, N., and Raghu Reddy, Y., ***Assessing New Hires' Programming Productivity Through UMETRIX – An Industry Case Study***, arXiv e-prints, 2023. doi:10.48550/arXiv.2305.03332.
3. M. W. Wani, Y. P. Gururaj, V. P, Sai Anirudh Karre, R. Reddy and S. Azeemuddin, ***VelGmat: Low Cost Gait Mat For Stance Phase Calculation***, 2022 IEEE Sensors, Dallas, TX, USA, 2022, pp. 1-4, doi: 10.1109/SENSORS52175.2022.9967332.
4. Y. P. Gururaj, Sai Anirudh Karre, R. Mittal, Y. R. Reddy and S. Azeemuddin, ***Customizable Head-mounted Device for Detection of Eye Disorders using Virtual Reality,*** 2022 35th Interna-

tional Conference on VLSI Design and 2022 21st International Conference on Embedded Systems (VLSID), Bangalore, India, 2022, pp. 50-55, doi: 10.1109/VLSID2022.2022.00022.

5. Y. P. Gururaj, R. Mittal, Sai Anirudh Karre, Y. R. Reddy and S. Azeemuddin, ***Towards Conducting Effective Locomotion Through Hardware Transformation in Head-Mounted-Device - A Review Study***, 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), Christchurch, New Zealand, 2022, pp. 660-661, doi: 10.1109/VRW55335.2022.00181.
6. Raghav Mittal, Sai Anirudh Karre, Y Pawan Kumar Gururaj, and Y Raghu Reddy. 2022. ***Enhancing Configurable Limitless Paths in Virtual Reality Environments.*** In 15th Innovations in Software Engineering Conference (ISEC 2022). Association for Computing Machinery, New York, NY, USA, Article 24, 1–5. https://doi.org/10.1145/3511430.3511452
7. Mittal, R., Sai Anirudh Karre, Reddy, Y.R. (2021). ***Designing Limitless Path in Virtual Reality Environment.*** In: Chen, J.Y.C., Fragomeni, G. (eds) Virtual, Augmented and Mixed Reality. HCII 2021. Lecture Notes in Computer Science(), vol 12770. Springer, Cham. https://doi.org/10.1007/978-3-030-77599-5_7
8. Sai Anirudh Karre, Lalit Mohan Sanagavarapu, and Y. Raghu Reddy. 2021. ***Using project-based approach to teach design patterns: An Experience Report.*** In 14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference) (ISEC 2021). Association for Computing Machinery, New York, NY, USA, Article 16, 1–5. https://doi.org/10.1145/3452383.3452399
9. Sai Anirudh Karre, Abhinav Gupta, S Lalit Mohan, Y Raghu Reddy, 2021. ***Introducing Software System Course to Engineering Undergraduate Students-An Experience Report***, Proceedings of 4th Software Engineering Education Workshop (SEED 2021) co-located with APSEC 2021, 06-Dec, 2021, Taipei, Taiwan, CEUR Workshop Proceedings, Vol-3062, pp: 1-7, ISSN 1613-0073
10. Shekhar, Shivang, Sai Anirudh Karre, Y. Raghu Reddy, and Rajat Kumar Gupta. ***Detecting Astigmatism Condition in Human Eye using VR.*** (2020). EURO VR.
11. Shekar, S., Pesaladinne, P.R., Sai Anirudh Karre, Reddy, Y.R. (2020). ***VREye: Exploring Human Visual Acuity Test Using Virtual Reality***. In: Chen, J.Y.C., Fragomeni, G. (eds) Virtual, Augmented and Mixed Reality. Industrial and Everyday Life Applications. HCII 2020. Lecture Notes in Computer Science(), vol 12191. Springer, Cham. https://doi.org/10.1007/978-3-030-49698-2_28
12. Neeraj Mathur, Sai Anirudh Karre, Y. Raghu Reddy. 2019. ***Grouping Semantically Related Change-Sets to Enhance Identification of Logical Coupling***, The 31st International Conference on Software Engineering and Knowledge Engineering, DOI: 10.18293/SEKE2019-166
13. Neeraj Mathur, Sai Anirudh Karre, Lalit S. Mohan, and Y. Raghu Reddy. 2018. ***Analysis of FinTech Mobile App Usability for Geriatric Users in India.*** In Proceedings of the 4th International Conference on Human-Computer Interaction and User Experience in Indonesia, CHI-

uXiD '18 (CHIuXiD '18). Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3205946.3205947

14. Neeraj Mathur, Sai Anirudh Karre, and Y. Raghu Reddy. 2018. ***Usability Evaluation Framework for Mobile Apps using Code Analysis.*** In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (EASE '18). Association for Computing Machinery, New York, NY, USA, 187–192. https://doi.org/10.1145/3210459.3210480

Bibliography

[1] Iso/iec/ieee international standard - systems and software engineering–life cycle management–part 5: Software development planning. *ISO/IEC/IEEE 24748-5:2017(E)*, pages 1–48, June 2017.

[2] Market research report: Augmented/virtual reality report q2 2018. *Digi-Capital*, pages 1–234, 2018.

[3] Oculus developer program, 2018. Accessed: 2018-04-30.

[4] *AframeJS Documentation*. August, 2023.

[5] *CRYENGINE Programming Documentation*. Crytek Technologies, August, 2023.

[6] *Lumberyard Documentation*. Amazon Web Services, August, 2023.

[7] *Unity3D Manual - Offline Documentation*. UNITY Technologies, August, 2023.

[8] *UnRealEngine 5 Documentation*. EPIC Games Inc., August, 2023.

[9] P. F. d. Abreu, V. M. B. Werneck, R. M. E. M. d. Costa, and L. A. V. d. Carvalho. Employing multi-agents in 3-d game for cognitive stimulation. In *2011 XIII Symposium on Virtual Reality*, pages 73–78, 2011.

[10] A. Achberger, F. Aust, D. Pohlandt, K. Vidackovic, and M. Sedlmair. Strive: String-based force feedback for automotive engineering. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 841–853, New York, NY, USA, 2021. Association for Computing Machinery.

[11] N. Adamo-Villani and K. Wright. Smile: An immersive learning game for deaf and hearing children. In *ACM SIGGRAPH 2007 Educators Program*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[12] S. Adolph, W. Hall, and P. Kruchten. Using grounded theory to study the experience of software development. *Empirical Software Engineering*, 16(4):487–513, Aug 2011.

[13] F. J. Agbo, S. Sunday Oyelere, and N. Bouali. A uml approach for designing a vr-based smart learning environment for programming education. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, 2020.

[14] J. I. Akpan and R. J. Brooks. Experimental investigation of the impacts of virtual reality on discrete-event simulation. In *Proceedings of the 37th Conference on Winter Simulation*, WSC '05, pages 1968–1975. Winter Simulation Conference, 2005.

[15] S. Aleem, L. F. Capretz, and F. Ahmed. Game development software engineering process life cycle: a systematic review. *Journal of Software Engineering Research and Development*, 4(1):6, Nov 2016.

[16] D. Alexandrovsky, G. Volkmar, M. Spliethöver, S. Finke, M. Herrlich, T. Döring, J. D. Smeddinck, and R. Malaka. Playful user-generated treatment: A novel game design approach for vr exposure therapy. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '20, page 32–45, New York, NY, USA, 2020. Association for Computing Machinery.

[17] M. Ali and R. E. Cardona-Rivera. Comparing gamepad and naturally-mapped controller effects on perceived virtual reality experiences. In *ACM Symposium on Applied Perception 2020*, SAP '20, New York, NY, USA, 2020. Association for Computing Machinery.

[18] S. Altarteer, V. Charissis, D. Harrison, and W. Chan. Interactive virtual reality shopping and the impact in luxury brands. In R. Shumaker, editor, *Virtual, Augmented and Mixed Reality. Systems and Applications*, pages 221–230, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[19] C. Antonya and D. Talaba. Design evaluation and modification of mechanical systems in virtual environments. *Virtual Reality*, 11(4):275–285, 2007.

[20] A. Aristidou, E. Stavrakis, P. Charalambous, Y. Chrysanthou, and S. L. Himona. Folk dance evaluation using laban movement analysis. *J. Comput. Cult. Herit.*, 8(4):20:1–20:19, Aug. 2015.

[21] S. Aromaa, I. Aaltonen, E. Kaasinen, J. Elo, and I. Parkkinen. Use of wearable and augmented reality technologies in industrial maintenance work. In *Proceedings of the 20th International Academic Mindtrek Conference*, AcademicMindtrek '16, pages 235–242, New York, NY, USA, 2016. ACM.

[22] J.-F. Auger. *Sociology*, 42(5):1032–1034, 2008.

[23] A. S. b. Azizo, F. b. Mohamed, C. V. Siang, and M. I. M. Isham. Virtual reality 360 utm campus tour with voice commands. In *2020 6th International Conference on Interactive Digital Media (ICIDM)*, pages 1–6, 2020.

[24] M. Azmandian, M. Hancock, H. Benko, E. Ofek, and A. D. Wilson. Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1968–1979, New York, NY, USA, 2016. ACM.

[25] P. R. K. B, P. Oza, and U. Lahiri. Gaze-sensitive virtual reality based social communication platform for individuals with autism. *IEEE Transactions on Affective Computing*, pages 1–1, 2018.

[26] O. Bamasak, R. Braik, H. Al-Tayari, S. Al-Harbi, G. Al-Semairi, and M. Abu-Hnaidi. Improving autistic children's social skills using virtual reality. In A. Marcus, editor, *Design, User Experience, and Usability. Health, Learning, Playing, Cultural, and Cross-Cultural User Experience*, pages 342–351, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[27] C. Bao and B. Wang. A open source based general framework for virtual surgery simulation. In *2008 International Conference on BioMedical Engineering and Informatics*, volume 1, pages 575–579, 2008.

[28] J. Bardzell, T. Pace, and J. Terrell. Virtual fashion and avatar design: A survey of consumers and designers. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 599–602, New York, NY, USA, 2010. ACM.

[29] M. Barrett and J. Blackledge. Development and evaluation of a desktop vr system for electrical services engineers. In *Lecture Notes in Engineering and Computer Science*, volume 2 LNECS, pages 1102–1107, 2013.

[30] E. Bastug, M. Bennis, M. Médard, and M. Debbah. Toward interconnected virtual reality: Opportunities, challenges, and enablers. *IEEE Communications Magazine*, 55(6):110–117, 2017.

[31] J. W. Bay. Turning video gamers into software developers. *IEEE Computer*, 47(10):99–101, 2014.

[32] L. A. Belfore, P. V. Krishnan, and E. Baydogan. Common scene definition framework for constructing virtual worlds. In *Proceedings of the 37th Conference on Winter Simulation*, WSC '05, page 1985–1992. Winter Simulation Conference, 2005.

[33] B. Bell, S. Feiner, and T. Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pages 101–110, New York, NY, USA, 2001. ACM.

[34] F. Bellalouna. Virtual-reality-based approach for cognitive design-review and fmea in the industrial and manufacturing engineering. In *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 41–46, 2019.

[35] J. Bertrand, D. Brickler, S. Babu, K. Madathil, M. Zelaya, T. Wang, J. Wagner, A. Gramopadhye, and J. Luo. The role of dimensional symmetry on bimanual psychomotor skills education in immersive virtual environments. In *2015 IEEE Virtual Reality (VR)*, pages 3–10, March 2015.

[36] A. Bhimani and P. Spoletini. Empowering requirements elicitation for populations with special needs by using virtual reality. In *Proceedings of the SouthEast Conference*, ACM SE '17, pages 268–270, New York, NY, USA, 2017. ACM.

[37] Bin Zhang and Ye-sho Chen. Enhancing uml conceptual modeling through the use of virtual reality. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, pages 11b–11b, Jan 2005.

[38] A. D. Blaga, M. Frutos-Pascual, C. Creed, and I. Williams. Too hot to handle: An evaluation of the effect of thermal visual representation on user grasping interaction in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–16, New York, NY, USA, 2020. Association for Computing Machinery.

[39] A. Boukerche, D. Duarte, R. Araujo, L. Andrade, and A. Zarrad. A novel solution for the development of collaborative virtual environment simulations in large scale. In *Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 86–94, Oct 2005.

[40] D. R. S. Boyd, J. R. Gallop, K. E. V. Palmen, R. T. Platon, and C. D. Seelig. Vivre: User-centred visualization. In P. Sloot, M. Bubak, A. Hoekstra, and B. Hertzberger, editors, *High-Performance Computing and Networking*, pages 807–816, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[41] E. Bozgeyikli, A. Raij, S. Katkoori, and R. Dubey. Locomotion in virtual reality for individuals with autism spectrum disorder. In *Proceedings of the 2016 Symposium on Spatial User Interaction*, SUI '16, pages 33–42, New York, NY, USA, 2016. ACM.

[42] M. Brennesholtz. *VR/AR Standards – Are We Confused Yet?* 2017.

[43] J. Broekens, M. Harbers, W.-P. Brinkman, C. Jonker, K. Van den Bosch, and J.-J. Meyer. Validity of a virtual negotiation training. In H. H. Vilhjálmsson, S. Kopp, S. Marsella, and K. R. Thórisson, editors, *Intelligent Virtual Agents*, pages 435–436, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[44] C. Buckl, I. Gaponova, M. Geisinger, A. Knoll, and E. A. Lee. Model-based specification of timing requirements. In *Proceedings of the Tenth ACM International Conference on Embedded Software*, EMSOFT '10, page 239–248, New York, NY, USA, 2010. Association for Computing Machinery.

[45] N. Bugalia, S. Kumar, P. Kalra, and S. Choudhary. Mixed reality based interaction system for digital heritage. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, VRCAI '16, page 31–37, New York, NY, USA, 2016. Association for Computing Machinery.

[46] G. Burnett. Information exchange in virtual communities: a typology. *Information research*, 5(4), 2000.

[47] J. Buur and A. Soendergaard. Video card game: An augmented environment for user centred design discussions. In *Proceedings of DARE 2000 on Designing Augmented Reality Environments*, DARE '00, pages 63–69, New York, NY, USA, 2000. ACM.

[48] F. M. Caputo. Gestural interaction in virtual environments: User studies and applications. In *CEUR Workshop Proceedings*, volume 1910, pages 78–89, 2017.

[49] C. Carvalheiro, R. Nóbrega, H. da Silva, and R. Rodrigues. User redirection and direct haptics in virtual environments. In *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, pages 1146–1155, New York, NY, USA, 2016. ACM.

[50] R. Chabra, A. Ilie, N. Rewkowski, Y. Cha, and H. Fuchs. Optimizing placement of commodity depth cameras for known 3d dynamic scene capture. In *2017 IEEE Virtual Reality (VR)*, pages 157–166, March 2017.

[51] S. Chen, Y. Zhang, Y. Li, Z. Chen, and Z. Wang. Spherical structural similarity index for objective omnidirectional video quality assessment. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2018.

[52] W. Chen. Design and analysis of an intelligent system for building virtual roaming. In *2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, pages 26–29, Aug 2018.

[53] Y.-T. Chen, C.-H. Hsu, C.-H. Chung, Y.-S. Wang, and S. V. Babu. ivrnote: Design, creation and evaluation of an interactive note-taking interface for study and reflection in vr learning environments. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 172–180, 2019.

[54] S. Chimalakonda and K. V. Nori. On the nature of roles in software engineering. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE 2014, pages 91–94, New York, NY, USA, 2014. ACM.

[55] O. H. Chimeremeze, S. Sulaiman, D. R. A. Rambli, and O. Foong. Multimodal interactions in traditional foot reflexology. In *2014 International Conference on Computer and Information Sciences (ICCOINS)*, pages 1–6, June 2014.

[56] M. Colley, M. Walch, J. Gugenheimer, A. Askari, and E. Rukzio. Towards inclusive external communication of autonomous vehicles for pedestrians with vision impairments. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[57] C. Colombo, N. D. Blas, I. Gkolias, P. L. Lanzi, D. Loiacono, and E. Stella. An educational experience to raise awareness about space debris. *IEEE Access*, 8:85162–85178, 2020.

[58] C. G. Corrêa, F. de Lourdes dos Santos Nunes, and R. Tori. Virtual reality-based system for training in dental anesthesia. In R. Shumaker and S. Lackey, editors, *Virtual, Augmented and Mixed Reality. Applications of Virtual and Augmented Reality*, pages 267–276, Cham, 2014. Springer International Publishing.

[59] D. S. Cruzes and T. Dyba. Recommended steps for thematic synthesis in software engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 275–284, 2011.

[60] A. C. C. d. Santos, M. E. Delamaro, and F. L. S. Nunes. The relationship between requirements engineering and virtual reality systems: A systematic literature review. In *2013 XV Symposium on Virtual and Augmented Reality*, pages 53–62, May 2013.

[61] C. C. da Silva, B. G. M. de Alcantara, J. C. S. Olimpio, C. M. G. de Gusmao, A. G. da Silva Filho, and W. P. dos Santos. Ilera-aiye: A virtual world for the development of serious games for health education and promotion in the northeastern brazilian countryside. In *2014 IEEE 3nd International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8, May 2014.

[62] L. Dammacco, R. Carli, V. Lazazzera, M. Fiorentino, and M. Dotoli. Designing complex manufacturing systems by virtual reality: A novel approach and its application to the virtual commissioning of a production line. *Computers in Industry*, 143:103761, 2022.

[63] F. Danieau, P. Guillotel, O. Dumas, T. Lopez, B. Leroy, and N. Mollet. Hfx studio: Haptic editor for full-body immersive experiences. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, VRST '18, pages 37:1–37:9, New York, NY, USA, 2018. ACM.

[64] Darryl and Y. Bandung. Designing evaluation system for virtual class service in limited network capacity using servqual methodology. In *2012 International Conference on Cloud Computing and Social Networking (ICCCSN)*, pages 1–5, April 2012.

[65] J. A. Deutsch, J. A. Lewis, E. Whitworth, R. Boian, G. Burdea, and M. Tremaine. Formative evaluation and preliminary findings of a virtual reality telerehabilitation system for the lower extremity. *Presence*, 14(2):198–213, April 2005.

[66] R. Diniz, P. G. Freitas, and M. C. Q. Farias. Towards a point cloud quality assessment model using local binary patterns. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2020.

[67] J. Du, Z. Zou, Y. Shi, and D. Zhao. Simultaneous data exchange between bim and vr for collaborative decision making. In *Congress on Computing in Civil Engineering, Proceedings*, volume 2017-June, pages 1–8, 2017. Cited By :1.

[68] J. Du, Z. Zou, Y. Shi, and D. Zhao. Zero latency: Real-time synchronization of bim data in virtual reality for collaborative decision-making. *Automation in Construction*, 85:51–64, 2018. Cited By :5.

[69] H. Duan, G. Zhai, X. Min, Y. Zhu, Y. Fang, and X. Yang. Perceptual quality assessment of omnidirectional images. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2018.

[70] E. Ebrahimi, A. Robb, L. S. Hartman, C. C. Pagano, and S. V. Babu. Effects of anthropomorphic fidelity of self-avatars on reach boundary estimation in immersive virtual environments. In *Proceedings of the 15th ACM Symposium on Applied Perception*, SAP '18, pages 2:1–2:8, New York, NY, USA, 2018. ACM.

[71] D. Economou, W. L. Mitchell, and T. Boyle. Requirements elicitation for virtual actors in collaborative learning environments. *Computers Education*, 34(3):225 – 239, 2000.

[72] D. Ehmann and C. Wittenberg. The idea of virtual teach-in in the field of industrial robotics. In *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, pages 680–685, 2018.

[73] M. El-Yamri, A. Romero-Hernandez, M. Gonzalez-Riojo, and B. Manero. Comunicarte: A public speaking trainer in virtual reality. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI EA '19, pages VS05:1–VS05:2, New York, NY, USA, 2019. ACM.

[74] S. R. Ellis. Development and history of head-mounted displays and viewers for virtual environments or augmented reality. 2014.

[75] S. Estupiñán, F. Rebelo, P. Noriega, C. Ferreira, and E. Duarte. Can virtual reality increase emotional responses (arousal and valence)? a pilot study. In A. Marcus, editor, *Design, User Experience, and Usability. User Experience Design for Diverse Interaction Platforms and Environments*, pages 541–549, Cham, 2014. Springer International Publishing.

[76] K. et al. Sample srs documentation, requirement completeness checklist, submission guidelines and evaluation rubric - https://github.com/vranonymous/srscomped. August 2022.

[77] A. C. F. Mattioli, D. Caetano and E. Lamounier. On the agile development of virtual reality systems. *Int'l Conf. Software Eng. Research and Practice*, pages 10–16, 2015.

[78] D. L. Farias, R. F. de Souza, P. A. Nardi, and E. F. Damasceno. A motor rehabilitation's motion range assessment with low-cost virtual reality serious game. In *2020 IEEE International Conference on E-health Networking, Application Services (HEALTHCOM)*, pages 1–5, 2021.

[79] S. Faroque, B. Horan, M. Mortimer, and M. Pangestu. Large-scale virtual reality micro-robotic cell injection training. In *World Automation Congress Proceedings*, volume 2016-October, 2016.

[80] S. Faroque, M. Mortimer, M. Pangestu, M. Seyedmahmoudian, and B. Horan. Evaluation of a new virtual reality micro-robotic cell injection training system. *Computers Electrical Engineering*, 67:656–671, 2018.

[81] J. Feng, K. Chen, C. Zhang, and H. Li. A virtual reality-based training system for ankle rehabilitation. In *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pages 255–259, 2018.

[82] P. Figueroa, M. Coral, P. Boulanger, J. Borda, E. Londoño, F. Vega, F. Prieto, and D. Restrepo. Multi-modal exploration of small artifacts: An exhibition at the gold museum in bogota. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, pages 67–74, New York, NY, USA, 2009. ACM.

[83] T. Finseth and C. C. Anderson. Re-engineering student thought processes using spaceflight operations. In *AIAA Space and Astronautics Forum and Exposition, SPACE 2016*, 2016.

[84] I. O. for Standardization and I. E. Commission. *Software Engineering–Product Quality: Quality model*, volume 1. ISO/IEC, 2001.

[85] J. Françoise and F. Bevilacqua. Motion-sound mapping through interaction: An approach to user-centered design of auditory feedback using machine learning. *ACM Trans. Interact. Intell. Syst.*, 8(2):16:1–16:30, June 2018.

[86] J. P. Freiwald, O. Ariza, O. Janeh, and F. Steinicke. Walking by cycling: A novel in-place locomotion user interface for seated virtual reality experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[87] J. P. Freiwald, L. Diedrichsen, A. Baur, O. Manka, P. B. Jorshery, and F. Steinicke. Conveying perspective in multi-user virtual reality collaborations. In *Proceedings of the Conference on Mensch Und Computer*, MuC '20, page 137–144, New York, NY, USA, 2020. Association for Computing Machinery.

[88] J. P. Freiwald, Y. Göbel, F. Mostajeran, and F. Steinicke. The cybersickness susceptibility questionnaire: Predicting virtual reality tolerance. In *Proceedings of the Conference on Mensch Und Computer*, MuC '20, page 115–118, New York, NY, USA, 2020. Association for Computing Machinery.

[89] A. Geiger, I. Bewersdorf, E. Brandenburg, and R. Stark. *Visual feedback for grasping in virtual reality environments for an interface to instruct digital human models*, volume 607 of *Advances in Intelligent Systems and Computing*. 2018. Cited By :1.

[90] C. George, P. Tamunjoh, and H. Hussmann. Invisible boundaries for vr: Auditory and haptic signals as indicators for real world boundaries. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2020.

[91] M. L. Gernhardt, K. H. Beaton, S. P. Chappell, O. S. Bekdash, and A. F. J. Abercromby. Development of a ground test analysis protocol for nasa's nextstep phase 2 habitation concepts. In *2018 IEEE Aerospace Conference*, pages 1–27, March 2018.

[92] V. Gervasi and B. Nuseibeh. Lightweight validation of natural language requirements: a case study. In *Proceedings Fourth International Conference on Requirements Engineering. ICRE 2000. (Cat. No.98TB100219)*, pages 140–148, 2000.

[93] N. Ghrairi, S. Kpodjedo, A. Barrak, F. Petrillo, and F. Khomh. The state of practice on virtual reality (vr) applications: An exploratory study on github and stack overflow. In *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pages 356–366, July 2018.

[94] D. R. Glowacki, M. D. Wonnacott, R. Freire, B. R. Glowacki, E. M. Gale, J. E. Pike, T. de Haan, M. Chatziapostolou, and O. Metatla. Isness: Using multi-person vr to design peak mystical type experiences comparable to psychedelics. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–14, New York, NY, USA, 2020. Association for Computing Machinery.

[95] S. Göbel, C. Geiger, C. Heinze, and D. Marinos. Creating a virtual archery experience. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, page 337–340, New York, NY, USA, 2010. Association for Computing Machinery.

[96] M. Green. Human machine interaction research at the oecd halden reactor project. In *1999 International Conference on Human Interfaces in Control Rooms, Cockpits and Command Centres*, pages 357–363, 1999.

[97] K. Group. *The OpenXR Specification 1.0.24*. 2019.

[98] K. Gu, V. Jakhetiya, J. Qiao, X. Li, W. Lin, and D. Thalmann. Model-based referenceless quality metric of 3d synthesized images using local image description. *IEEE Transactions on Image Processing*, 27(1):394–405, Jan 2018.

[99] N. Gu and K. London. Understanding and facilitating bim adoption in the aec industry. *Automation in Construction*, 19(8):988 – 999, 2010. The role of VR and BIM to manage the construction and design processes.

[100] U. Gulec, M. Yilmaz, V. Isler, R. V. O'Connor, and P. Clarke. Adopting virtual reality as a medium for software development process education. In *Proceedings of the 2018 International Conference on Software and System Process*, ICSSP '18, pages 71–75, New York, NY, USA, 2018. ACM.

[101] S. Gunkel, H. Stokking, M. Prins, O. Niamut, E. Siahaan, and P. Cesar. Experiencing virtual reality together: Social vr use case study. In *Proceedings of the 2018 ACM International Conference on Interactive Experiences for TV and Online Video*, TVX '18, pages 233–238, New York, NY, USA, 2018. ACM.

[102] H. Guo, R. Brown, and R. Rasmussen. A theoretical basis for using virtual worlds as a personalised process visualisation approach. In X. Franch and P. Soffer, editors, *Advanced Information Systems Engineering Workshops*, pages 229–240, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[103] J. Guo, D. Weng, H. Fang, Z. Zhang, J. Ping, Y. Liu, and Y. Wang. Exploring the differences of visual discomfort caused by long-term immersion between virtual environments and physical environments. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 443–452, 2020.

[104] V. Gómez, K. Peñaranda, and P. Figueroa. Lessons learned from requirements gathering for virtual reality simulators. *Virtual Reality Intelligent Hardware*, 3(5):407–422, 2021.

[105] A. Hamam, A. E. Saddik, and J. Alja'am. A quality of experience model for haptic virtual environments. *ACM Trans. Multimedia Comput. Commun. Appl.*, 10(3):28:1–28:23, Apr. 2014.

[106] J. Hamid, N. H. Ubaidullah, K. A. Samsudin, and A. Saad. Designing a web based non-immersive virtual environment application (wb-nivea) for diagnosing dyslexic children' potential. In *2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC)*, pages 1–6, 2015.

[107] B. Han and J. Xie. Thesis: Practical experience: Adopt agile methodology combined with kanban for virtual reality development. *Dept of CS, University of Gothenburg Publications*, pages 1–16, 2011.

[108] R. R. Hariadi and I. Kuswardayan. Design and implementation of virtual indonesian musical instrument (vimi) application using leap motion controller. In *2016 International Conference on Information Communication Technology and Systems (ICTS)*, pages 43–48, Oct 2016.

[109] L. Harley, S. Robertson, M. Gandy, S. Harbert, and D. Britton. The design of an interactive stroke rehabilitation gaming system. In J. A. Jacko, editor, *Human-Computer Interaction. Users and Applications*, pages 167–173, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[110] P. Harms. Automated usability evaluation of virtual reality applications. *ACM Trans. Comput.-Hum. Interact.*, 26(3):14:1–14:36, Apr. 2019.

[111] L. Hastings and B. E. Riecke. The influence of shading, display size and individual differences on navigation in virtual reality. In *Proceedings of the ACM Symposium on Applied Perception, SAP 2014*, pages 51–58, 2014.

[112] S.-P. He, Z. Qin, and X.-P. He. Artificial intelligence applications in the design of spe virtual reality operation training system. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, volume 4, pages 2324–2328 Vol.4, 2003.

[113] D. Herumurti, A. Yuniarti, P. Rimawan, and A. A. Yunanto. Overcoming glossophobia based on virtual reality and heart rate sensors. In *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 139–144, July 2019.

[114] R. Hoda. Socio-technical grounded theory for software engineering. *IEEE Transactions on Software Engineering*, pages 1–1, 2021.

[115] Y. Hong, H. Zhen, P. Chen, S. Zheng, Y. Du, Z. Chen, and C. Gan. 3d-llm: Injecting the 3d world into large language models, 2023.

[116] C.-F. Hsu, A. Chen, C.-H. Hsu, C.-Y. Huang, C.-L. Lei, and K.-T. Chen. Is foveated rendering perceivable in virtual reality?: Exploring the efficiency and consistency of quality assessment methods. In *Proceedings of the 25th ACM International Conference on Multimedia*, MM ’17, pages 55–63, New York, NY, USA, 2017. ACM.

[117] C. . Huang, H. . Yien, Y. . Chen, Y. . Su, and Y. . Lin. Developing a bim-based visualization and interactive system for healthcare design. In *ISARC 2017 - Proceedings of the 34th International Symposium on Automation and Robotics in Construction*, pages 372–379, 2017.

[118] M.-H. Huang and S.-Y. Tsau. A flow experience analysis on the virtual reality artwork: La camera insabbiata. In *Proceedings of the International Conference on Machine Vision and Applications*, ICMVA 2018, pages 51–55, New York, NY, USA, 2018. ACM.

[119] G. Inc. *COLLADA - Digital Asset and FX Exchange Schema.* 2010.

[120] S. C. E. Inc. *COLLADA - Digital Asset and FX Exchange Schema.* Khronos Group, 2004.

[121] ISO. ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. Technical report, International Organization for Standardization, 1998.

[122] L. Itzel, R. Suselbeck, G. Schiele, and C. Becker. Specifying consistency requirements for massively multi-user virtual environments. In *2011 IEEE International Workshop on Haptic Audio Visual Environments and Games*, pages 1–2, Oct 2011.

[123] G. Ivan, C. Pacheco, and M. Reyes. Requirements elicitation techniques: A systematic literature review based on the maturity of the techniques. *IET Software*, 12:365–378, 04 2018.

[124] A. Jaramillo Franco. Requirements elicitation approaches: A systematic review. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pages 520–521, May 2015.

[125] H. Järvinen, U. Bernardet, and P. F. Verschure. Interaction mapping affects spatial memory and the sense of presence when navigating in a virtual environment. In *Proceedings of the Fifth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '11, pages 321–324, New York, NY, USA, 2011. ACM.

[126] S. Jayaram, H. I. Connacher, and K. W. Lyons. Virtual assembly using virtual reality techniques. *Computer-Aided Design*, 29(8):575–584, 1997.

[127] Ji-yi Xu, Bin Xie, and Jin-lin Zhai. The implementation method and improvement of media piug-in in secondlife 3d platform. In *Proceedings of 2011 International Conference on Computer Science and Network Technology*, volume 3, pages 1669–1673, Dec 2011.

[128] G. Jin and S. Nakayama. Virtual reality game for safety education. In *2014 International Conference on Audio, Language and Image Processing*, pages 95–100, July 2014.

[129] P. Jindal, V. Khemchandani, S. Chandra, and V. Pandey. A multiplayer shooting game based simulation for defence training. In *2021 International Conference on Computational Performance Evaluation (ComPE)*, pages 592–597, 2021.

[130] B. John, P. Raiturkar, A. Banerjee, and E. Jain. An evaluation of pupillary light response models for 2d screens and vr hmds. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, VRST '18, pages 19:1–19:11, New York, NY, USA, 2018. ACM.

[131] J. A. Jones, J. E. Swan, II, G. Singh, E. Kolstad, and S. R. Ellis. The effects of virtual reality, augmented reality, and motion parallax on egocentric depth perception. In *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization*, APGV '08, pages 9–14, New York, NY, USA, 2008. ACM.

[132] A. Juhana, R. Yusuf, A. S. Prihatmanto, and A. G. Abdullah. Basic electrical installation trainer boards : Virtual reality based laboratory for electrical basic education. In *2020 6th International Conference on Interactive Digital Media (ICIDM)*, pages 1–6, 2020.

[133] K. S. Kabir, A. Alsaleem, and J. Wiese. The impact of spinal cord injury on participation in human-centered research. In *Designing Interactive Systems Conference 2021*, DIS '21, page 1902–1914, New York, NY, USA, 2021. Association for Computing Machinery.

[134] K. Kandhari. Vr bowling alley game, 2023.

[135] S. A. Karre. Vreqst - online documentation for virtual reality requirement analysts, 2023.

[136] S. A. Karre. *VReqST Sample Specifications, https://github.com/sai11101989/VReqST/tree/main/SampleSpec*. 2023.

[137] S. A. Karre, N. Mathur, 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 (Formerly Known As India Software Engineering Conference)*, ISEC'19, pages 3:1–3:11, New York, NY, USA, 2019. ACM.

[138] S. A. Karre, N. Mathur, 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 (formerly known as India Software Engineering Conference), ISEC 2019, Pune, India, February 14-16, 2019*, pages 3:1–3:11, 2019.

[139] S. A. Karre, N. Mathur, 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 (formerly known as India Software Engineering Conference), ISEC 2019, Pune, India, February 14-16, 2019*, pages 3:1–3:11. ACM, 2019.

[140] S. A. Karre, V. Pareek, R. Mittal, and Y. R. Reddy. A role based model template for specifying virtual reality software. In *In proceedings International Workshop on Virtual and Augmented Reality Software Engineering, in conjucture with Automated Software Engineering (ASE 2022)*, Oakland Center, Michigan, 2022. ACM.

[141] S. A. Karre, K. Vaidhyanathan, and Y. R. Reddy. A tool based experiment to teach elicitation and specification of virtual reality product requirements. In V. Choppella, D. B. Phatak, A. Luxton-Reilly, and M. Craig, editors, *Proceedings of the ACM Conference on Global Computing Education Vol 2, CompEd 2023, Hyderabad, India, December 5-9, 2023*, page 195. ACM, 2023.

[142] M. Keckeisen, S. L. Stoev, M. Feurer, and W. Strasser. Interactive cloth simulation in virtual environments. In *Proceedings - IEEE Virtual Reality*, volume 2003-January, pages 71–78, 2003.

[143] C. Keighrey, R. Flynn, S. Murray, S. Brennan, and N. Murray. Comparing user qoe via physiological and interaction measurements of immersive ar and vr speech and language therapy applications. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, Thematic Workshops '17, pages 485–492, New York, NY, USA, 2017. ACM.

[144] C. Keighrey, R. Flynn, S. Murray, and N. Murray. A physiology-based qoe comparison of interactive augmented reality, virtual reality and tablet-based applications. *IEEE Transactions on Multimedia*, pages 1–1, 2020.

[145] D. Kim, R. B. France, S. Ghosh, and E. Song. 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*, page 452. IEEE Computer Society, 2003.

[146] H. G. Kim, W. J. Baddar, H.-t. Lim, H. Jeong, and Y. M. Ro. Measurement of exceptional motion in vr video contents for vr sickness assessment using deep convolutional autoencoder. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, VRST '17, pages 36:1–36:7, New York, NY, USA, 2017. ACM.

[147] S. Kim, G. C. Park, and S. Kim. Usability evaluation of humanoid-animation avatar with physiological signals. In *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, pages 628–636, Oct 2007.

[148] T. Kim. Theoretical analysis of the physiologic mechanism for visual comfort in 3d virtual reality. In *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pages 302–305, Jan 2017.

[149] K. Kirchbach and C. Runde. Augmented reality for construction control. In *2012 16th International Conference on Information Visualisation*, pages 440–444, July 2012.

[150] B. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. G. Linkman. Systematic literature reviews in software engineering - A systematic literature review. *Information & Software Technology*, 51(1):7–15, 2009.

[151] R. Kneuper. Sixty years of software development life cycle models. *IEEE Annals of the History of Computing*, 39(3):41–54, 2017.

[152] A. Krekhov, K. Emmerich, P. Bergmann, S. Cmentowski, and J. Krüger. Self-transforming controllers for virtual reality first person shooters. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '17, page 517–529, New York, NY, USA, 2017. Association for Computing Machinery.

[153] B. Krogfoss, J. Duran, P. Perez, and J. Bouwen. Quantifying the value of 5g and edge cloud on qoe for ar/vr. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–4, 2020.

[154] K. Lano, S. Kolahdouz-Rahimi, and S. Fang. Model transformation development using automated requirements analysis, metamodel matching, and transformation by example. *ACM Trans. Softw. Eng. Methodol.*, 31(2), nov 2021.

[155] S. LaValle. Virtual reality. 2016.

[156] S. M. LaValle. *Virtual Reality*. Cambridge University Press, 2020.

[157] Y. Lee, J. Choi, S. Kim, S. Lee, and S. Jang. Social augmented reality for sensor visualization in ubiquitous virtual reality. In R. Shumaker, editor, *Virtual and Mixed Reality - New Trends*, pages 69–75, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[158] Y. S. Lee and B. . Sohn. Immersive gesture interfaces for navigation of 3d maps in hmd-based mobile virtual environments. *Mobile Information Systems*, 2018, 2018.

[159] J. R. Levy and H. Bjelland. *Create Your Own Virtual Reality System*. McGraw-Hill, Inc., USA, 1994.

[160] S. Li, Y. Chui, and P. Heng. Simulation of thermal damage to bone tissue during bone drilling. In *2014 4th IEEE International Conference on Information Science and Technology*, pages 569–573, April 2014.

[161] Y. Li, T. Yue, S. Ali, and L. Zhang. Enabling automated requirements reuse and configuration. In *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A*, SPLC '19, page 206, New York, NY, USA, 2019. Association for Computing Machinery.

[162] M. Lieze, V. C. Jelle, D. Benedicte, V. de Weghe Nico, M. Mario, and D. Van Dyck. Using virtual reality to investigate physical environmental factors related to cycling in older adults: A comparison between two methodologies. *Journal of Transport Health*, 19:100921, 2020.

[163] X. Liu, Y. Zhu, H. Huo, P. Wei, L. Wang, A. Sun, C. Hu, X. Yin, Z. Lv, and Y. Fan. Design of virtual guiding tasks with haptic feedback for assessing the wrist motor function of patients with upper motor neuron lesions. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(5):984–994, May 2019.

[164] P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. 01 1995.

[165] J. Lucas and W. Thabet. Implementation and evaluation of a vr task-based training tool for conveyor belt safety training. *Electronic Journal of Information Technology in Construction*, 13:637–659, 2008.

[166] J.-L. Lugrin, M. Cavazza, F. Charles, M. Le Renard, J. Freeman, and J. Lessiter. Immersive fps games: User experience and performance. In *Proceedings of the 2013 ACM International Workshop on Immersive Media Experiences*, ImmersiveMe '13, pages 7–12, New York, NY, USA, 2013. ACM.

[167] D.-N. Ly, T.-T. La, K.-D. Le, C. Nguyen, M. Fjeld, T. N.-D. Tran, and M.-T. Tran. 360tourguiding: Towards virtual reality training for tour guiding. In *Adjunct Publication of the 24th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[168] O. López Chávez, L.-F. Rodríguez, and J. O. Gutierrez-Garcia. A comparative case study of 2d, 3d and immersive-virtual-reality applications for healthcare education. *International Journal of Medical Informatics*, 141:104226, 2020.

[169] G. L. M.-L. Champel, R. Koenen and M. Budagavi. Working draft 0.4 of tr: Technical report on architectures for immersive media., 2017.

[170] M. Ma, R. Proffitt, and M. Skubic. Quantitative assessment and validation of a stroke rehabilitation game. In *Proceedings of the Second IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies*, CHASE '17, pages 255–257, Piscataway, NJ, USA, 2017. IEEE Press.

[171] T. Manninen. Contextual virtual interaction as part of ubiquitous game design and development. *Personal Ubiquitous Comput.*, 6(5–6):390–406, jan 2002.

[172] J. Mapar, K. Brown, J. Medina, K. Laskey, and C. Conaty. Nasa goddard space flight center virtual system design environment. In *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, volume 7, pages 7–3580 vol.7, March 2001.

[173] C. Marañes, D. Gutierrez, and A. Serrano. Exploring the impact of 360° movie cuts in users' attention. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 73–82, 2020.

[174] R. M. Marra, J. L. Moore, and A. K. Klimczak. Content analysis of online discussion forums: A comparative analysis of protocols. *Educational Technology Research and Development*, 52(2):23, Jun 2004.

[175] D. Martin, S. Malpica, D. Gutierrez, B. Masia, and A. Serrano. Multimodality in vr: A survey. *ACM Comput. Surv.*, 54(10s), sep 2022.

[176] N. Mathur and S. A. Karre. *VR Empirical Study Supplementary Resources, URL: https://goo.gl/iY7JuV*. [Last Accessed]:07-15-2018, 2018.

[177] C. Mégard, F. Gosselin, S. Bouchigny, F. Ferlay, and F. Taha. User-centered design of a maxillo-facial surgery training platform. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, pages 265–266, New York, NY, USA, 2009. ACM.

[178] 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*, ISEC 2022, New York, NY, USA, 2022. Association for Computing Machinery.

[179] M. Moehring and B. Froehlich. Pseudo-physical interaction with a virtual car interior in immersive environments. In *Proceedings of the 11th Eurographics Conference on Virtual Environments*, EGVE'05, pages 181–189, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[180] N. D. Mohd Muhaiyuddin and D. R. Awang Rambli. Navigation in image-based virtual reality as the factor to elicit spatial presence experience. In *2014 International Symposium on Technology Management and Emerging Technologies*, pages 349–354, May 2014.

[181] Y. R. R. Mohit Kuri, Sai Anirudh Karre. Slr suppliment data. Last Accessed - Sept 2020.

[182] A. N. Moraes, R. Flynn, A. Hines, and N. Murray. Evaluating the user in a sound localisation task in a virtual reality application. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2020.

[183] J. Morse. *Critical Issues in Qualitative Research Methods*. SAGE Publications, 1994.

[184] S. E. Motlagh Tehrani, N. M. M. Zainuddin, and T. Takavar. Heuristic evaluation for virtual museum on smartphone. In *2014 3rd International Conference on User Science and Engineering (i-USEr)*, pages 227–231, Sep. 2014.

[185] M. A. Muhanna. Virtual reality and the cave: Taxonomy, interaction challenges and research directions. *Journal of King Saud University - Computer and Information Sciences*, 27(3):344–361, 2015.

[186] E. Murphy-Hill, T. Zimmermann, and N. Nagappan. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 1–11, New York, NY, USA, 2014. ACM.

[187] G. Narasimham, H. Adams, J. Rieser, and B. Bodenheimer. Encoding height: Egocentric spatial memory of adults and teens in a virtual stairwell. In *ACM Symposium on Applied Perception 2020*, SAP '20, New York, NY, USA, 2020. Association for Computing Machinery.

[188] M. Narbutt, S. O'Leary, A. Allen, J. Skoglund, and A. Hines. Streaming vr for immersion: Quality aspects of compressed spatial audio. In *2017 23rd International Conference on Virtual System Multimedia (VSMM)*, pages 1–6, Oct 2017.

[189] A. Nikitin, N. Reshetnikova, I. Sitnikov, and O. Karelova. Vr training for railway wagons maintenance: architecture and implementation. *Procedia Computer Science*, 176:622 – 631, 2020. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020.

[190] M. aaNebeling 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)*, pages 333–337, 2018.

[191] R. abHorst, R. Naraghi-Taghi-Off, L. Rau, and R. Doerner. Authoring with virtual reality nuggets—lessons learned. *Frontiers in Virtual Reality*, 3:17, 2022.

[192] T. Nukarinen, J. Kangas, J. Rantala, T. Pakkanen, and R. Raisamo. Hands-free vibrotactile feedback for object selection tasks in virtual reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, VRST '18, pages 94:1–94:2, New York, NY, USA, 2018. ACM.

[193] F. Nusrat, F. Hassan, H. Zhong, and X. Wang. How developers optimize virtual reality applications: A study of optimization commits in open source unity projects. In *Proceedings of the 43rd International Conference on Software Engineering*, ICSE '21, page 473–485. IEEE Press, 2021.

[194] H. C. Okere, S. Sulaiman, D. R. A. Rambli, and O.-M. Foong. A multimodal interaction design guidelines for vr foot reflexology therapy application. *Int. J. Operat. Res. Inf. Syst.*, 7(3):74–91, jul 2016.

[195] D. M. Oliveira, S. C. Cao, X. F. Hermida, and F. M. Rodriguez. Virtual reality system for industrial training. In *2007 IEEE International Symposium on Industrial Electronics*, pages 1715–1720, June 2007.

[196] Y. S. Pai, B. Outram, N. Vontin, and K. Kunze. Transparent reality: Using eye gaze focus depth as interaction modality. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16 Adjunct, pages 171–172, New York, NY, USA, 2016. ACM.

[197] I. Paliokas, A. Tsakiris, A. Vidalis, and D. Tzovaras. Sense of presence and metacognition enhancement in virtual reality exposure therapy in the treatment of social phobias and the fear of flying. In R. Shumaker and S. Lackey, editors, *Virtual, Augmented and Mixed Reality. Applications of Virtual and Augmented Reality*, pages 316–328, Cham, 2014. Springer International Publishing.

[198] V. Paneva, M. Bachynskyi, and J. Müller. Levitation simulator: Prototyping ultrasonic levitation interfaces in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[199] R. Paris, M. Joshi, Q. He, G. Narasimham, T. P. McNamara, and B. Bodenheimer. Acquisition of survey knowledge using walking in place and resetting methods in immersive virtual environments. In *Proceedings of the ACM Symposium on Applied Perception*, SAP '17, pages 7:1–7:8, New York, NY, USA, 2017. ACM.

[200] L. Pasquale, P. Spoletini, D. Pometto, F. Blasi, and T. Redaelli. Requirements engineering meets physiotherapy: An experience with motion-based games. In J. Doerr and A. L. Opdahl, editors, *Requirements Engineering: Foundation for Software Quality*, pages 315–330, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[201] T. B. Pence, L. C. Dukes, L. F. Hodges, N. K. Meehan, and A. Johnson. An eye tracking evaluation of a virtual pediatric patient training system for nurses. In T. Bickmore, S. Marsella, and C. Sidner, editors, *Intelligent Virtual Agents*, pages 329–338, Cham, 2014. Springer International Publishing.

[202] X. Peng, J. Huang, A. Denisova, H. Chen, F. Tian, and H. Wang. A palette of deepened emotions: Exploring emotional challenge in virtual reality games. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[203] M. Pérez, S. Casteleyn, I. Sanz, and M. J. Aramburu. Requirements gathering in a model-based approach for the design of multi-similarity systems. In *Proceedings of the First International Workshop on Model Driven Service Engineering and Data Quality and Security*, MoSE+DQS '09, page 45–52, New York, NY, USA, 2009. Association for Computing Machinery.

[204] K. Petersen, S. Vakkalanka, and L. Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1 – 18, 2015.

[205] J. Plouzeau, J. Chardonnet, and F. Merienne. Using cybersickness indicators to adapt navigation in virtual reality: A pre-study. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 661–662, March 2018.

[206] D. Pohlandt, B. Preim, and P. Saalfeld. Supporting anatomy education with a 3d puzzle in a vr environment - results from a pilot study. In *Proceedings of Mensch Und Computer 2019*, MuC'19, page 91–102, New York, NY, USA, 2019. Association for Computing Machinery.

[207] T. Poitschke, M. Ablaßmeier, S. Reifinger, and G. Rigoll. A multifunctional vr-simulator platform for the evaluation of automotive user interfaces. In J. A. Jacko, editor, *Human-Computer Interaction. HCI Applications and Services*, pages 1120–1129, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[208] C. Pontonnier, A. Samani, M. Badawi, P. Madeleine, and G. Dumont. Assessing the ability of a vr-based assembly task simulation to evaluate physicalrisk factors. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):664–674, May 2014.

[209] F. A. Purnomo, M. Purnawati, E. H. Pratisto, and T. N. Hidayat. Archery training simulation based on virtual reality. In *2022 1st International Conference on Smart Technology, Applied Informatics, and Engineering (APICS)*, pages 195–198, 2022.

[210] S. Putze, D. Alexandrovsky, F. Putze, S. Höffner, J. D. Smeddinck, and R. Malaka. Breaking the experience: Effects of questionnaires in vr user studies. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–15, New York, NY, USA, 2020. Association for Computing Machinery.

[211] P. Pyk, D. Wille, E. Chevrier, Y. Hauser, L. Holper, I. Fatton, R. Greipl, S. Schlegel, L. Ottiger, B. Ruckriem, A. Pescatore, A. Meyer-Heim, D. Kiper, and K. Eng. A paediatric interactive therapy system for arm and hand rehabilitation. In *2008 Virtual Rehabilitation*, pages 127–132, Aug 2008.

[212] J. Qiao, M. Liu, S. Li, Z. He, and Z. Yang. Highly efficient quality assessment of 3d-synthesized views based on compression technology. *IEEE Access*, 6:42309–42318, 2018.

[213] F. Rabbi, T. Park, B. Fang, M. Zhang, and Y. Lee. When virtual reality meets internet of things in the gym: Enabling immersive interactive machine exercises. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(2):78:1–78:21, July 2018.

[214] C. Ragkhitwetsagul, M. Choetkiertikul, A. Hoonlor, and M. Prachyabrued. Virtual reality for software engineering presentations. In *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, pages 507–516, 2022.

[215] F. Rahim, M. P. Queluz, and J. Ascenso. Objective assessment of line distortions in viewport rendering of 360º images. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 68–75, Dec 2018.

[216] P. Rajagopal, R. Lee, T. Ahlswede, Chia-Chu Chiang, and D. Karolak. A new approach for software requirements elicitation. In *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network*, pages 32–42, May 2005.

[217] A. Rajan. Automated requirements-based test case generation. *SIGSOFT Softw. Eng. Notes*, 31(6):1–2, nov 2006.

[218] P. Rajeswaran, T. Kesavadas, P. Jani, and P. Kumar. Airwayvr: Virtual reality trainer for endotracheal intubation-design considerations and challenges. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1130–1131, 2019.

[219] R. Ramadan and Y. Widyani. Game development life cycle guidelines. In *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 95–100, Sept 2013.

[220] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[221] B. Randell. Software engineering in 1968. In *Proceedings of the 4th International Conference on Software Engineering*, ICSE '79, pages 1–10, Piscataway, NJ, USA, 1979. IEEE Press.

[222] A. Rankin, J. Field, R. Kovordanyi, M. Morin, J. Jenvald, and H. Eriksson. Training systems design: Bridging the gap between users and developers using storyboards. In *Proceedings of the 29th Annual European Conference on Cognitive Ergonomics*, ECCE '11, pages 205–212, New York, NY, USA, 2011. ACM.

[223] A. Riegler, B. Aksoy, A. Riener, and C. Holzmann. Gaze-based interaction with windshield displays for automated driving: Impact of dwell time and feedback design on task performance and subjective workload. In *12th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '20, page 151–160, New York, NY, USA, 2020. Association for Computing Machinery.

[224] S. Rings, S. Karaosmanoglu, L. Kruse, D. Apken, T. Picker, and F. Steinicke. Using exergames to train patients with dementia to accomplish daily routines. In *Extended Abstracts of the 2020 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '20, page 345–349, New York, NY, USA, 2020. Association for Computing Machinery.

[225] S. Rings, F. Steinicke, T. Picker, and C. Prasuhn. Enabling patients with neurological diseases to perform motor-cognitive exergames under clinical supervision for everyday usage. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 360–364, 2020.

[226] P. Rodrigues, H. Coelho, M. Melo, and M. Bessa. Virtual reality for training: A computer assembly application. In *2022 International Conference on Graphics and Interaction (ICGI)*, pages 1–6, 2022.

[227] K. Rogers, G. Ribeiro, R. R. Wehbe, M. Weber, and L. E. Nacke. Vanishing importance: Studying immersive effects of game audio perception on player experiences in virtual reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 328:1–328:13, New York, NY, USA, 2018. ACM.

[228] R. A. Ruddle, E. Volkova, and H. H. Bülthoff. Learning to walk in virtual reality. *ACM Trans. Appl. Percept.*, 10(2):11:1–11:17, June 2013.

[229] D. P. Saha, L. Thomas Martin, and R. Benjamin Knapp. Towards defining a quality-metric for affective feedback in an intelligent environment. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 609–614, March 2018.

[230] M. N. b. Said, N. b. Senan, M. F. Othman, M. H. A. Wahab, and M. N. Derahman. Virtual kiosk: Taman herba. In *2020 International Conference on Computer Science and Its Application in Agriculture (ICOSICA)*, pages 1–8, 2020.

[231] K. Sakamoto, S. Shirai, J. Orlosky, H. Nagataki, N. Takemura, M. Alizadeh, and M. Ueda. Exploring pupillometry as a method to evaluate reading comprehension in vr-based educational comics. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 422–426, 2020.

[232] D. P. Salgado, F. R. Martins, T. B. Rodrigues, C. Keighrey, R. Flynn, E. L. M. Naves, and N. Murray. A qoe assessment method based on eda, heart rate and eeg of a virtual reality assistive technology system. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, pages 517–520, New York, NY, USA, 2018. ACM.

[233] S. Sanders and A. Rolfe. The use of virtual reality for preparation and implementation of jet remote handling operations. *Fusion Engineering and Design*, 69(1):157 – 161, 2003. 22nd Symposium on Fusion Technology.

[234] F. Sanfilippo, P. B. T. Weustink, and K. Y. Pettersen. A coupling library for the force dimension haptic devices and the 20-sim modelling and simulation environment. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pages 168–173, 2015.

[235] K. Sanjaya, F. Henning, and K. R. Purba. 3d lidar city model application and marketing plan development. In *2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT)*, pages 238–242, Sep. 2017.

[236] M. E. C. Santos, T. Taketomi, C. Sandor, J. Polvi, G. Yamamoto, and H. Kato. A usability scale for handheld augmented reality. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, VRST '14, pages 167–176, New York, NY, USA, 2014. ACM.

[237] P. Savioja, P. Järvinen, T. Karhela, P. Siltanen, and C. Woodward. Developing a mobile, service-based augmented reality tool for modern maintenance work. In R. Shumaker, editor, *Virtual Reality*, pages 554–563, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[238] D. Schouten, N. Smets, M. Driessen, M. Hanekamp, A. H. M. Cremers, and M. A. Neerincx. User requirement analysis of social conventions learning applications for non-natives and low-literates. In D. Harris, editor, *Engineering Psychology and Cognitive Ergonomics. Understanding Human Cognition*, pages 354–363, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[239] H. Schumann, S. Burtescu, and F. Siering. Applying augmented reality techniques in the field of interactive collaborative design. In R. Koch and L. Van Gool, editors, *3D Structure from Multiple Images of Large-Scale Environments*, pages 290–303, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[240] S. Schwarz and M. M. Hannuksela. Perceptual quality assessment of hevc main profile depth map compression for six degrees of freedom virtual reality video. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 181–185, Sep. 2017.

[241] V. Schwind, P. Knierim, N. Haas, and N. Henze. Using presence questionnaires in virtual reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 360:1–360:12, New York, NY, USA, 2019. ACM.

[242] J. Seo and G. J. Kim. Teaching structured development of virtual reality systems using p-vot. In K.-c. Hui, Z. Pan, R. C.-k. Chung, C. C. L. Wang, X. Jin, S. Göbel, and E. C.-L. Li, editors, *Technologies for E-Learning and Digital Entertainment*, pages 69–80, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[243] D. Shao, X. Liu, B. Cheng, O. Wang, and T. Hoang. Edge4real: A cost-effective edge computing based human behaviour recognition system for human-centric software engineering. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, page 1287–1291, New York, NY, USA, 2020. Association for Computing Machinery.

[244] Q. Shen, J. Gausemeier, J. Bauch, and R. Radkowski. A cooperative virtual prototyping system for mechatronic solution elements based assembly. *Adv. Eng. Inform.*, 19(2):169–177, apr 2005.

[245] W. R. Sherman and A. B. Craig. Introduction to virtual reality systems. In W. R. Sherman and A. B. Craig, editors, *Understanding Virtual Reality*, The Morgan Kaufmann Series in Computer Graphics, pages 70–73. Morgan Kaufmann, San Francisco, 2003.

[246] E. Shihab. Practical software quality prediction. In *30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014*, pages 639–644, 2014.

[247] G. Shochat, S. Maoz, A. Stark-Inbar, B. Blumenfeld, D. Rand, S. Preminger, and Y. Sacher. Motion-based virtual reality cognitive training targeting executive functions in acquired brain injury community-dwelling individuals: A feasibility and initial efficacy pilot. In *International Conference on Virtual Rehabilitation, ICVR*, volume 2017-June, 2017.

[248] N. Sidaty, P. Cabarat, W. Hamidouche, D. Menard, and O. Deforges. Performance and computational complexity of the future video coding. In *2018 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 31–36, Oct 2018.

[249] A. Simon. First-person experience and usability of co-located interaction in a projection-based virtual environment. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '05, pages 23–30, New York, NY, USA, 2005. ACM.

[250] T. M. Simon, R. T. Smith, B. Thomas, S. Von Itzstein, M. Smith, J. Park, and J. Park. Merging tangible buttons and spatial augmented reality to support ubiquitous prototype designs. In *Proceedings of the Thirteenth Australasian User Interface Conference - Volume 126*, AUIC '12, pages 29–38, Darlinghurst, Australia, Australia, 2012. Australian Computer Society, Inc.

[251] S. P. Smith and S. Todd. Evaluating a haptic-based virtual environment for venepuncture training. In *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*, VRST '07, pages 223–224, New York, NY, USA, 2007. ACM.

[252] B. N. Steele, M. T. Draney, J. P. Ku, and C. A. Taylor. Internet-based system for simulation-based medical planning for cardiovascular disease. *IEEE Transactions on Information Technology in Biomedicine*, 7(2):123–129, June 2003.

[253] E. Stefanidi, J. Schöning, S. S. Feger, P. Marshall, Y. Rogers, and J. Niess. Designing for care ecosystems: A literature review of technologies for children with adhd. In *Interaction Design and Children*, IDC '22, page 13–25, New York, NY, USA, 2022. Association for Computing Machinery.

[254] F. Steinicke and G. Bruder. A self-experimentation report about long-term use of fully-immersive technology. In *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*, SUI '14, pages 66–69, New York, NY, USA, 2014. ACM.

[255] A. Strauss and J. Corbin. Discovery of grounded theory, 1967.

[256] A. Su, Y. Wang, and C.-H. Wu. Online visual communication and scene reconstruction of cruise travel under the background of virtual reality. *Wirel. Commun. Mob. Comput.*, 2022, jan 2022.

[257] M. Sufian, Z. Khan, S. Rehman, and W. Haider Butt. A systematic literature review: Software requirements prioritization techniques. In *2018 International Conference on Frontiers of Information Technology (FIT)*, pages 35–40, Dec 2018.

[258] W. Sun, K. Gu, G. Zhai, S. Ma, W. Lin, and P. Le Calle. Cviqd: Subjective quality evaluation of compressed virtual reality images. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3450–3454, Sep. 2017.

[259] X. Tan, C. Chng, B. Duan, Y. Ho, R. Wen, X. Chen, K. Lim, and C. Chui. Design and implementation of a patient-specific cognitive engine for robotic needle insertion. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 000560–000565, Oct 2016.

[260] R. I. Tavares da Costa Filho, F. De Turck, and L. P. Gaspary. From 2d to next generation vr/ar videos: Enabling efficient streaming via qoe-aware mobile networks. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6, 2020.

[261] E. Tikhonova, G. Efremova, and I. Terehina. Virtual reality as a tool for foreign students' anti-stress adaptation to a new educational environment. In *2020 The 4th International Conference on Education and Multimedia Technology*, ICEMT 2020, page 127–132, New York, NY, USA, 2020. Association for Computing Machinery.

[262] M. Tomlein and K. Grønbæk. Augmented reality supported modeling of industrial systems to infer software configuration. *Proc. ACM Hum.-Comput. Interact.*, 2(EICS):5:1–5:17, June 2018.

[263] Y. Tong, F. Xie, X. Zheng, and Y. Wei. Design and application of virtual training system for computer hardware assembly. In *Proceedings of the 2nd World Symposium on Software Engineering*, WSSE '20, page 146–150, New York, NY, USA, 2020. Association for Computing Machinery.

[264] D. Triantafyllou, A. Antoniou, and G. Lepouras. Sound and kinesthesis in virtual environments: Pilot experiment to compare physical and digital sound contradictions. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, PCI '16, pages 55:1–55:4, New York, NY, USA, 2016. ACM.

[265] Ulas, M. Yilmaz, and V. Isler. A literature survey: Is it necessary to develop a new software development methodology for virtual reality projects? *J. UCS*, 23(8):725–754, 2017.

[266] S. Van Damme, M. T. Vega, and F. De Turck. Human-centric quality management of immersive multimedia applications. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 57–64, 2020.

[267] E. van Wyk and R. de Villiers. Usability context analysis for virtual reality training in south african mines. In *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*, SAICSIT '08, pages 276–285, New York, NY, USA, 2008. ACM.

[268] A. Vegendla, A. N. Duc, S. Gao, and G. Sindre. A systematic mapping study on requirements engineering in software ecosystems. *J. Inf. Technol. Res.*, 11(1):49–69, Jan. 2018.

[269] W3C. *VRML Virtual Reality Modeling Language*. Web3D Consortium, 1997.

[270] M. Walch, J. Frommel, K. Rogers, F. Schüssel, P. Hock, D. Dobbelstein, and M. Weber. Evaluating vr driving simulation from a player experience perspective. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 2982–2989, New York, NY, USA, 2017. ACM.

[271] B. Wan and J. Guo. Learning immersion assessment model based on multi-dimensional physiological characteristics. In *2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pages 87–90, 2020.

[272] S. N. Wan Shamsuddin, H. Ugail, V. Lesk, and E. Walters. Towards early diagnosis of dementia using a virtual environment. In M. L. Gavrilova, C. J. K. Tan, and A. Kuijper, editors, *Transactions on Computational Science XVIII*, pages 232–247, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[273] D. Wang, Y. Zhang, J. Hou, Y. Wang, P. Lv, Y. Chen, and H. Zhao. idental: A haptic-based dental simulator and its preliminary user evaluation. *IEEE Transactions on Haptics*, 5(4):332–343, Fourth 2012.

[274] H. Wang, L. Xu, and W. Chen. Design and implementation of visual inspection system handed in tokamak flexible in-vessel robot. *Fusion Engineering and Design*, 106:21 – 28, 2016.

[275] M. W. Wani, Y. P. Gururaj, V. P, S. A. Karre, R. Reddy, and S. Azeemuddin. Velgmat: Low cost gait mat for stance phase calculation. In *2022 IEEE Sensors*, pages 1–4, 2022.

[276] S. Werrlich, P.-A. Nguyen, and G. Notni. Evaluating the training transfer of head-mounted display based training for assembly tasks. In *Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference*, PETRA '18, pages 297–302, New York, NY, USA, 2018. ACM.

[277] T. Widiyaningtyas, D. D. Prasetya, and A. P. Wibawa. Adaptive campus virtual tour using location-based services. In *2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS)*, pages 419–423, Oct 2018.

[278] D. Wiebusch, M. Fischbach, M. E. Latoschik, and H. Tramberend. Evaluating scala, actors, &#38; ontologies for intelligent realtime interactive systems. In *Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology*, VRST '12, pages 153–160, New York, NY, USA, 2012. ACM.

[279] F. Willicks, V. Stehling, A. Richert, and I. Isenhardt. The students' perspective on mixed reality in higher education: A status and requirement analysis. In *2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 656–660, April 2018.

[280] P. S. Windyga, R. Medina, and G. M. Onik. Augmented vision for minimally invasive abdominal cancer surgery. In *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE Cat. No.03CH37439)*, volume 2, pages 1160–1163 Vol.2, Sep. 2003.

[281] M. Wirth, S. Gradl, W. A. Mehringer, R. Kulpa, H. Rupprecht, D. Poimann, A. F. Laudanski, and B. M. Eskofier. Assessing personality traits of team athletes in virtual reality. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 101–108, 2020.

[282] M. Wirth, S. Gradl, J. Sembdner, S. Kuhrt, and B. M. Eskofier. Evaluation of interaction techniques for a virtual reality reading room in diagnostic radiology. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 867–876, New York, NY, USA, 2018. ACM.

[283] C. Wohlin, M. Höst, and K. Henningsson. *Empirical Research Methods in Software Engineering*, pages 7–23. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[284] J. Wolfartsberger, J. Zenisek, C. Sievi, and M. Silmbroth. A virtual reality supported 3d environment for engineering design review. In *2017 23rd International Conference on Virtual System Multimedia (VSMM)*, pages 1–8, 2017.

[285] X3D. *Extensible 3D*. Web3D Consortium.

[286] H. . Yang, Z. . Pan, Bing-Xu, and M. . Zhang. Machine learning-based intelligent recommendation in virtual mall. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2634–2639, 2004.

[287] H. Yang, Z. Shi, and Y. Zhong. The virtual campus system based on vr-platform. In *2011 International Conference on Multimedia Technology*, pages 916–919, 2011.

[288] J. Yang, T. Liu, B. Jiang, H. Song, and W. Lu. 3d panoramic virtual reality video quality assessment based on 3d convolutional neural networks. *IEEE Access*, 6:38669–38682, 2018.

[289] S. Yang, J. Zhao, T. Jiang, J. W. T. Rahim, B. Zhang, Z. Xu, and Z. Fei. An objective assessment method based on multi-level factors for panoramic videos. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4, Dec 2017.

[290] U. Yang, D. Jo, and W. Son. Uvmode: Usability verification mixed reality system for mobile devices. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 3573–3578, New York, NY, USA, 2008. ACM.

[291] U. Yang, G. A. Lee, Y. Kim, D. Jo, J. Choi, and K.-H. Kim. Virtual reality based welding training simulator with 3d multimodal interaction. In *2010 International Conference on Cyberworlds*, pages 150–154, 2010.

[292] Y.-C. Yang. Role-play in virtual reality game for the senior. In *Proceedings of the 2019 7th International Conference on Information and Education Technology*, ICIET 2019, pages 31–35, New York, NY, USA, 2019. ACM.

[293] A. M. Yasin, F. H. Yusoff, M. A. M. Isa, and N. H. M. Zain. Avatar implementation in virtual reality environment using situated learning for "sa'i" (muslim hajj ritual). In *2010 International Conference on Educational and Information Technology*, volume 2, pages V2–286–V2–290, Sept 2010.

[294] Ying Zhang, T. Fernando, R. Sotudeh, and Hannan Xiao. The use of visual and auditory feedback for assembly task performance in a virtual environment. In *Ninth International Conference on Information Visualisation (IV'05)*, pages 779–784, July 2005.

[295] S. Yu, X. Fan, and Z. Liao. A study of the endoscopic surgery simulation training system based on 3d virtual reality. In *2009 International Conference on Computational Intelligence and Software Engineering*, pages 1–4, Dec 2009.

[296] A. Zare, A. Aminlou, and M. M. Hannuksela. Virtual reality content streaming: Viewport-dependent projection and tile-based techniques. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1432–1436, Sep. 2017.

[297] H. Zhang, X. Zhang, and J. Huang. A demonstration system for the generation and interaction of battlefield situation based on hololens. In *2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pages 293–296, 2019.

[298] Y. Zhang, Y. Wang, F. Liu, Z. Liu, Y. Li, D. Yang, and Z. Chen. Subjective panoramic video quality assessment database for coding applications. *IEEE Transactions on Broadcasting*, 64(2):461–473, June 2018.

[299] L. Zhao, Y. Liu, D. Ye, Z. Ma, and W. Song. Implementation and evaluation of touch-based interaction using electrovibration haptic feedback in virtual environments. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 239–247, 2020.

[300] B. Zhou, Q. Jia, and Z. Chen. The research and development of the earthquake ruins computer aided design system for rescue training. In *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, pages 1303–1307, 2017.

[301] A. Zutshi and G. Sharma. A study of virtual environments for enterprise collaboration. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '09, pages 331–333, New York, NY, USA, 2009. ACM.