DEVELOPING A FRAMEWORK TO EVALUATE THE USABILITY OF VIRTUAL AND MIXED REALITY ENVIRONMENTS TO PRACTICE MODEL-BASED SYSTEMS ENGINEERING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

KAAN KARATAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF MULTIMEDIA INFORMATICS

NOVEMBER 2024

DEVELOPING A FRAMEWORK TO EVALUATE THE USABILITY OF VIRTUAL AND MIXED REALITY ENVIRONMENTS TO PRACTICE MODEL-BASED SYSTEMS ENGINEERING

Submitted by **Kaan Karataş** in partial fulfillment of the requirements for the degree of **Master of Science in Graduate School of Informatics, Middle East Technical University** by,

Prof. Dr. Banu Günel Kılıç _____
Dean, **Graduate School of Informatics**

Assoc. Prof. Elif Sürer _____
Head of Department, **Modeling and Simulation, METU**

Assoc. Prof. Elif Sürer _____
Supervisor, **Modeling and Simulation, METU**

Examining Committee Members:

Prof. Dr. Alptekin Temizel _____
Modeling and Simulation, METU

Assoc. Prof. Elif Sürer _____
Modeling and Simulation, METU

Assoc. Prof. Ufuk Çelikcan _____
Computer Engineering, Hacettepe University

Date: *26.11.2024*

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Kaan Karataş

Signature : _____

ABSTRACT

DEVELOPING A FRAMEWORK TO EVALUATE THE USABILITY OF VIRTUAL AND MIXED REALITY ENVIRONMENTS TO PRACTICE MODEL-BASED SYSTEMS ENGINEERING

Karataş, Kaan

MSc., Graduate School of Informatics

Supervisor: Assoc. Prof. Elif Sürer

November 2024, 48 Pages

Systems Engineering is an interdisciplinary engineering field that focuses on the identification of the required components of a product and their specifications to achieve the purpose or objective. The emergence of Model-Based Systems Engineering (MBSE) has shown that the prominent approach, Document-Based Systems Engineering (DBSE), is prone to allow over engineering of problems, having to follow a set process reducing flexibility in iterative implementation and hinder effectiveness by having repeated redundant information in multiple documents. In MBSE, the systems design process where the engineer is responsible for defining the properties and interfaces of a system, is executed in digital workspaces. The focus of this thesis is to define a framework to perform systems design with MBSE approach in virtual reality (VR) or mixed reality (MR) environments and identify the benefits and drawbacks of adapting to such environments compared to desktop environments. System modeling language (SysML) is a general-purpose modeling language stated to be capable of encapsulating all required information. A prototype application, MRSysML, as a proof-of-concept for the framework is developed to support using SysML in VR or MR environments and another prototype for desktop environments, 2DSysML, is developed for comparison. User test sessions with 30 participants with equal number of participants from systems engineering background and interactive design background are executed. The outcomes of these sessions demonstrate that the VR or MR adaptation is a useful approach which increases the enjoyability and engagement of the user while having room for improvement regarding textual input and object manipulation.

Keywords: Mixed Reality, Virtual Reality, Model-Based Systems Engineering, Systems Design, Framework

ÖZ

SANAL VE KARMA GERÇEKLİK ORTAMLARININ MODEL TABANLI SİSTEM MÜHENDİSLİĞİ UYGULUANMASINDA KULLANILABİLİRLİĞİNİN DEĞERLENDİRİLMESİ İÇİN ÇERÇEVE YAZILIM GELİŞTİRİLMESİ

Karataş, Kaan

Yüksek Lisans, Enformatik Enstitüsü

Tez Yöneticisi: Doç. Dr. Elif Sürer

Kasım 2024, 48 Sayfa

Sistem mühendisliği bir ürünün alt bileşenlerinin ve bileşenlerin özelliklerinin belirlenmesi amacına sahip bir disiplinlerarası mühendislik alanıdır. Yaygınlaşan Model Tabanlı Sistem Mühendisliği (MTSM) yaklaşımı, önde gelen Doküman Tabanlı Sistem Mühendisliği (DTSM) yaklaşımının gereğinden fazla mühendislik uygulanmasına yatkın olduğunu, sabit bir süreç takip edilerek aşamalı geliştirmelerin yapılmasının esnekliği azalttığını ve tekrar eden lüzumsuz bilgilerin birçok dokümanda tekrarlandığını göstermiştir. Mühendisin sistemin özelliklerini ve arayüzlerini belirlemekle sorumlu oluğu MTSM tasarım süreçleri dijital çalışma ortamlarında gerçekleştirilmektedir. Bu tezde, MTSM yaklaşımıyla sistem tasarım süreçlerinin sanal gerçeklik (SG) ve karma gerçeklik (KG) ortamlarda yapabileceği bir çerçeve yazılım tanımlanmıştır ve bu çerçeve yazılımın bilgisayar ortamlarına karşı yararlarını ve eksikliklerini tanımlamaya odaklanılmıştır. Sistem modelleme dili (SysML) genel kullanıma uyumlu ve bütün sistemleri tanımlama yeteneğine sahip olduğu belirtilen modelleme dilidir. SysML dilini destekleyen, SG ve KG ortamlarında çalışan, soyut çerçeve yazılımına kavram kanıtı olan, MRSysML isimli bir örnek yazılım ve karşılaştırma yapılması amacıyla bilgisayar ortamlarında çalışabilen 2DSysML isimli bir yazılım geliştirilmiştir. Yarısı sistem mühendisliği geçmişi olan ve diğer yarısı etkileşimli uygulama geliştirme geçmişi toplam 30 katılımcı ile kullanıcı test oturumları gerçekleştirilmiştir. Elde edilen bilgilere bağlı olarak SG ve KG uyarlanması eğlenceyi ve etkileşimi arttıran, yazısal girdi ve nesne idaresinde gelişmeye açık olan, kullanışlı bir yaklaşım olduğu tespit edilmiştir.

Anahtar Sözcükler: Karma Gerçeklik, Sanal Gerçeklik, Model Tabanlı Sistem Mühendisliği, Sistem Tasarım, Çerçeve Yazılım

To My Dearest Family

ACKNOWLEDGEMENTS

First of all, I would like to express my sincerest gratitude to my supervisor, Assoc. Prof. Elif Sürer, for their immense support and guidance. Since the beginning of my admission to the department, they have shared their astonishing knowledge and inspired me to aim for accomplishments I did not believe I could have achieved. I am fortunate and grateful for having the opportunity to work with them.

I want to express my appreciation to all participants in the user tests who have shared their valuable time to attend the sessions. This work would not have been possible without their insights, comments, and contributions.

I am thankful to my family for their encouragement. They motivate me to become an engineer and want to explore the related fields. In turn, they are why I decided to study this field. Especially my grandmother, Prof. Dr. Eren Kum, and my grandfather, Prof. Dr. İlhan Kum. While they are no longer with us, their works and achievements have always inspired me. Without their example, I may never have taken any steps in the academic path.

I am deeply grateful to my previous manager, Murat Şahin, and my current manager, Ertan Eyimaya, who have supported my academic work and allowed me to prioritize it as needed. They have shown me that while we have worked together, what we accomplish is more than just our job.

To my lovely wife İpek İmdat Karataş, who has shown endless patience and heartened me throughout this journey and since the day we met. You have always been a source of peace and calm. You have stood by me in the good and the bad. Thank you for supporting me and never giving up on me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AADL	Architecture Analysis and Design Language
AUTOSAR	Automotive Open System Architecture
CGS	Curved Grading Scale
DBSE	Document-Based Systems Engineering
INCOSE	International Council on Systems Engineering
IoT	Internet of Things
ITQ	Immersive Tendencies Questionnaire
MBSE	Model-Based Systems Engineering
MIL-STD	Military Standard
MR	Mixed Reality
OCL	Object Constraint Language
OMG	Object Management Group
PQ	Presence Questionnaire
SoS	System-of-Systems
SoSE	System-of-Systems Engineering
SUS	System Usability
SysML	System Modeling Language
UML	Unified Modeling Language
UX	User Experience
V&V	Verification and Validation
VR	Virtual Reality
XR	Extended Reality

CHAPTER 1

INTRODUCTION

1.1 Foundation and Motivation of the Research

Systems engineering is an interdisciplinary field of engineering that dates back to the early 1940s [1]. It aims to set the requirements and define the components and elements of a product. To overcome the complexities of large-scale projects, systems engineering is key to understanding the primary goal as individual simpler challenges instead of one complex problem. The resulting base-level work packages can then be implemented with a divide-and-conquer approach, where different teams work on separate packages simultaneously or by applying the predefined requirements iteratively in multiple sequential phases.

With the ever-improving capabilities of computers, many engineering practices shifted to using digital workspaces to design their products. From printed circuit boards to software design, almost all engineering fields rely on some software. Systems engineering has not adapted to these workspaces as quickly. The prominent methodology, Document-Based Systems Engineering (DBSE), aims to define the products' requirements and design into written documents. Therefore, digital workspaces are limited to documentation tools such as Microsoft Word or Microsoft Excel for systems design.

The documents for DBSE are commonly standardized based on the industry and type of the product. For example, MIL-STD-498 [2] is a military standard for software development and documentation, which describes the requirements of a software system in the System Subsystem Specification (SSS) document, the design decisions and architecture in the System Subsystem Design Description (SSDD) document, and many other types of documents. However, it is identified that the DBSE methodology may cause unnecessary complexity by distributing information into too many sources and causing redundancy between the documents [3]. In recent years, the solution to these problems has been proposed to adapt to a different methodology named Model-Based Systems Engineering (MBSE). The history of MBSE dates back to 1993 [3], the application of the methodology has not been widespread until recently. MBSE describes that the systems' design and architecture can be captured into models, allowing the systems engineers to utilize the digital workspace to design and represent their work.

Following the increasing adaptation of MBSE, several standard modeling languages have become available, such as Architecture Analysis and Design Language (AADL)–a modeling language for safety critical systems– standardized by the Society of Automotive Engineers [5] and System Modeling Language (SysML)–an adaptive modeling language for general purpose usage–standardized by Object Management Group (OMG) [6]. With standard modeling languages, modeling tools such as Enterprise Architect by Sparx Systems or Engineering Systems Design Rhapsody by IBM have started supporting modeling in these languages. Yet, no commercially available system modeling tools allow systems engineers to work using

virtual reality (VR) or mixed reality (MR) devices. Furthermore, in the literature, while there are works utilizing VR for systems design, no common framework is used in such research.

Using VR and MR devices provides unique capabilities compared to two-dimensional work environments. Viewing something from multiple angles in three dimensions enables the differentiation of objects. The controllers allow a more intuitive way of working since they are designed to act similarly to hand-based actions such as grabbing, pointing, or pulling a trigger. This sensation is advanced even further in the hand-interaction mode of MR devices. Concurrent collaboration is one of the primary research topics. The capabilities of VR or MR devices enhance the experience by letting one user see the other as virtual avatars. Finally, the advancement of hardware suggests that in the future, people will be able to carry their devices anywhere. The newest announced hardware, such as Snap Spectacles 5 or Meta Orion, shows that carrying around the MR devices may be easier than laptops, and the ergonomics may become even better than any computer environment.

1.2 Purpose of the Research

The purpose of this research is to evaluate the usability of VR or MR environments for conducting MBSE and compare it to desktop environments. The objectives of the thesis are as follows:

- To define a framework for system design using VR or MR devices alone without needing to use desktop-based applications and provide the fundamental capabilities.
- To develop a prototype for the framework that works on commercially accessible hardware.
- To compare the prototype to a desktop-based prototype to assess the advantages and disadvantages of working in VR or MR environments.
- To evaluate the framework through different perspectives and identify the adjustments required for future work.

1.3 Contributions to the Field

This thesis serves as a fundamental study to understand the benefits and needs of applications where systems engineers can employ MBSE systems design practices in VR or MR environments. The findings are supported by user testing sessions conducted with people familiar and unfamiliar with such devices or systems engineering practices. Since there is a lack of a common framework for such applications in the literature, the research describes an abstract framework's basic requirements and design. A prototype o framework is developed to examine the possible capabilities and compare its usage to desktop applications. The growing technologies in VR and MR devices may allow tools that utilize these environments to replace desktop environments in specific fields. This thesis serves as a guideline for what improvements must be made to achieve it for systems design tools. Future studies may use this thesis' outcomes to further investigate the framework or similar applications from different perspectives, such as determining the productivity improvements or disadvantages of using such applications.

1.4 Thesis Outline

The thesis has seven chapters, and the contents of each chapter are as follows:

- Chapter 1 contains the scope of the research and fundamental information about the field, the purpose of the thesis, its contributions, and the thesis outline.

- Chapter 2 explains the state-of-the-art through recent work on systems engineering, MBSE, utilizing VR for MBSE, and utilization of VR in similar fields such as software development and design.

- Chapter 3 presents an abstract modeling framework's fundamental requirements and design along with the prototype's design and implementation details.

- Chapter 4 describes the evaluation procedure through user tests, the tools used for data collection, and the determination methods of the gathered data.

- Chapter 5 includes the results of each evaluation criterion.

- Chapter 6 discusses the gathered data results and explains the determined outcomes.

- Chapter 7 summarizes the thesis's results and provides insight for future work in the field.

CHAPTER 2

LITERATURE REVIEW

2.1 Standardization and Evolution of Systems Engineering

The origins of systems engineering practices date back to World War II [1]. The name of the term was first introduced back in 1950, yet there is no single accepted definition for it in the literature. Each organization or group redefines the term with slight differences and provides its unique lifecycle. Military Standard Systems Engineering Management (MIL-STD-499) [7] is the first formal systems engineering process standard dating back to 1969, and it was introduced by the U.S. Air Force. Its applicability and maintenance were declared canceled in 1995 [8]. In 2017, the IEEE Standard for Application of Systems Engineering on Defense Programs was declared as the replacement for the U.S. Department of Defense [9]. The International Council on Systems Engineering (INCOSE) is another large organization that attempts to standardize the meaning and practice of systems engineering, and it was formally founded in 1990. It began as the National Council on Systems Engineering (NCOSE) to train systems engineers and was later expanded to incorporate an international scope [1]. They publish the "Systems Engineering" journal monthly, which contains the current advancements in the systems engineering field, and host an annual international symposium and an annual international workshop. The definition for the term by INCOSE is "a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods" [10]. MITRE Corporation is an organization founded back in 1958 that has worked on applying systems engineering solutions to countless military and civilian projects [11]. They claim that there is no single definition of what systems engineering is, but its meaning is defined as context-specific based on where and how it is employed [12]. While the definition of the term may vary based on the context, the general practice in the industry and literature seems to be consistent that it is an interdisciplinary engineering field ranging from chemistry to software and more, and its purpose is to define the users' needs and provide the architecture and design that would accomplish it while providing a roadmap on when and what to do.

System of Systems Engineering (SoSE) is an expansion of classical systems engineering to provide additional capabilities through the integration of multiple systems [13]. In SoSE, a group of working systems, individual systems serve particular purposes independently and can interact with other systems for further features. For example, in a home setting, some smart home hub devices can allow the user to access an artificial intelligence (AI) assistant while a smart light bulb can be programmed to turn on at specific times, but combining these two systems can let your AI assistant to control the lights without the need of the programming software of the light bulb. With the ever-expanding capabilities of computer networks and technologies, SoSE has become one of the hot topics in the literature in recent years. It has its unique standard definition and processes. Vargas and Braga [14] have systematically reviewed 31 studies filtered out from 3495 studies to identify the key and pain points of applying SoSE. The primary issue defined in SoSE is SoS management, which requires having a holistic view from the start. With a lack of tools to support SoS, management reduces the efficiency of practicing SoSE.

2.2 Model-Based Systems Engineering

The Model-Based Systems Engineering term is coined by A. Wayne Wymore in their book with the same name [4]. The MBSE process suggests that the architecture of the system and its design should be encapsulated in visual models. Madni and Sievers [15] have investigated the field in 2018 defining the less explored opportunities and explaining the status of the field. They state that the effort in academia shows that the MBSE is beneficial compared to document-centric practices. Still, there was not enough support from the management and customers standpoint. They also list a set of required advancements regarding processes and tools. Afterwards, in 2022, Campo et al. [16] conducted a similar investigation. The outcome also suggested an overwhelming number of studies suggesting MBSE is beneficial. However, they also noted that 47% were opinion-based, and most of the claims were not supported by any metrics. Regarding the drawback against MBSE, only 37% were opinion-based. They speculate that the increasingly positive outlook about MBSE may originate from unintentional favoritism by large organizations such as INCOSE promoting MBSE's benefits more than its impediments. Even after removing the opinion-based studies, MBSE's advantages and preferences still outweigh its disadvantages, even though they are less overwhelming.

Adedjouma et al. explore the advantages and challenges of incorporating MBSE in a work environment [3]. In their study, they investigate Plastic Omnium's switch to MBSE in order to comply with specific standards and share their designs with many suppliers while developing an application using Automotive Open System Architecture (AUTOSAR), a model-based architecture adopted by many automotive manufacturers. Plastic Omnium's previous work focused on text-based specifications. However, when adapting to AUTOSAR, instead of preparing text-based specifications and converting them to system models, they decided to embrace MBSE using SysML company-wide. SysML inherently satisfied AUTOSAR's compliance requirements, so they stopped producing text-based designs and discovered that MBSE reduced the risks of human error and redundant work. Adedjouma et al. noted that MBSE is beneficial for identifying the over-engineering done with the DBSE approach and provides freedom in workflow since there is no need for a certain order of procedures.

Yang et al. have utilized MBSE to simulate a battlefield environment and systems deployed [17]. It is a SoS environment where each system, such as an early warning aircraft or drone, can act independently while being able to interact and operate with one another as a joint force. Their design allowed operators and commanders to exercise and plan many unique battlefield scenarios. They used SysML as the modeling language and augmented it with an environment simulation developed in Unreal Engine 4. They could follow the interaction of systems and foresee problems. They have shown that system models are a strong tool for simulating and preemptively solving interoperability problems that could arise during runtime operations, and they have also provided proof-of-concept for digital twin implementations using MBSE.

While there are several works identifying the benefits of adapting MBSE, Cameron and Adsit [18] conducted a poll to find industries' tendencies to use MBSE. The poll was sent to the 4200 attendants of an online course, and around 1000 responded. They asked about the reason behind their interest and their workplace's stance on the subject, and 35% of the respondents were from workplaces using the MBSE approach. Even though the years of work experience were on the high end, over 10 years of experience, the usage of commercial programs and modeling languages was comparatively low. 1% stated high usage, 2% stated moderate usage, 4% stated low usage, and 93% noted no usage, indicating an inefficient adaptation to the model-based approach. Cameron and Adsit state that there are two outcomes of their poll regarding the adoption of MBSE, the adoption rate is low, and the current form of adoption may be ineffective; therefore, it should be evaluated further.

2.3 Model-Based Systems Engineering through Virtual Reality

Conducting MBSE in virtual environments is not a new topic. Lutfi and Valerdi [19] have found Kande's thesis [20] as one of the first attempts to integrate them, Kande used a VR environment to display a simulated environment and connected it to a SysML model on the back end. Lutfi and Valerdi explored other MBSE and VR integrations to identify what could be built upon and stated that one of the missing elements was a common framework that academic works can utilize. Therefore, they proposed to define a framework that can be used by researchers. In their later work, they defined a framework that combined models developed in Cameo System Modeler (CSM) and used Unity Engine to create an application to display these models [21] in a virtual environment. They used MATLAB scripts to input and output information between their VR tool and CSM. This way, the SysML model can be displayed on the VR tool, and it can be simulated as a system as well. Furthermore, Lutfi and Valerdi performed a case study that highlighted their framework [22]. In this case study, they designed a ground-based telescope system and simulated it in the VR tool. It served as proof-of-concept and displayed how the framework could be applied. They suggested this could also be used as a Digital Twin technology. As long as data are gathered in the model form, it can be visualized like simulated information.

Another strength of using virtual environments for system model display is the third dimension. All commercially available modeling tools are limited to desktop environments, and the number of displays required can get out of hand when it comes to data being distributed into many diagrams, such as in SysML. Oberhauser defined a unique way where the third dimension can be utilized [23]. In their early work, they identified that the third dimension can be utilized to show the interaction of different diagrams within one another if each diagram is displayed in a stacked manner. Their initial work suggested that as a standard feature that can be utilized in any modeling language and used UML to create a prototype. They then expanded this prototype to support SysML and compared their prototype to Enterprise Architect by Sparx Systems [24]. They state that the three-dimensional virtual display allows for a better understanding of the system, especially for stakeholders who are not familiar with systems engineering processes. Finally, Oberhauser included test coverage and status to improve further the already strong Verification and Validation capabilities of MBSE [25]. By adding annotations to a program and test cases, they trace the code to the SysML model. This generates new types of connectors to their diagrams that could show the test status by color and coverage percentages.

2.4 Application of Virtual Reality in Similar Practices

While using VR or MR devices for MBSE is a relatively new topic, software architecture design usage of VR has advanced. Yigitbas et al. [26] [27] have proposed using VR environments as an immersive environment to train students on how to design UML diagrams. They prepared three variant minigames to perform training scenarios in their application named GaMoVR. The minigames contain a hangman-like figure that progresses based on the mistakes made. The user aims to avoid making mistakes and finish the games before the hangman gets hanged. In their design, Yigitbas et al. did not follow the diagram shapes of UML specifications but customized them into three-dimensional objects. They plan on improving their application by adding new minigames to encapsulate more aspects of UML modeling. Additionally, GaMoVR also provides a baseline framework that supports multi-viewpoint model visualization and interaction. Similar applications may become available for MBSE to train beginners. Since systems engineering is an interdisciplinary field, the barrier to entry is a difficult challenge to overcome.

The usage of VR for designing UML diagrams is also advanced in collaborative work environments. Yigitbas et al. [28] have developed an application that allows class diagrams to be designed collaboratively in VR. Two users could see and interact with the same class diagram concurrently. They have gathered 24 participants as a case study to investigate the effectiveness of such a work environment. Their findings suggested that users preferred using desktop applications if the users were in the same physical space. Yigitbas et al. suggest that a VR approach may be more applicable to remote work environments. Compared to web-based collaborative design tools, sharing the virtual environment felt natural. The same arguments can be claimed for MBSE using VR. It shows that while VR may not be the replacement for the desktop environment, it may be an alternative for specific cases.

CHAPTER 3

FRAMEWORK AND PROTOTYPES

3.1 Framework Architecture

The proposed framework consists of four main components: 1) The modeling language data structure library, which includes the metadata for forming the model, 2) A data storage manager which stores the data in the applications' cache or some form of database, 3) Model accessor which provides the create, read, update, and delete (CRUD) functionality for the model, and 4) A visualization adapter responsible for displaying the model information. The data structure library must be accessible by all other components. The visualization adapter sends CRUD queries received by the user through the model accessor. The accessor triggers the data storage manager to update the model data in runtime without requiring any constant save or load operation. The queries must utilize the data structures as properties. The architecture of the framework is depicted in Figure 1:

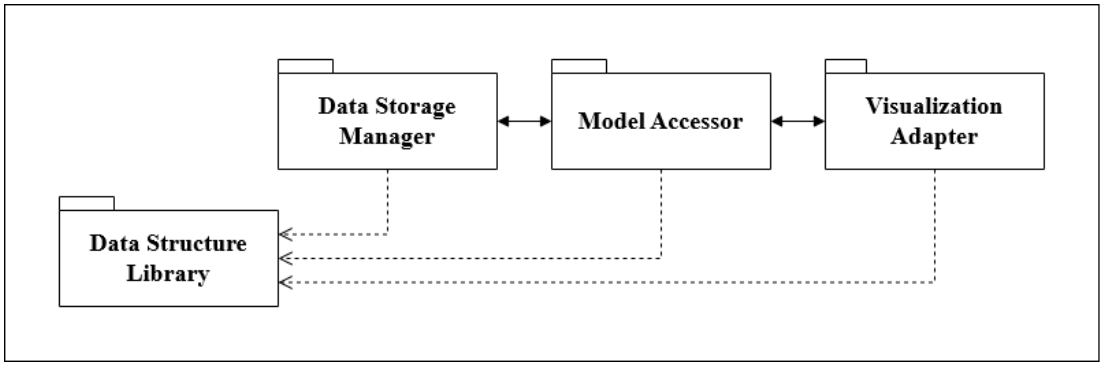


Figure 1 – The abstract architecture representation of the proposed framework.

Having the abstraction of a model accessor, the framework can be further improved by adding separate components for additional capabilities. Several examples of additional capabilities are importing and exporting the model, digital twin representation, and system simulation.

3.2 Design Decisions for Prototypes

The MR or VR prototype is a proof-of-concept for the abstract framework. The capabilities of modeling tools are vast. Therefore, the prototypes' development processes are simplified by building a monolithic software where each system component is a software package that is part of the singular application. In an ideal scenario, for commercial products, it is more beneficial to implement separate components that are unique applications and services that support a distributed architecture. A distributed architecture could divide the workload from the VR or MR device to a central server, allowing access to multiple actors for collaborative work and reducing the processing done by the device.

To perform a comparative analysis of desktop applications with VR or MR applications, the same framework is developed for both environments, implementing separate visualization adapters based on the target environment. Since the data structure library, data storage, and model accessor components are backend components without user interface components, changing the visualization adapter is enough.

3.2.1 Development Environment

The prototypes are developed using Unity Engine [29]. It hosts a vast range of third-party dependencies for VR or MR development, and several large companies, such as Meta, provide their own integration tools and software development kits for that purpose. Unity Engine has many features that reduce the effort required to develop any project. It has a package manager to maintain the dependencies, tools, and assets. Unity Engine includes a physics engine that supports two-dimensional (2D) and three-dimensional (3D) environments. Its editor also provides inherent real-time visualization and debugging tools.

The prototypes are developed in C# high-level programming language [30]. Unity Engine utilizes C# as the primary programming language, and it is possible to integrate different languages through specific libraries. Due to the simplicity of the prototypes, there is no significant advantage in favor of any programming language.

3.2.2 Third Party Dependencies

The prototypes employ Mixed Reality Toolkit version 2 (MRTK2) [31] for device integration, look-and-feel, and user interface. MRTK2 is a toolkit that allows developers to integrate their projects into many VR and MR devices easily. The primary configuration of the VR or MR prototype is configured to run on Meta Quest 2. It can be reconfigured before compilation to fit the other devices or capabilities, such as Meta Quest 2's black-and-white passthrough feature to enable mixed reality. Switching between VR and MR during runtime is not enabled but has to be performed before the start of the prototype. It also has pre-defined materials that can be used to keep a common look and feel for both the desktop and VR or MR applications. MRTK2 supports the usage of Unity GUI. The raycast from the controllers can interact with Unity GUI elements, and a set of Unity GUI elements can be implemented to work in all environments.

Text Mesh Pro is a dependency utilized to display texts and their integration. MRTK2 also depends on Text Mesh Pro for its prefabricated assets. All GUI elements that include textual display use Text Mesh Pro and its accessory functions for user interaction.

3.2.3 Modeling Language

System Modeling Language (SysML) is a modeling language maintained by Object Management Group (OMG) [6]. It is derived from Unified Modeling Language (UML), also maintained by OMG [32]. SysML follows Object Oriented Systems Engineering Methodology meaning every detail of the specification is described as objects. SysML is derived from UML and uses Object Constraint Language (OCL) to define each object and expression [33]. The use of SysML is widespread in the literature and industry. It is a general-purpose language that is proclaimed to be able to contain all necessary information for a system. Thus, it is used as the target modeling language for the prototypes.

In the specifications, each classifier has eight properties. These properties are name, description, attributes, association ends, operations, generalizations, specifications, and constraints. The name is a unique identifier of the type of classifier. The description is for users to understand the purpose of the classifier. The attributes are the primitive variables each classifier contains. The association ends are the relation of a classifier to others. The quantity of each association end varies and may be one-to-one, one-to-many, many-to-one, many-to-many, or any specific number of classifiers to classifiers. The association ends may be one of three types, simple, derived, and composition. Also, association ends may have specific properties, such as redefines. Operations are the types of functions the classifier can compute.

Generalizations are the parent classifier types where the classifier inherits all properties. Specializations are the child classifier types that inherit their properties. Constraints are types of calculations that prevent the classifiers from being initialized. A description of the properties from a software development standpoint is listed in Table 1:

Table 1 – Description of SysML terms from a software development standpoint.

Property	Description
Classifier	Object type or object class
Name	Objects name
Description	Any form of documentation comment for the object
Attribute	Object's primitive non-static variables and enumerations
Association end	Object's non-primitive non-static variables except enumerations
Operation	Object's methods
Generalizations	Object's inherited object types
Specializations	Object's implementor object types
Constraints	Logical validation operations that may result in exceptions during initialization

UML and SysML are designed so that each modeling language can be described through the language. SysML utilizes a subset of UML object types grouped as UML4SYSML and introduces its unique classifications for the existing object types.

There are three root object types where all other objects inherit at least one type. These root object types are element, diagram, and diagram element. To avoid confusion, elements are aliased as model elements. Diagram elements are visual objects such as shapes, edges, or images. Each diagram element relates to a model element that contains its logical information. Model elements are the actual sources of data and processing. Diagrams are containers for diagram elements and have individual purposes. The hierarchy of type for diagrams in SysML is as depicted in Figure 2:

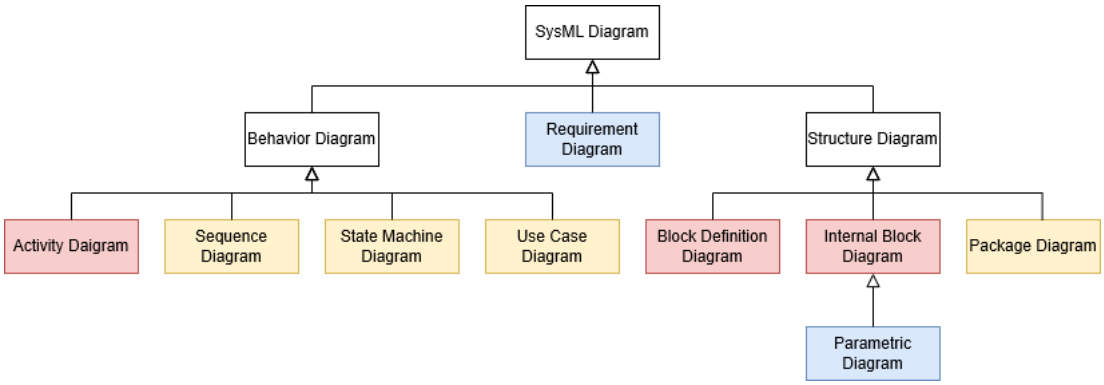


Figure 2 – The hierarchy of type for diagrams in SysML.
Source: Adapted from [6].

White-colored diagram types are abstract diagram types used to categorize their specializations. Yellow-colored diagram types are inherited from UML without any changes. Red diagram types are inherited from UML but are altered in certain ways. Blue diagram types are diagrams unique to SysML. A requirement diagram presents the requirements of a system and the requirements' relationships to other requirements and tests. A block definition diagram includes the properties of a specific block, which can be described as a system component or configuration item. An internal block diagram depicts the structural elements of a block, such as a connector, interface, and part. A parametric diagram describes the arithmetic and logical operations as constraint blocks. A package diagram contains the whole project and its objects.

An activity diagram instructs how the system behaviors are through control flow and object flow. A sequence diagram illustrates the sequential information flow between two or more blocks. A state machine diagram encompasses a block's states and state transition flows. A use case diagram highlights the actors, the use cases of the systems, and their relationships.

3.2.4 Utilized Design Patterns

Four main design patterns are utilized in the prototypes. These design patterns are the singleton pattern [34], event listener pattern [35], context pattern [35] and factory method pattern [36]. Each design pattern is utilized by multiple software packages.

Singleton pattern [34] suggests having a static object that is accessible by the whole application. It is implemented by a class with a private constructor and a static function that returns the static instance of the object. To prevent double instantiation, the instance function has a thread lock in case multiple threads attempt to access the instance simultaneously. The thread lock holder checks if the private static instance is initialized. If the instance is initialized, it returns the instance. If not, it constructs the instance and returns it. After initialization or before returning the instance, the thread lock is released. Having a singleton class can simplify complicated algorithms and reduce the need for redundant passing of variables in function calls. An example use case for the singleton pattern is using a logger object to log debug information. The logger is initialized and configured once so that all classes can use the same configuration without having to reconfigure the logger. The diagram of a singleton logger is shown in Figure 3.

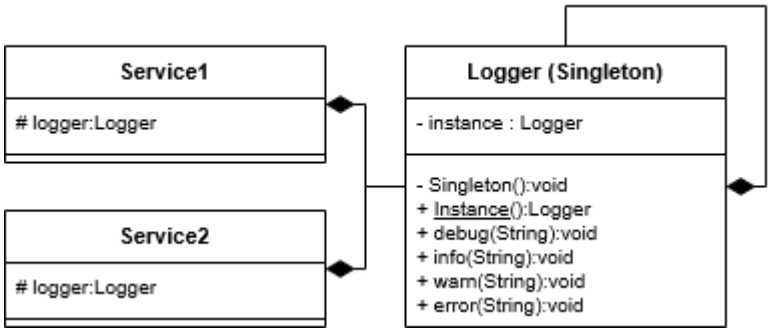


Figure 3 – Diagram of a singleton logger object shared by two services.

Event listener pattern [35] is utilized for real-time updates and visualization. The event listener pattern defines events, invokers, and listeners. An event is an object that is accessed by both the invoker and the listener. An invoker is the triggering object that invokes the event. A listener is an object that performs a specific function when the event is invoked. The implementation of the event listener pattern can use both multi-threaded or single-threaded approaches where the listeners' functions may be triggered in the order of listens or concurrently by multiple threads. The design pattern is frequently used for front-end development to trigger events based on user actions, such as a button press. Physics engines like Unity Engine can also implement the event listener pattern to trigger time-based events such as frame refresh. The prototypes utilize a single-threaded approach and use C# programming language's inherent EventHandler class and event primitive class defined in the scope of the System namespace. Any modification in the SysML model is triggered as an event at the visualization adapts accordingly. All user actions also invoke unique events that the prototypes act accordingly. A diagram of the event listener pattern for dynamic visualization is included in Figure 4.

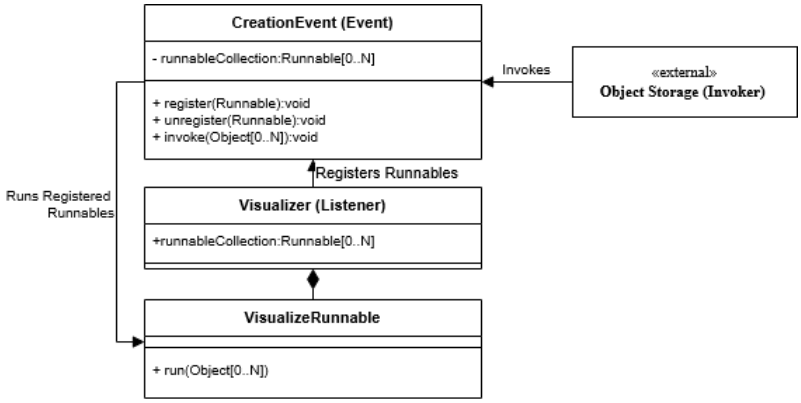


Figure 4 – Diagram of an abstract event listener pattern for visualization based on object change.

The context pattern [35] requires having application-level shared objects registered to the context, commonly known as application bundles. The context involves the registration of each object to the context by keys. An id or the class of an object is generally used as the key. In monolithic applications, classes responsible for main computation–also known as services– are commonly initialized upon start-up and registered to a context. Each class that requires access to a particular service can query it from the context. The context is similar to a database where instead of storing the objects in the disk space, they are stored in the cache. Prototypes employ context patterns to store the system model information. A diagram of the context pattern for data storage is shown in Figure 5.

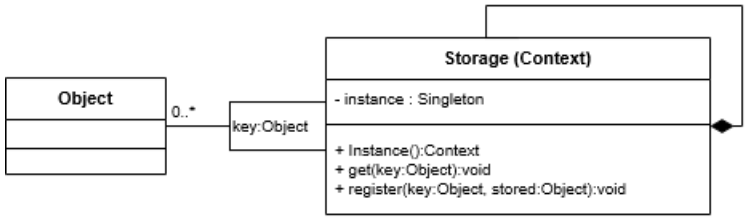


Figure 5 – Diagram of an abstract context pattern for data storage.

The factory method pattern [36] allows a creator interface to instantiate objects that implement a standard abstract class or interface through a single factory method. By implementing the creator interface differently, the objects instantiated can be modified. This pattern is beneficial if the same kind of object has to be instantiated many times based on limited arguments. For example, it can generate standard buttons shared amongst many windows where each implementation of the creator instantiates different buttons. A depiction of the factory method pattern in the context of SysML is exhibited in Figure 6.

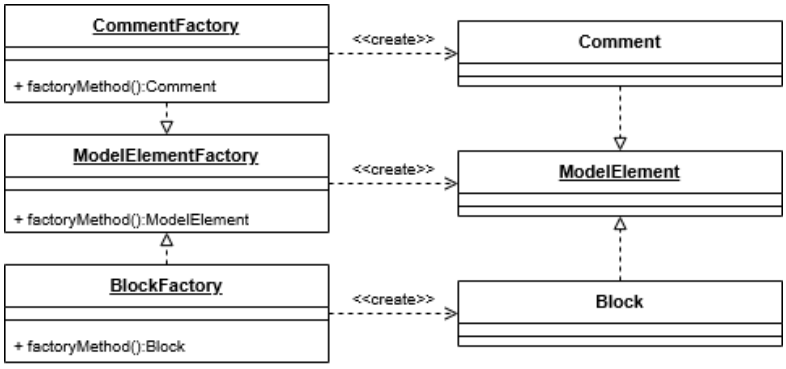


Figure 6 – An example of the factory method pattern in the context of SysML to produce model elements.

3.3 Prototype Implementation Details

The prototypes developed for the study are named MRSysML and 2DSysML. The prototypes consist of six software packages. These packages are OCL Based Data Storage Manager (OBDSM), UML Data Structures (UDS), SysML Data Structures (SDS), SysML Model Manager (SMM), MR Visualization Adapter (MRVA) and Two-Dimensional Visualization Adapter (2DVA). The software packages of the prototypes are depicted in Figure 7.

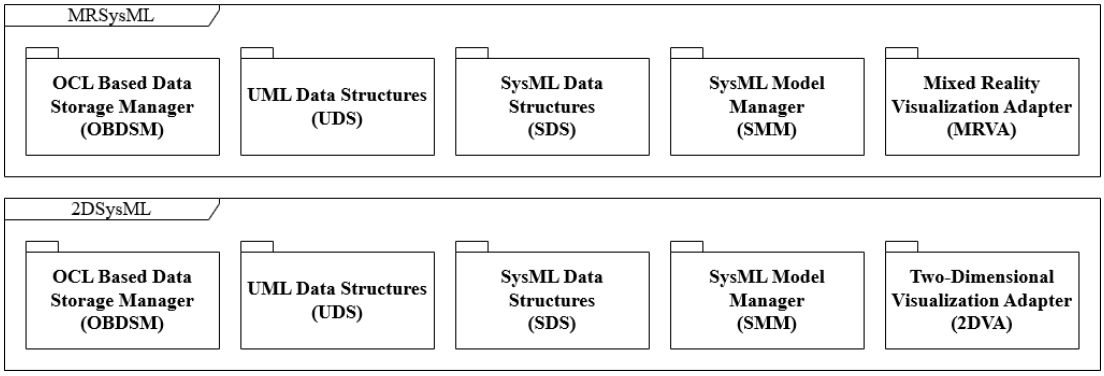


Figure 7 – The package structure of the prototypes and the components of these packages.

The packages do not match with the framework's components one-to-one, but they provide all defined features. OBDSM acts as a basis for the data structure library and the data storage manager. UDS and SDS contain the classes for the data structure library. SMM provides a library and a service that the visualization adapters can use as the model accessor. MRVA and DVA are the visualization adapters for the prototypes. The relationship of the packages' components to the framework's components is depicted in Figure 8.

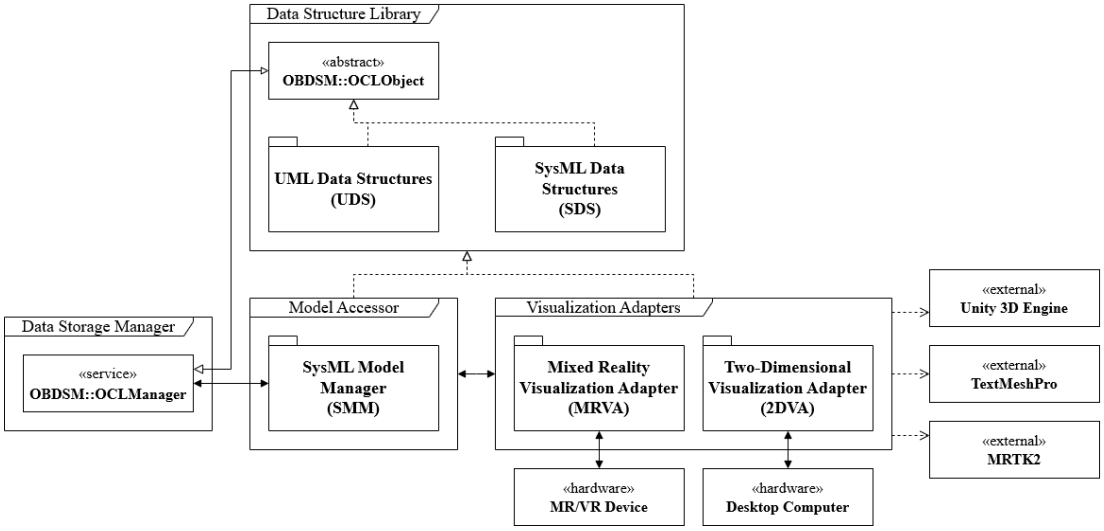


Figure 8 – Illustration of each software package's responsibilities based on the framework description.

3.3.1 OCL-Based Data Storage Manager

Since C# high-level programming language does not support multiple inheritance except for interfaces, an implementation for an application-level context-based data storage design is adopted. OBDSM contains two subcomponents. The OCLObject class is an abstract class that declares the attributes of any object type and each data structure to implement, and the OCLManager class acts as the data storage manager to contain any data structure that implements OCLObject. The package diagram for OBDSM is depicted in Figure 9.

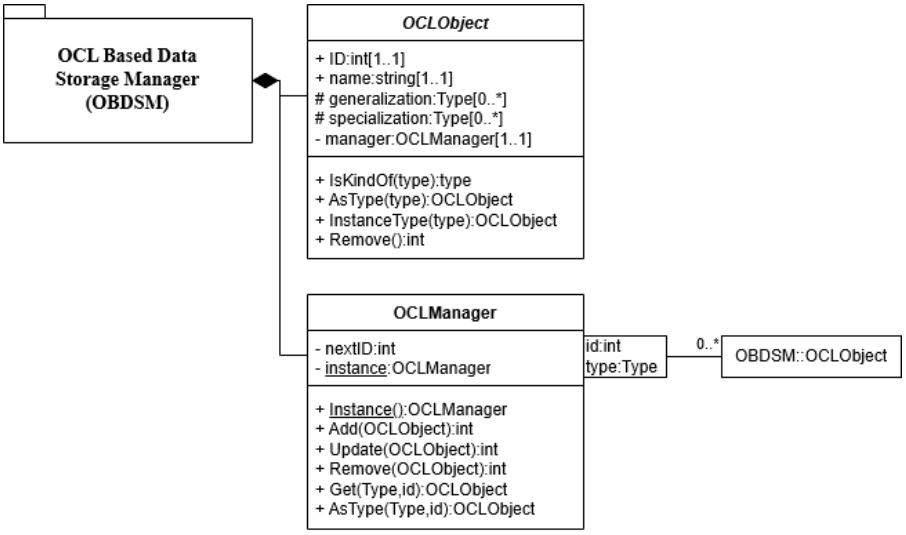


Figure 9 – Package Diagram of OBDSM.

The OBDSM provides the application with the capability of storing all objects in the cache. It follows the context design pattern [35] where all model and diagram elements are registered in the application context. The OCLManager is responsible for storing all objects in a two-dimensional dictionary. It implements the singleton pattern [34] so that all packages of the prototype use the same context. Each object is identified by its class and unique identifier number. The OCLObject abstract class has four main attributes: the name of the class, its generalizations, specializations, and unique identifier. The specializations and generalizations of any OCLObject use the same identifier. Any instantiated OCLObject is responsible for instantiating its generalizations and registers itself to the OCLManager. During construction, if no identifier is provided as an argument, an OCLObject queries the next available identifier from the OCLManager. OCLManager provides an identifier as a 64-bit integer in cyclic order, skipping any occupied identifier. Each OCLObject instance can access a specialization or generalization of itself by querying the same identifier and different class from the OCL manager. Upon removal of any OCL object from OCLManager, all its generalizations are recursively removed.

3.3.2 UML and SysML Data Structures

The SysML Data Structures (SDS) package contains all classifiers and associations defined in the SysML specification by the Object Management Group (OMG) [6] and the UML Data Structures (UDS) as a subpackage. UDS contains all classifiers and associations defined in the SysML specification marked as UML4SysML in accordance with the UML specification by the OMG [32]. Each root class of UML specification–model element, diagram element, and associations–has its own abstract class, which implements the OCLObject. The distinction is used to separate the events and arguments that utilize the several types of objects. For example, the visualization adapter, when a diagram element information is changed, only modifies the diagram information of the model and does not redefine the model elements. This reduces the iteration required to find the desired element and provides a more understandable implementation. The package diagram for SDS and UDS is depicted in Figure 10.

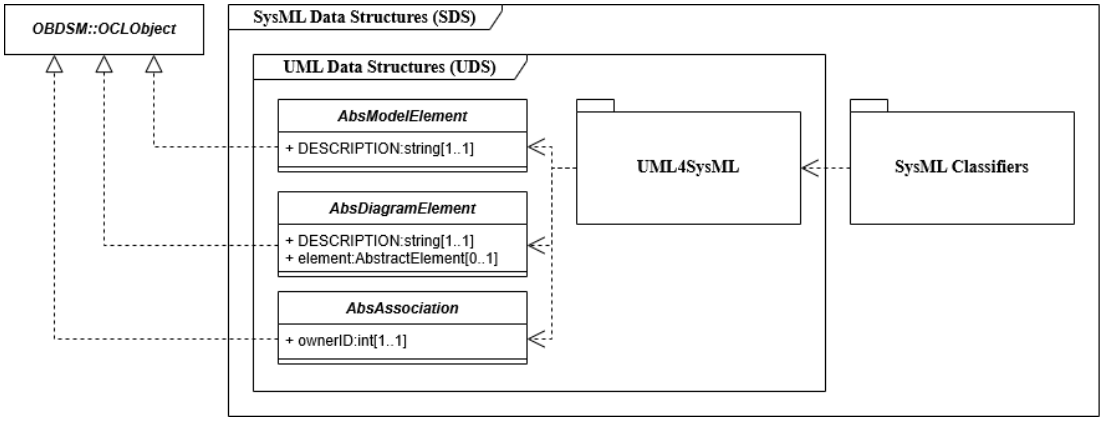


Figure 10 – Package Diagram of SDS and UDS.

All classes have all attributes defined in accordance with their specifications. The name of the classifier, its description, generalizations, specializations, associations, and attributes can be accessed as public variables. All classes access the generalizations of themselves through the OCLManager and inherit the operations and attributes through this relationship. Any redefined operation is re-implemented in the scope of the most specialized class.

3.3.3 SysML Model Manager

SysML Model Manager (SMM) is responsible for initializing the SysML model and providing an application programming interface for the OBDSM package. It serves as the model accessor in the framework. It enables CRUD operations and additional capabilities like saving or loading a model. It provides an event-based notification for interacting with external packages. It consists of two main components. The components are the SysMLProject object and the SysMLEventManager service. The package diagram for SMM is depicted in Figure 11.

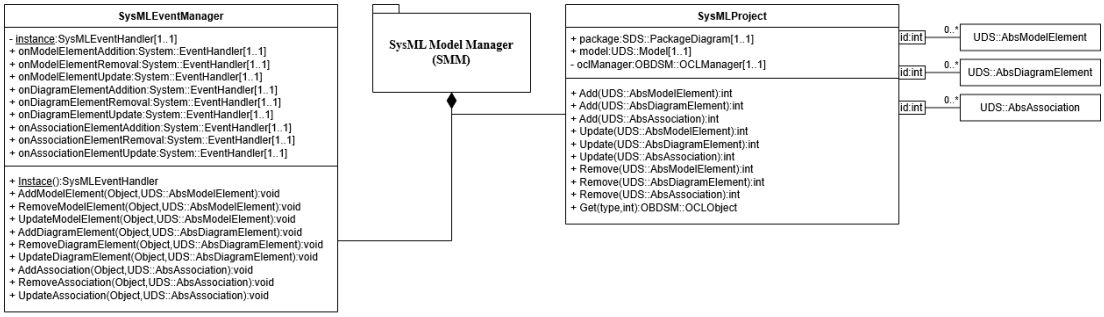


Figure 11 – Package Diagram of SMM.

The SysMLProject is an object that separates the application-level context from the project-level context the user is working on. It directs CRUD operations to the OBDSM and stores the related elements in its lists. Upon instantiation, it creates an empty model with a single empty package diagram. Any form of modification to the project triggers SysMLEventManager to invoke certain events. SysMLEventManager enables external packages to adjust any changes caused by any packages through an event listener pattern. Each addition, deletion, or update event contains the related OCL object so that the listeners can process it.

3.3.4 User Interface

The user interface (UI) for the prototypes consists of six sections. These sections are the top bar, the action bar, the model summary panel, the diagram summary panel, the dynamic

window area, and the primary display area. All UI elements of the prototypes share a familiar look and feel achieved by using the prefabricated materials of MRTK2. The UI of MRSysML in perspective and isometric views and 2DSysML are shown in Figure 12, Figure 13, and Figure 14, respectively.

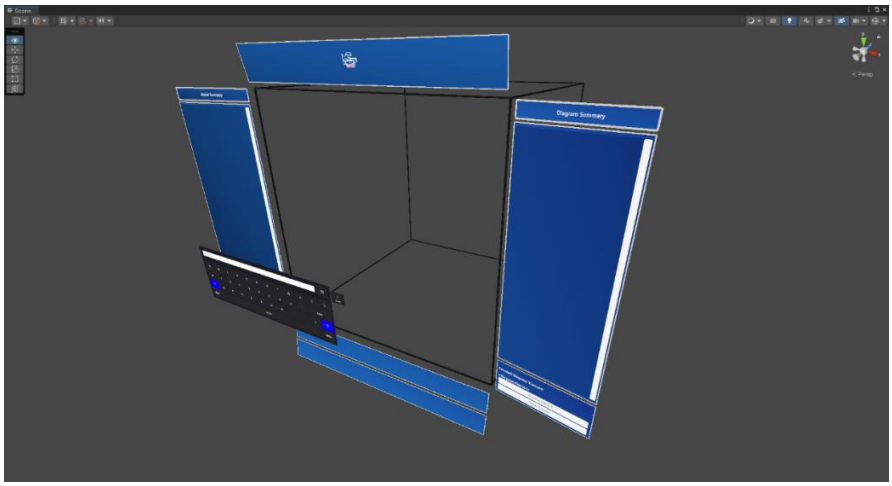


Figure 12 – MRSysML user interface in perspective view with non-native keyboard.

Figure 13 – MRSysML user interface in isometric view with non-native keyboard.

Figure 14 – User interface of 2DSysML with new comment window.

The top bar is responsible for displaying the title of the loaded diagram. For MRSysML, it also acts as an anchor bar to move the display around, zoom in, and zoom out. For 2DSysML, it serves no additional purpose.

The action bar is located at the bottom of the display and has options to add certain diagram elements to the diagram, either by defining a new one or importing an already existing diagram element.

The model summary panel is on the left side of the view, allowing the user to see a list of all elements in the model. The user can choose to delete a model element from the model or navigate to a specific diagram from this panel. The diagram summary panel is located on the right side of the panel and shows the loaded diagram and the included diagram elements. The user can choose to delete any diagram element or reset the location of the diagram element. On the summary panels, the user can see each object's label and unique identifier. Upon removal of any model element, its related diagram elements are automatically removed.

The primary display area is where the diagram is displayed. In MRSysML, it is contained in a box with thin black lines. There are invisible walls preventing the diagram elements from leaving the display area.

The dynamic window area is a hidden frame where opened windows are shown. In MRSysML, the dynamic window area follows the VR or MR device's location and orientation, showing in front of the user whenever a window is displayed. In 2DSysML, it is located in the middle of the view. The user can drag and drop the window to move it around. Upon opening or reopening any window, its view is reset back to the middle of the view. MRSysML includes a non-native virtual keyboard modified from the MRTK2 non-native keyboard. The keyboard is displayed in place of any dynamic window when any text field is selected.

3.3.5 Visualization Adapters

MRVA and 2DVA are designed to integrate the data into the user interface. It consists of seven components. These components are the visualization event manager service, the project loader behavior, the model summary behavior, the diagram summary behavior, the window behavior subpackage, the UI element behavior subpackage, and the diagram visualization subpackage. UI element behavior subpackage, diagram visualization subpackage, and window behavior subpackage are customized to fit the unique needs of MRVA and 2DVA. The abstract package diagram for visualization adapters is depicted in Figure 15.

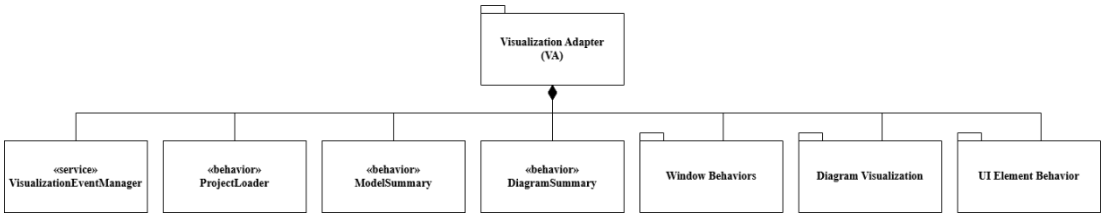


Figure 15 – Abstract Package Diagram of MRVA and 2DVA. The objects with behavior classifier implement Unity Engine's MonoBehavior class.

MRVA and 2DVA also contain MRTK2 profiles, prefabricated objects for each UI element, prefabricated objects for each window, and prefabricated displays for each diagram element.

VisualizationEventManager service serves as the primary middleware for all components of the MRVA and 2DVA. It employs the singleton pattern so all services can access the same instance. It contains events categorized as requests that are invoked with respect to user actions

and other events that are invoked based on computation. The class information of VisualizationEventManager as a diagram element is shown in Figure 16.

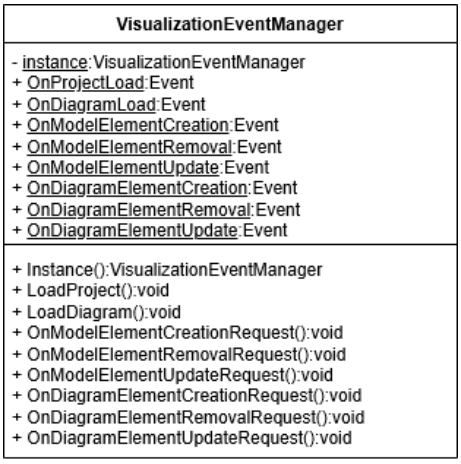


Figure 16 – Diagram Element for VisualizationEventManager.

ProjectLoader behavior extends Unity Engine's monobehavior and manages SSM's SysMLProject. It converts the user's actions into SysMLProject object's functions. During instantiation, a new SysMLProject is initialized. Upon successful initialization, an OnProjectLoad event is invoked. Afterward, the primary model's package diagram is loaded, and an OnDiagramLoad event is invoked. Whenever a new SysMLProject is loaded or initialized or when a different diagram is loaded, these events are invoked, respectively. The user's actions are received as invoked events. The types of listened requests are model element creation or removal requests and diagram element creation or deletion requests. ProjectLoader uses a factory method pattern to create elements. It is a factory for individual model elements, associations, and diagram elements. Upon invocation of a creation request, the ProjectLoader initializes the specified model element and adds it to the SysMLProject. Upon invocation of a deletion request event, the ProjectLoader calls the removal of the specified element from the SysMLProject. ProjectLoader does not invoke any response events since SSM invokes specific events for any addition, update, or removal. The class diagram for ProjectLoader is shown in Figure 17.

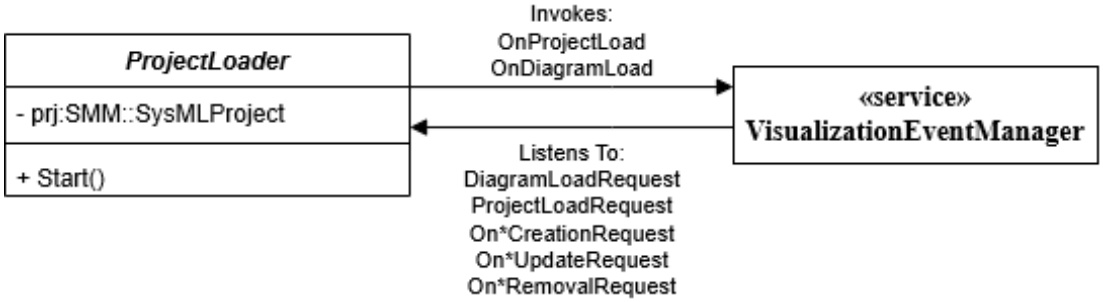


Figure 17 – Class Diagram of ProjectLoader.

ModelSummary service and DiagramSummary service are the behavior classes for the model summary panel and diagram summary panel, respectively. They utilize the factory method pattern to instantiate new rows in the summary panel. The list rows are instantiated as buttons that, when pressed, invoke OnModelElementSelected and OnDiagramElementSelected events. When SSM invokes the OnModelElementAddition event, the ModelSummary service finds the owner model element in the list and adds the new element as its child element. When SSM invokes the OnDiagramElementAddition event, the DiagramSummary service checks if the owner diagram is the currently displayed diagram and adds the element to the list

accordingly. Upon invocation of OnModelElementRemoval or OnDiagramElementRemoval, the list rows are removed. This service also listens to the OnDiagramLoaded event and refreshes the list based on the new diagram when invoked. The summary services can also trigger deletion request events based on the most recent element selected. Class diagrams of ModelSummary and DiagramSummary are displayed in Figure 18.

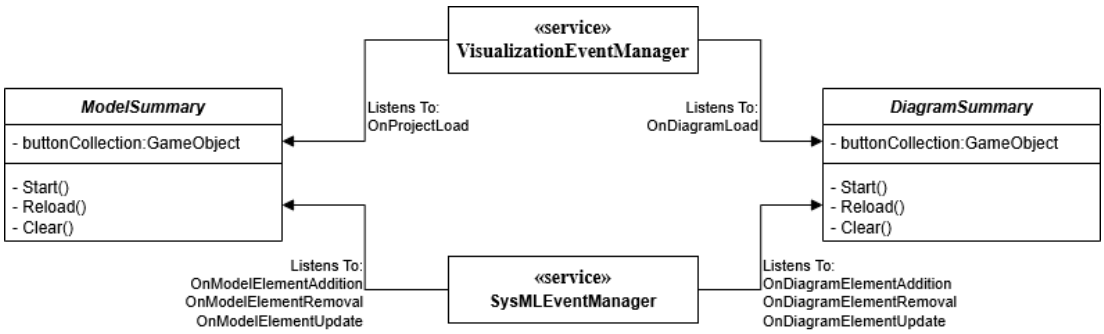


Figure 18 – Class Diagram of ModelSummary and DiagramSummary with its relationship lines to Event Managers.

The window behavior subpackage contains the behaviors for each dynamic window and the non-native virtual keyboard. Each model element has unique dynamic windows that contain the input fields for the attributes of each element. The dynamic windows have two standardized buttons to apply or cancel the operation. A new element creation request event is invoked after pressing the apply button. Upon selection of the cancel button, the window is hidden. When the window is reopened, all of its fields are cleared.

The Diagram Visualization subpackage contains the required behaviors for the primary display area, the top bar, and the individual diagram element displays. The primary display area behavior listens to the OnDiagramLoad event, destroying the previous loaded diagram element displays and instantiates new displays for all owned diagram elements of the loaded diagram. Furthermore, it listens to the OnDiagramElementAddition and OnDiagramElementRemoval events to instantiate or destroy the diagram element displays. The process is only applicable if the owner diagram for the diagram element is the currently displayed diagram. The top bar behavior listens to the OnDiagramLoad event and displays the title for the loaded diagram accordingly.

CHAPTER 4

EVALUATION PROCESS

4.1 User Tests

User test sessions are conducted to investigate the advantages and disadvantages of VR or MR environments compared to the desktop environment. Each participant is requested to perform a simple scenario to design certain aspects and capabilities of a calculator as a system of interest. An example of a calculator is selected to ensure all participants are familiar with it. The scenario is performed in both environments, and the users are asked to fill in multiple standard questionnaires and answer some open-ended questions.

4.1.1 Participant Selection

The participants are divided into two groups. The first group is people with experience working as or with systems engineers. They are referred to as the Systems Engineering (SE) focus group. The second group is people who study or work in developing interactive applications and video games. They are called interactive application and game development (IAGD) focus groups. The participants were invited to the user tests and their attendance was on a voluntary basis. Thirty participants attended the user tests, with an equal number of participants from both focus groups.

4.1.2 User Test Procedure

The test scenario consisted of designing four diagrams: 1) A package diagram that defines the components of the calculator, 2) A block definition diagram that defines the attributes of the calculator, 3) A use case diagram that shows the primary use cases of a calculator and its actor, and 4) An activity diagram that defines the algorithm of calculating expression. These diagrams are selected so that the user can interact with the prototype to design both the structural and behavioral sides of a system. After the execution of the scenario, the users were allowed to freely explore additional features of the prototypes. Each participant had a 45-minute time limit for their session, excluding the time to fill in the questionnaires, and only one participant was attending at a time. The set scenario in both environments took 15 to 25 minutes in total, based on the performance of the participant.

4.1.3 Data Collection

The participants are asked to fill in four standard questionnaires and respond to four open-ended questions. These questionnaires are the Immersive Tendencies Questionnaire (ITQ) [37] [38], the Presence Questionnaire (PQ) [37] [39], the System Usability Scale (SUS) [40] and the User Experience Questionnaire (UEQ) [41] Each questionnaire attempts to evaluate a different aspect of the application through responses within itself or correlation with each other. Scores in the standard questionnaires are compared regarding the two focus groups and based on the environments to find any statistically significant differences.

4.2 Standard Questionnaires

Each questionnaire attempts to identify a different attribute of the prototype. Each questionnaire has different evaluation methods and scoring methods.

4.2.1 Immersive Tendencies Questionnaire (ITQ)

Immersive Tendencies Questionnaire (ITQ) [37] [38] is a questionnaire that attempts to identify the participants' responses to immersive media. The immersive media can be something watched similar to a movie, something played such as a video game, something read like a novel, or anything that keeps the person's attention. It scores the participant in four main factors. The factors are focus, involvement, emotion, and game. Focus measures how easily the participant focuses on the instrument and disconnects from what is happening around them. Involvement weighs how likely the participant feels as if they are part of what is going on in the environment. Emotion estimates how emotional the person reacts to the events in the media. Games assess how inclined the person is to make a game out of their situation and increase the enjoyability or competitiveness of the task.

ITQ was initially designed by Witmer and Singer [37]. UQO Cyberpsychology Lab [38] have simplified the original questionnaire from 29 to 18 questions by removing redundant ones and updating the scoring of each attribute accordingly. The participants answer questions with a score ranging from one to seven, where each question focuses on one factor, and the score in each factor is calculated as follows:

$$Q_N = Score\ of\ the\ question\ number\ N \quad (1)$$

$$Focus = Average(Q_1, Q_2, Q_3, Q_4, Q_5) \quad (2)$$

$$Involvement = Average(Q_4, Q_5, Q_{10}, Q_{12}, Q_{18}) \quad (3)$$

$$Emotion = Average(Q_{11}, Q_{15}, Q_{16}, Q_{17}) \quad (4)$$

$$Games = Average(Q_6, Q_9, Q_{14}) \quad (5)$$

The participants are asked to fill in the ITQ based on themselves and not the prototype. Its results are not used to evaluate the prototype but are used to identify if any statistically significant difference exists between the two focus groups and attempt to find any correlation between these factors and other attributes of the framework or the prototype.

4.2.2 Presence Questionnaire (PQ)

Presence Questionnaire (PQ) is another questionnaire designed by Wither and Singer [37]. The main focus of this questionnaire is to investigate the sense of presence in a virtual environment. The questionnaire is conducted in reference to any specific VR experience. For the study, it is utilized to understand the strengths and weaknesses of the MRSysML prototype's virtual environment variant.

PQ includes 32 questions, which the participants answer with a score from one to seven. The original design suggested computing seven unique factors to evaluate the presence, but Witmer et al. [39] have discovered that the correlation deemed some factors unnecessary and assessed using four distinct factors is more beneficial. These factors are involvement, sensory fidelity, adaptation and immersion, and interface quality. The scores of each factor are calculated as follows:

$$Q_N = Score\ of\ the\ question\ number\ N \quad (6)$$

$$Involvement = Average(Q_1, Q_2, Q_3, Q_4, Q_6, Q_7, Q_8, Q_{10}, Q_{14}, Q_{17}, Q_{18}, Q_{29}) \quad (7)$$

$$\text{Sensory Fidelity} = Average(Q_5, Q_{11}, Q_{12}, Q_{13}, Q_{15}, Q_{16}) \quad (8)$$

$$Adaptation\ \&\ Immersion = Average(Q_9, Q_{20}, Q_{21}, Q_{24}, Q_{25}, Q_{30}, Q_{31}, Q_{32}) \quad (9)$$

$$Interface\ Quality = Average(Q_{19}, Q_{22}, Q_{23}) \quad (10)$$

Involvement is how inclusive the virtual environment feels to the user. Sensory fidelity evaluates the precision of their senses regarding the events happening and their actions. Adaptation and immersion determine how realistic the environment feels and how hard it is to adapt to the virtual experience. Interface quality weighs how intuitive the user interface is.

The participants are asked to complete the questionnaire based solely on their experience using MRSysML in a VR environment. It is used as a general evaluation of the experience and in correlation with the System Usability Scale. The result is also evaluated to see if there is a statistically significant difference in the experience of the focus groups on any factor.

4.2.3 System Usability Scale

System Usability Scale (SUS) [40] is a questionnaire designed by Brooke that estimates the usability of a system in relation to its applicability, understandability, and ease of use. It consists of 10 statements with a score of one to five, where one is equivalent to strongly disagree, and five is equivalent to strongly agree. The consecutive questions alternate positive and negative statements starting with positive. The responses are used to determine the single factor of usability. The formulas to calculate the usability score ranging from 0 (worst) to 100 (best) are as follows:

$$Q_N = Score\ of\ the\ question\ number\ N \tag{11}$$

$$S^+ = Positive\ Statement\ Score = Q_1 + Q_3 + Q_5 + Q_7 + Q_9 \tag{12}$$

$$S^- = Negative\ Statement\ Score = Q_2 + Q_4 + Q_6 + Q_8 + Q_{10} \tag{13}$$

$$SUS\ Score = (20 + S^+ - S^-) \times 2.5 \tag{14}$$

The participants are asked to fill out the SUS questionnaire regarding MRSysML and 2DSysML. A comparative analysis compares the focus groups' scores and two environments. Grading and ratings in accordance with Sauro and Lewis [42], and Bangor et al. [43] [44] are applied to measure the usability of the prototypes on a generalized scale.

The most recent commonly used grading system for SUS score is the Curved Grading Scale (CGS), defined by Sauro and Lewis [42]. They studied hundreds of studies with thousands of individual responses to calculate the CGS. The CGS follows the common grading scale with letter grades F, D, C, B, and A with plus or minus for intermediate grades. The grading scale and their percentile ranges are listed in Table 2.

Table 2 – Grades of SUS by CGS and percentile of scores by Sauro and Lewis.

Grade	Threshold	Percentile
A+	84.1-100	96-100
A	80.8-84.0	90-95
A-	78.9-80.7	85-89
B+	77.2-78.8	80-84
B	74.1-77.1	70-79
B-	72.6-74.0	65-69
C+	71.1-72.5	60-64
C	65.0-71.0	41-59
C-	62.7-64.9	35-40
D	51.7-62.6	15-34
F	0.0-51.7	0-14

In their early work, Bangor et al. studied over 2,300 SUS surveys [43] to define acceptability ratings and around 1,000 SUS surveys [44] to give ratings to different scores. They compared the surveys in environments similar to web interface applications, desktop applications, and

cellphone equipment. They separate acceptability ratings using quartile ranges of the results into not acceptable range, marginal low range, marginal high range, and acceptable range, with marginal meaning the product requires improvements to be satisfactory. The separation of low and high marginal ranges is done at the end of the first quartile range, with an SUS score of 50. They have also defined adjective ratings as worst imaginable, awful, poor, OK, good, excellent, and best imaginable. All ratings except the worst imaginable and awful are found to be significantly different, and awful ratings joined into the worst imaginable rating. The acceptability ranges and adjective ratings are quantified in Table 3 and Table 4.

Table 3 – Acceptability rating ranges defined by Bangor et al. [43].

Acceptability Rating	Range
Acceptable	70-100
Marginal (High)	60-70
Marginal (Low)	50-60
Not Acceptable	0-50

Table 4 – Adjective rating mean values and standard deviations defined by Bangor et al. [44] based on the number of survey results with that rating.

Adjective Rating	Survey Count	Mean Value	Standard Deviation
Best imaginable	16	90.9	13.4
Excellent	289	85.5	10.4
Good	345	71.4	11.6
OK	211	50.9	13.8
Poor	72	35.7	12.6
Awful	22	20.3	11.3
Worst imaginable	4	12.5	13.1

4.2.4 User Experience Questionnaire

The User Experience Questionnaire (UEQ) is designed by Laugwitz et al. [41] as a way of determining the user's experience when using a product, it contains a total of 26 attribute pairs, each including two ends of a measurement. Three examples of such pairs are annoying and enjoyable, complicated, and easy or conservative and innovative. Attractiveness is an overall scale that determines whether the user has liked the product. Perspicuity is how quickly the user adapted to the product and learned to use it. Efficiency is how easy it is to use the product. Dependability is the measurement of the sense of control over the product. Stimulation is how exciting and enjoyable the product is to use. Novelty determines the uniqueness of the experience. Each attribute is assigned to one scale, and the score on each scale is determined based on these attributes alone. The number of attributes per scale is summarized in Table 5.

Table 5 – Distribution of UEQ attributes per scale in accordance with Laugwitz et al. [41].

Attractiveness	Perspicuity	Efficiency	Dependability	Stimulation	Novelty
6	4	4	4	4	4

Schrepp et al. [45] defined a benchmark for grading the attributes. To calculate a score for each scale, they designed a matrix transform that would change the value from a range of one to seven to a range of negative three to positive three. They also flipped the scores if the left side of the pair had a positive meaning instead of the right side. The final score of each scale is calculated by averaging the score of each related attribute. Schrepp et al. gathered a dataset of 21,175 responses from 468 unique studies to calculate percentile thresholds for each scale. They have defined six ranges and assigned adjective grades for each range. These grades are

bad, below average, above average, good, and excellent. Their findings also suggest that while the possible range of scores varied from negative three to positive three, in the application, the scores would be in the range from negative one to positive two and a half. The grades, percentiles, and ranges for each scale are provided in Table 6.

Table 6 – Grades, percentiles, and ranges for each scale in the benchmark by Schrepp et al. [45].

Grade	Percentile	Attractiveness	Perspicuity	Efficiency	Dependability	Stimulation	Novelty
Excellent	90-100	> 1.84	> 2.00	> 1.88	> 1.70	> 1.70	> 1.60
Good	75-90	1.58-1.84	1.73-2.00	1.50-1.88	1.48-1.70	1.35-1.70	1.12-1.60
Above Avg.	50-75	1.18-1.58	1.20-1.73	1.05-1.50	1.14-1.48	1.00-1.35	0.70-1.12
Below Avg.	25-50	0.69-1.18	0.72-1.20	0.60-1.05	0.78-1.14	0.50-1.00	0.16-0.70
Bad	0-25	< 0.69	< 0.72	< 0.60	< 0.78	< 0.50	< 0.16

The participants are asked to fill in UEQ twice, once for MRSysML and again for 2DSysML. The results from both focus groups and for both prototypes are compared to find any statistically significant difference.

4.3 Open-Ended Questions

The participants are asked to answer four open-ended questions that encapsulate the broad experience they had. The first question asked for the strong and improvable areas of applying MBSE in a VR or MR environment. The second question asked which environment, VR, MR, or Desktop, they would prefer to work in and why. The third question asked which input system (keyboard, mouse, controller, or hand interaction) they preferred to work with and why. The last question asked what improvements they would like to see in such a prototype or framework. The answers gathered are used in correlation with the standard questionnaires and as possible groundwork for future improvements.

4.4 Determining Statistical Significance

To evaluate the prototype and framework using the scores from the standard questionnaires, it is crucial to understand which data gathered are different in comparison. Statistical significance allows the researchers to determine if the data are different enough to be significant where the data are consistent with the null hypothesis. The consistency of the data is measured using the p-value with 0.05 as the threshold for statistical significance. Since the data gathered are not normally distributed, usage of parametric tests such as independent sample t-test and paired samples t-test are not applicable. Wilcoxon Signed-Rank Test [46] is used for dependent datasets and Mann-Whitney U-Test [47] is used for independent datasets are applied. The non-parametric tests result in Z-scores, which are converted to p-values. The critical threshold of the Z-score for a two-tailed p-value of 0.05 is roughly 1.96. Any absolute Z-score above this threshold is considered statistically significant.

For this thesis, the calculations for statistical significance are done using the Social Science Statistics website's pre-defined Wilcoxon Signed Rank Test Calculator [48] and Mann-Whitney U Test Calculator [49].

4.4.1 Wilcoxon Signed-Rank Test

Wilcoxon Signed-Rank Test, also known as Wilcoxon Test or Wilcoxon T-Test, is proposed by Frank Wilcoxon in 1945 [46]. This non-parametric test is used to find significant differences in dependent samples from two datasets that are not normally distributed. For

example, participants' SUS scores for MRSysML and 2DSysML are compared as paired samples. The absolute value of the difference in each pair is ranked starting with one and going up to the number of paired samples. The difference between the total of positive and negative ranks should satisfy the null hypothesis. The Z-score is calculated using the following formulas. The critical W value for a Z-score above 1.96 with 15 participants is 25, and with 30 participants, it is 137.

$$n = Number\ of\ Paired\ Samples \tag{15}$$

$$t = Number\ of\ Ranks\ with\ Multiple\ Samples \tag{16}$$

$$c_i = Number\ of\ Samples\ at\ Rank\ i \tag{17}$$

$$R_+ = Sum\ of\ Positive\ Ranks \tag{18}$$

$$R_- = Sum\ of\ Negative\ Ranks \tag{19}$$

$$W = min(R_+, R_-) \tag{20}$$

$$Expected\ Value = \mu_W = \frac{n \cdot (n+1)}{4} \tag{21}$$

$$Standard\ Deviation = \sigma_W = \sqrt{\frac{n \cdot (n+1) \cdot (2n+1) - \sum_{i=1}^{k} \frac{{t_i}^3 - t_i}{2}}{24}} \tag{22}$$

$$Z = \frac{W - \mu_W}{\sigma_W} \tag{23}$$

4.4.2 Mann-Whitney U-Test

Mann-Whitney U-Test, also known as Mann-Whitney-Wilcoxon, is proposed by Henry Mann and Donald Ransom Whitney in 1947 [47]. This non-parametric test is used to find significant differences in two non-dependent datasets with equal or unequal number of samples measuring the same score. This test aims to calculate a U-value based on the number of samples and the sum of the rank for the dataset. The Z-score is calculated based on the lesser U-value, expected value, and standard deviation of the joint dataset. For example, focus groups' PQ scores are compared using this test. The Z-score is calculated using the following formulas. The critical U value for a Z-score above 1.96 with 15 participants is 64.

$$n_1 = Number\ of\ Samples\ from\ First\ Dataset \tag{24}$$

$$n_2 = Number\ of\ Samples\ from\ second\ Dataset \tag{25}$$

$$R_1 = Sum\ of\ Ranks\ for\ First\ Dataset \tag{26}$$

$$R_2 = Sum\ of\ Ranks\ for\ Second\ Dataset \tag{27}$$

$$U_1 = n_1 \cdot n_2 + \frac{n_1 \cdot (n_1+1)}{2} - R_1 \tag{28}$$

$$U_2 = n_1 \cdot n_2 + \frac{n_2 \cdot (n_2+1)}{2} - R_2 \tag{29}$$

$$U = \min(U_1, U_2) \tag{30}$$

$$Expected\ Value = \mu_U = \frac{n_1 \cdot n_2}{2} \tag{31}$$

$$Standard\ Deviation = \sigma_U = \sqrt{\frac{n_1 \cdot n_2 \cdot (n_1 + n_2 + 1)}{12}} \tag{32}$$

$$Z = \frac{U - \mu_U}{\sigma_U} \tag{33}$$

CHAPTER 5

RESULTS

5.1 Immersive Tendencies Questionnaire Results

The data collected from SE and IAGD focus groups are depicted as box plots in Figure 19. Since the results from both groups are independent and the distribution is not a normal distribution, the results are compared using the Mann-Whitney U Test to find any statistically significant difference. The statistical information is shown in Table 7.

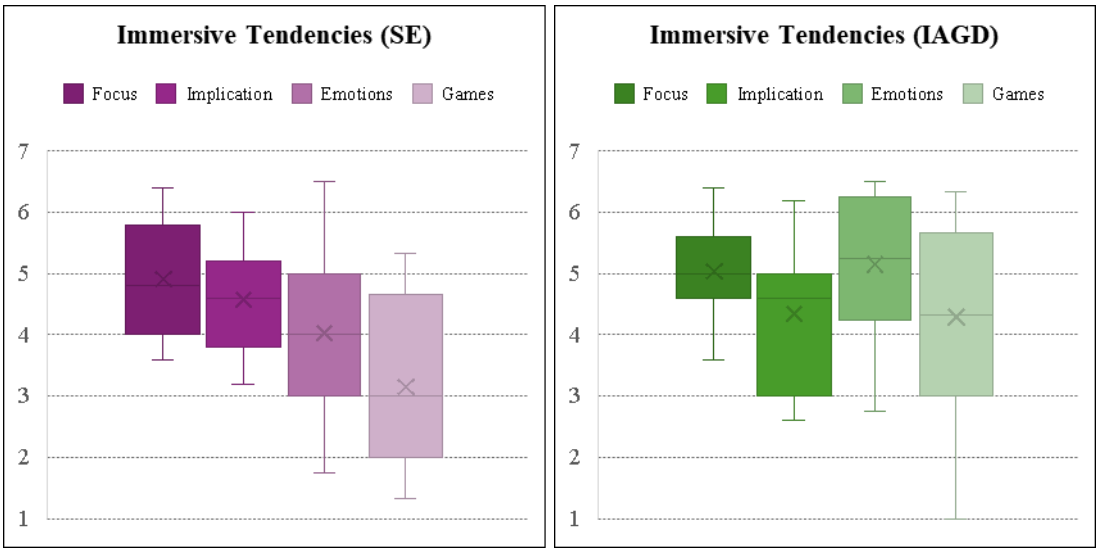


Figure 19 – Box plots of ITQ results from both focus groups are shown side by side, with the Systems Engineering focus group on left and interactive applications and game development focus group on the right.

Table 7 – Mean Value and Mann-Whitney U Test results for immersive tendencies questionnaire comparing systems engineering focus group and interactive applications and game development focus group.

Factor	Mean (SE)	Mean (IAGD)	U value	Z-score	p-value
Focus	4.91	5.04	101.5	0.436	0.660
Implication	4.57	4.35	101.0	0.456	0.646
Emotion	4.03	5.15	61.0	2.115	0.034
Games	3.16	4.29	63.0	2.032	0.042

The differences and variances of focus and implication factors have Z-scores below 1.96 and p-values above 0.05, which indicates that the differences between these factors are not statistically significant. However, the differences in the emotion and games factors are statistically significant.

5.2 Presence Questionnaire Results

SE and IAGD focus groups' responses to the PQ regarding MRSysML in the VR environment are depicted as box plots in Figure 20. Since the results from both groups are independent and the distribution is not a normal distribution, the results are compared using the Mann-Whitney

U Test to find any statistically significant difference. The statistical information is shown in Table 8.

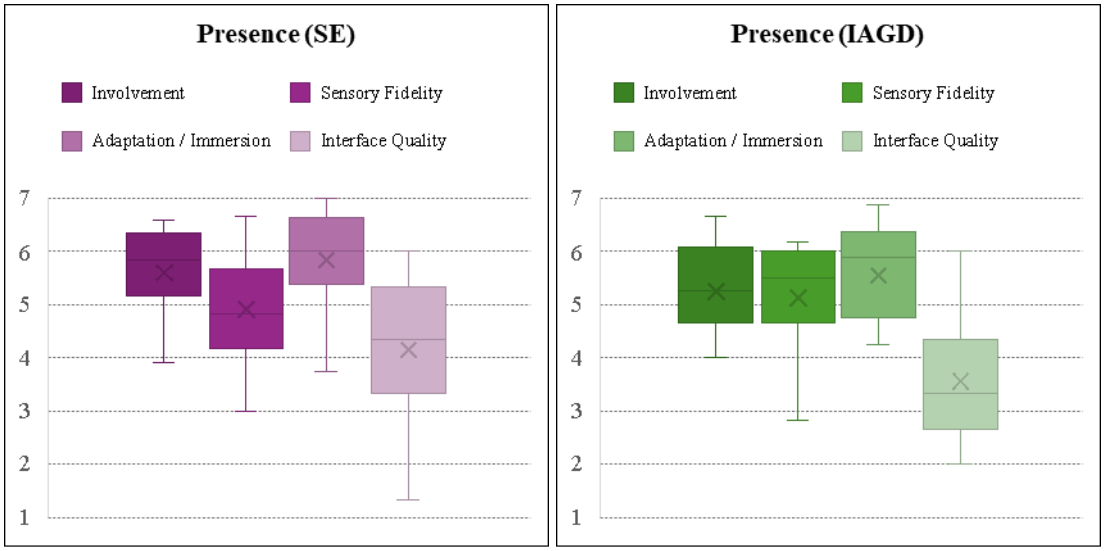


Figure 20 – Box plots of PQ results from both focus groups shown side by side, systems engineering focus group on left and interactive applications and game development on the right.

Table 8 – Mean Value and Mann-Whitney U Test results for presence questionnaire comparing systems engineering focus group and interactive applications and game development focus group.

Factor	Mean (SE)	Mean (IAGD)	U value	Z-score	p-value
Involvement	5.60	5.24	81.5	1.265	0.204
Sensory Fidelity	4.91	5.12	89.5	0.933	0.352
Adaptation / Immersion	5.83	5.55	93.0	0.788	0.430
Interface Quality	4.16	3.56	78.0	1.410	0.159

The differences and variances of all factors have Z-scores below 1.96 and p-values above 0.05, indicating that none are statistically significant.

5.3 System Usability Scale Results

The system usability scale (SUS) scores are put through comparative analysis in two separate ways. Firstly, the scores of both groups are compared. Since the distribution is not a normal distribution, and the two groups are independent, the Mann-Whitney U Test is applied to find any statistically significant difference and variance. Secondly, each focus group and a joint dataset of both groups are compared regarding MRSysML and 2DSysML. While the distribution is not a normal distribution, the samples are paired based on the participant. The Wilcoxon Signed-Rank Test is applied to investigate statistically significant differences and variances.

The mean values of either focus group's SUS score and the joint datasets' SUS scores are compared within the scope of the CGS by Sauro and Lewis [42] and acceptability and adjective ratings by Bangor et al. [43] [44] to put a generalized perspective on the usability of the prototypes and framework.

The SUS scores by SE focus group, IAGD focus group and joint dataset for MRSysML and 2DSysML prototypes are depicted in Figure 21.

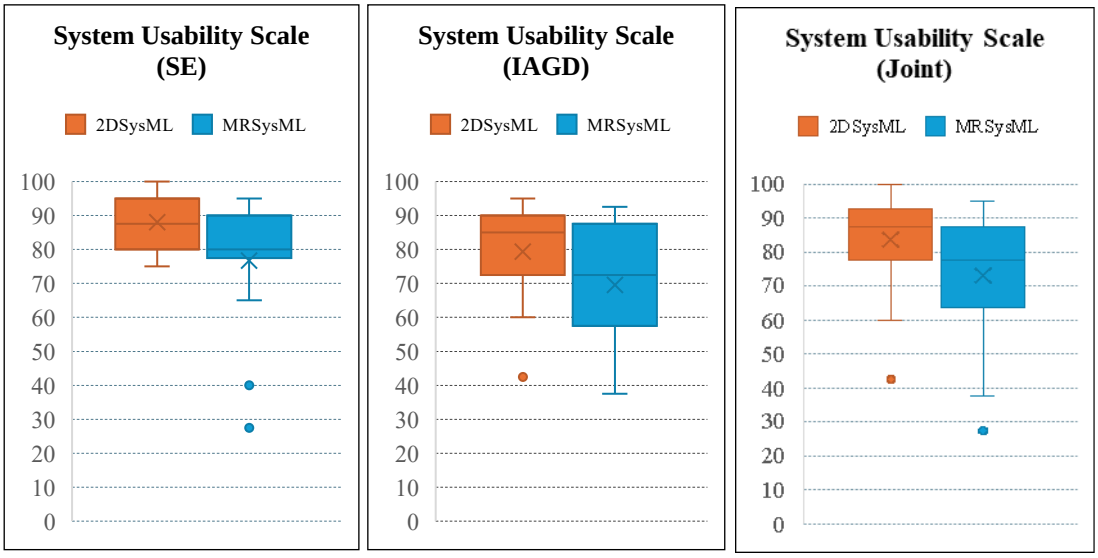


Figure 21 – Box plots of SUS scores for either group and joint dataset, systems engineering focus group on left and interactive applications and game development in the middle and joint dataset on the right.

The SUS score comparison based on focus groups is investigated using the Mann-Whitney U Test since the focus groups are independent, and the scores are not of a normal distribution. The mean values per focus group and p-value for both prototypes are listed in Table 9.

Table 9 – Mean Value and Mann-Whitney U Test results for SUS scores comparing systems engineering focus group and interactive applications and game development focus group.

Prototype	Mean (SE)	Mean (IAGD)	U value	Z-score	p-value
2DSysML	88.0	79.3	71.0	1.701	0.089
MRSysML	76.7	69.5	71.5	1.680	0.092

SUS scores by both groups show a statistically significant difference and variance in favor of 2DSysML. The mean difference in scores is 11.3 for the SE focus group and 9.8 for the IAGD focus group.

The SUS score comparison based on prototypes is evaluated using the Wilcoxon Signed-Rank Test since the scores for either prototype can be paired based on the participant, and the scores are not of a normal distribution. The mean value per prototype and p-value for each focus group and joint dataset are listed in Table 10.

Table 10 – Mean Value and Wilcoxon Signed-Rank Test results for SUS score for prototypes by systems engineering focus group, interactive applications and game development focus group, and joint dataset.

Dataset	Mean (2DSysML)	Mean (MRSysML)	W value	Z-score	p-value
SE	88.0	76.7	14.0	2.612	0.009
IAGD	79.3	69.5	20.5	2.243	0.025
Joint	83.7	73.1	67.5	3.393	0.001

SUS score difference and variance for both prototypes are statistically significant independent of the focus group.

Based on the SUS score CGS by Sauro and Lewis [42], the MRSysML is a B- grade tool within the 65-69 percentile. However, within the IAGD focus group's perspective, it is lowered to a C grade tool, and within the SE focus group's perspective, it is increased to a B

grade. On the other hand, 2DSysML is an A grade tool according to the joint dataset, A+ for the SE focus group and A- according to the IAGD focus group. The acceptability rating, according to Bangor et al. [43], is Acceptable for all datasets and prototypes except for MRSysML based on the IAGD focus group, which has a mean value of the High Marginal range. The adjective rate, according to Bangor et al. [44], is rated closest to Excellent for the 2DSysML prototype and rated closest to Good for MRSysML. The detailed graph of the grades and ratings are shown in Figure 22 and Figure 23, respectively. A summary is listed in Table 11.

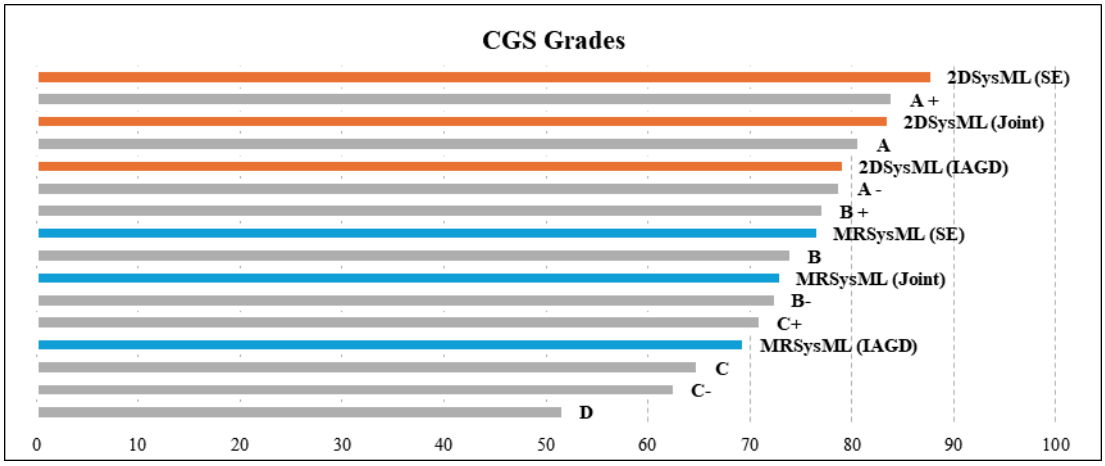


Figure 22 – Mean values of grades in accordance with Sauro and Lewis' CGS [42] and the mean values of each prototype and dataset.

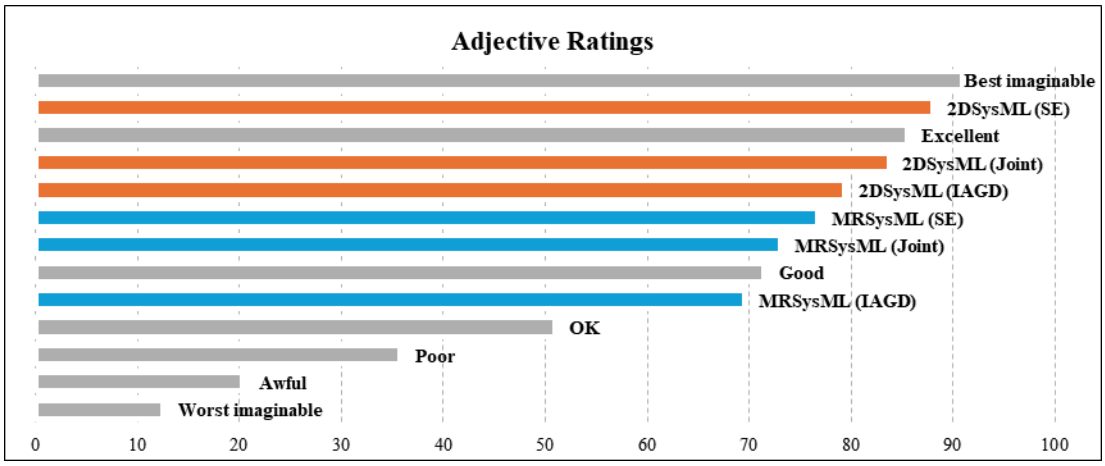


Figure 23 – Mean values adjective ratings in accordance with Bangor et al. [44] and the mean values of each prototype by dataset.

Table 11 – 2DSysML and MRSysML mean value grades according to CGS by Sauro and Lewis [42] and acceptability and adjective ratings, according to Bangor et al. [43] [44].

Dataset	2DSysML			MRSysML		
	Acceptability	**Adjective**	**CGS Grade**	**Acceptability**	**Adjective**	**CGS Grade**
SE	Acceptable	Excellent	A+	Acceptable	Good	B
IAGD	Acceptable	Good	A-	Marginal	Good	C
Joint	Acceptable	Excellent	A	Acceptable	Good	B-

5.4 User Experience Questionnaire Results

The benchmark graphics of prototypes by different focus groups are provided in Figure 24, Figure 25, Figure 26 and Figure 27, respectively.

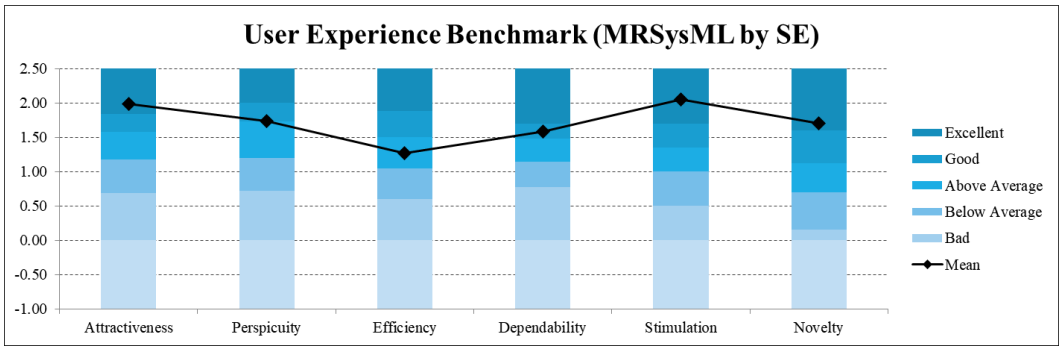


Figure 24 – UEQ benchmark [45] of MRSysML based on responses of the SE focus group.

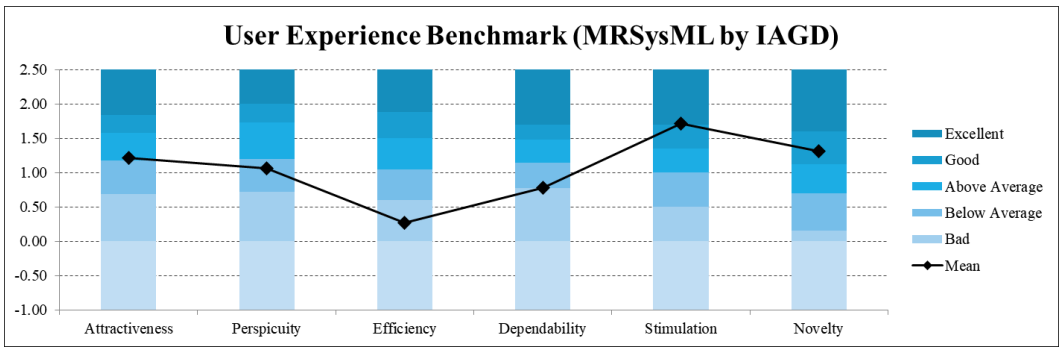


Figure 25 – UEQ benchmark [45] of MRSysML based on responses of the IAGD focus group.

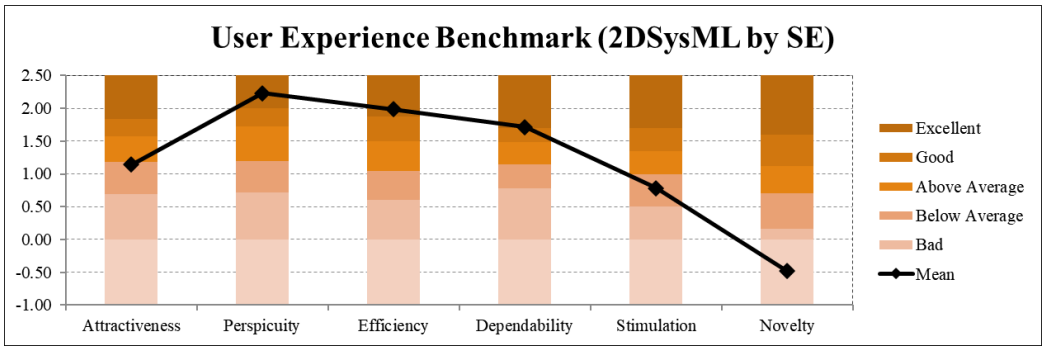


Figure 26 – UEQ benchmark [45] of 2DSysML based on responses of the SE focus group.

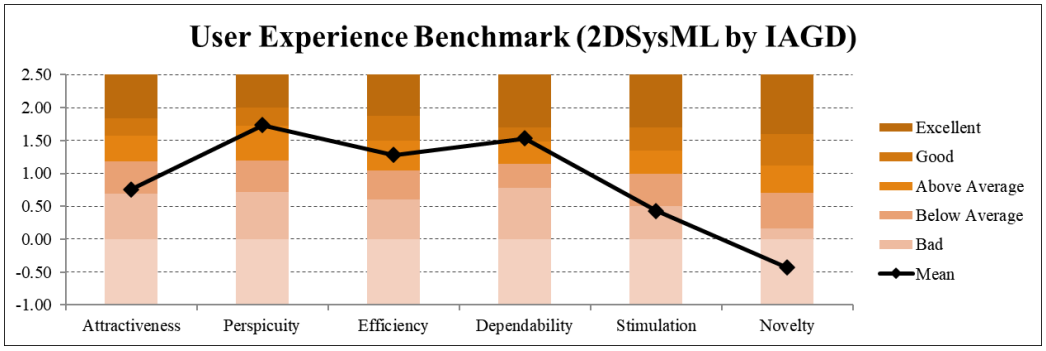


Figure 27 – UEQ benchmark [45] of 2DSysML based on responses of the IAGD focus group.

The UEQ scores for each scale differ based on the prototype and focus group. The scores and ratings in each factor are summarized in Table 12. For both focus groups, 2DSysML is stronger on the Perspicuity, Efficiency, and Dependability scales, while MRSysML is higher on the Attractiveness, Stimulation, and Novelty scales.

Table 12 – Summary of UEQ benchmark [45] scores and ratings by prototype and focus group.

Attribute	2DSysML		MRSysML	
	SE	IAGD	SE	IAGD
Attractiveness	1.14 (Below Avg.)	0.76 (Below Avg.)	1.99 (Excellent)	1.21 (Above Avg.)
Perspicuity	2.23 (Excellent)	1.73 (Good)	1.73 (Good)	1.07 (Below Avg.)
Efficiency	1.98 (Excellent)	1.28 (Above Avg.)	1.27 (Above Avg.)	0.27 (Bad)
Dependability	1.72 (Excellent)	1.53 (Good)	1.58 (Good)	0.78 (Below Avg.)
Stimulation	0.78 (Below Avg.)	0.43 (Bad)	2.05 (Excellent)	1.72 (Excellent)
Novelty	-0.48 (Bad)	-0.43 (Bad)	1.70 (Excellent)	1.32 (Good)

5.5 Open-Ended Questions Answers

5.5.1 Strong and Improvable Areas of Using VR and MR for MBSE

The most common strong area is enjoyability, where the user has found the scenario and uses the application to be enjoyable and wants to use the application further. The second most common strong area is immersion, where the users find the experience helping them focus and feel like a part of the environment. The third strong area is 3D viewing, where the users find the depth that allows them to differentiate shapes and edges of diagram elements more easily. The final strong area is intuitiveness, where the users feel similar to holding objects in real life and can perform the scenario without requiring external help.

Two areas are tied for the most common improvable area. Text input difficulty, where users find using the non-native keyboard difficult, and cumbersomeness of the general usage, where the user felt they had to take more steps than necessary to achieve any specific goals. The third most common improvable area is unfamiliarity, where the users are not experienced with using VR or MR devices and have to overcome a learning curve. The fourth improvable area is health concerns. The users hesitate to use the devices long-term due to possible ergonomic issues and face certain health troubles such as neck pain or headaches.

The strong areas had more average participant responses than the improvable areas. The number of responses for the most common strong and improvable areas are summarized in Table 13.

Table 13 – Strong and improvable areas of applying MBSE in VR and MR environments and the number of responses for each area.

Strong Areas	Improvable Areas
Enjoyability (12)	Cumbersomeness (11)
Immersion (11)	Text Input Difficulty (11)
3D View (10)	Unfamiliarity (8)
Intuitiveness (8)	Health Concerns (5)

5.5.2 Work Environment Preferences

When asked about the preferred work environment, VR, MR, or desktop, an equal number of participants answered that they preferred either environment. Four participants responded that

they preferred VR or MR for short-term work but the desktop environment for long-term work. These four participants are marked as “depends”. Two participants do not specify preference in either work environment. The graphic view of the work environment preferences is shown in Figure 28.

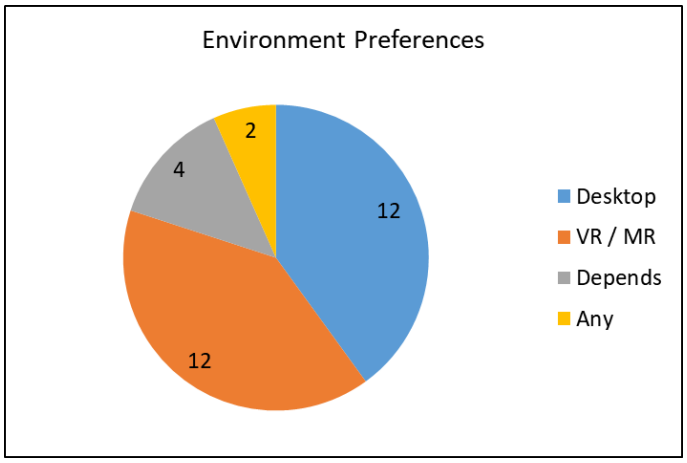


Figure 28 – Pie chart of participants' work environment preferences.

5.5.3 Interaction Device Preferences

Regarding the interaction devices–controllers, hand or keyboard, and a mouse–the participants favor the keyboard due to the cumbersomeness and unfamiliarity of a virtual keyboard. However, some explain that designing with a controller is more enjoyable and interactive compared to using a mouse. They have created their own category and suggested combining controllers with physical keyboards would be their preference. Only four of the participants preferred using their hands as the primary interaction method. The summary of the interaction device preferences is shown in Figure 29.

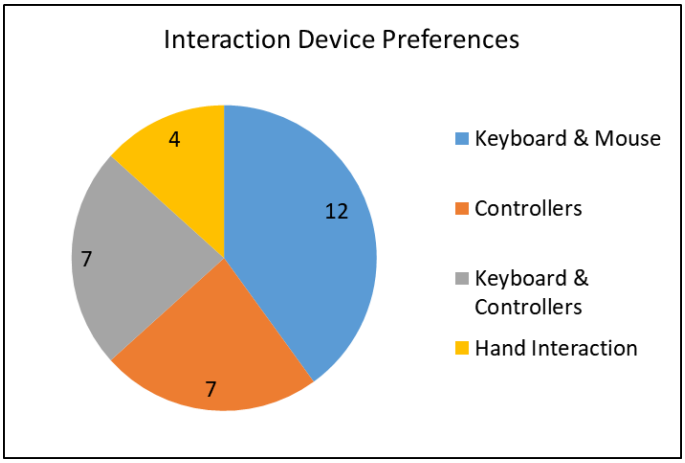


Figure 29 – Pie chart of interaction device preferences of participants.

5.5.4 Participants’ Improvement Suggestions

Participants suggested multiple ways to improve the prototype in its current form. They have not provided any improvement regarding the framework. The primary source of suggestions is based on the look and feel of the prototype, such as changing the color of certain elements and ordering of buttons or labels. The second most common advice is to add alternative text input methods, such as a physical keyboard interacting during MR usage or speech-to-text. Four of the participants said some texts are blurry, and an increased resolution could help with

such a problem. One participant suggested using improved hardware, such as Meta Quest 3, to feel more comfortable. One participant proposed adding a panel to navigate diagram elements, while another recommended that selection through the model's diagram elements can be beneficial. Finally, one participant indicated that collaborative work could be much better in MR environments, with an increase in remote work and concurrent work not being ideal in current MBSE applications. The suggestions are categorized and listed in Figure 30.

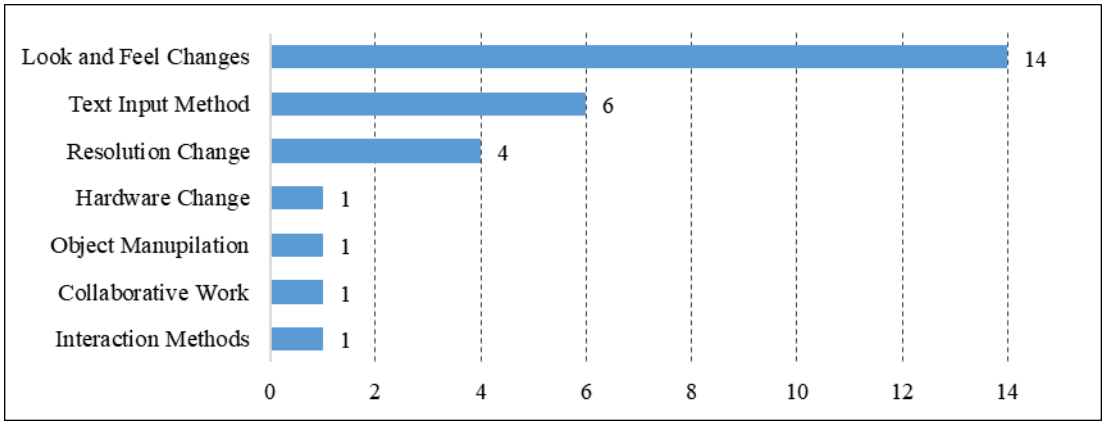


Figure 30 – Pie chart of the summary of participants' improvement suggestions.

CHAPTER 6

DISCUSSION

Extended Reality (XR)–a term that simultaneously covers virtual, augmented, and mixed reality–is becoming even more widespread with devices designed by large corporations. With so many actors competing in this single field, technological advancements are also gaining momentum. The technical specifications of these devices have become similar to cellphones and many base-level laptops. With the increased processing power and storage, many previously known desktop-only applications may be ported over to these devices. This study proposes that modeling tools for MBSE can be used with VR or MR devices and investigates the advantages and disadvantages of such applications. The outcomes of the research can be gathered into five key points.

6.1 The Enjoyable Experience

The strongest point of the MRSysML compared to 2DSysML identified by the participants is the enjoyable experience. They enjoyed spending their time in MR and VR when working on the task. Even though the emotion scores and games scores of the SE group are significantly lower, the number of participants listed enjoyability as the strength is roughly the same for each group. The primary source of enjoyment originates from object manipulation. Compared to using a mouse, being able to use hand interaction or hand-imitating controllers feels more intuitive and allows the user to feel in control. Instead of observing the elements as small windows on the screen, having them as virtual objects in the work environment can feel more stimulating and attractive, which in turn increases the enjoyability of the process. However, it should be noted that the enjoyability of the experience for the SE group may be biased by the novelty of the experience of VR or MR. Many participants had no experience using any XR devices and may find the experience of using the XR enjoyable instead of the actual application.

The enjoyability of the product can assist the user in long-term work by reducing boredom. Working on models for long hours can become repetitive and tiresome, with the added benefit of a more enjoyable experience that may balance it out for a longer work period. An additional advantage of the more enjoyable experience can be using this environment to train beginner systems engineers. Gamification is a field that uses game features to incentivize the user to complete certain tasks. Combining gamification and virtual environments can be an enhanced method of training similar to research by Yigitbas et al. [26] [27].

6.2 Text Input Limitations

The primary issue faced for modeling in VR or MR environments is text input. All participants were familiar with using keyboards and have been doing so for many years. There is an experience bias towards using keyboards, and virtual keyboards do not feel the same way in VR or MR. If the user is using hand interaction, there is no haptic feedback, and for controllers, the kinds of haptic feedback that can be given may feel uncanny when typing. This may frustrate the user and reduce the usability. Over a third of the participants noted this as the weakness of the MRSysML prototype. All modeling languages rely heavily on the texts in the diagram shapes or over the connector edges. Having difficulty entering this information is a significant issue regarding adaptation of the framework in the industry.

There are alternative solutions to these problems that come with newer hardware. Firstly, some of the latest AR and MR hardware can identify the room around the user as meshes and help define a room-scale workspace. These virtual keyboards can be placed on the virtual mesh and assist the user with an actual keyboard feeling. However, this may still struggle with haptic feedback. A second alternative is integrating physical keyboards to work with these devices. This can be done easily with devices such as Meta Orion or Snap Spectacles, where the virtual elements are overlayed on the users' visuals. For devices that employ spatial reconstruction technologies to reconstruct the surroundings using cameras on the device, this can be more troublesome since most devices have problems reconstructing objects nearby and struggle to keep them in focus. The third alternative for text input is speech-to-text. Virtual assistants are seen in everyday products such as phones, home hubs, and cars. It comes with any form of new operating system and relies heavily on speech-to-text. The same technology can be used to input text in any form of application. The primary limiting factor is that many corporations decide to develop their own speech-to-text technology, which is commonly not open to the public to implement anywhere and for commercial products might be costly. This requires integrating the device's specific speech-to-text feature in the application and increases the workload development of the applications adapted to multiple devices.

6.3 The Usability and The User Experience

According to CGS by Sauro and Lewis [42] and the ratings by Bangor et al. [43] [44] the prototypes are above average in general scales. However, 2DSysML has surpassed MRSysML in SUS score by all participants. This highlights the general bias towards the desktop work environment. Familiarity with the mouse and keyboard increases the efficiency of the participant and allows them to complete their tasks more intuitively. While some participants had experience with VR or MR devices, almost none had used its capabilities as a workspace, and adaptation to the environment was necessary. For almost all participants from the SE focus group, the fundamental actions, such as dragging an object, zooming in or out, and selecting, had to be explained before the start of the test sessions.

Another subject that reduced usability is imprecise object manipulation. Having the 3D model use the physics engine's capabilities through MRTK2, the movements of the object are smoother; however, some participants found it to be more troublesome than pleasing. There are no capabilities to align any objects, causing the accelerating movement of objects to cause alignment issues.

According to the acceptability rating by Bangor et al. [43], the SE group found the MRSysML prototype acceptable, while the IAGD group found it marginal. The SE group finding the prototype acceptable shows that the features required to develop SysML models are there, but the marginal rating from the IAGD group shows that the user experience had certain shortcomings. The UEQ benchmark [45] scores have shown that the MRSysML is more novel, attractive, and stimulating, while 2DSysML is more dependable, efficient, and perspicuous. Several aforementioned problems with textual input and object manipulation reduce the dependability of the product. Unfamiliarity with the devices also reduces the perspicuity since the user not just adapts to the prototype but to the environment as well. One of the expressed improvements for the prototypes is to reduce the cumbersomeness of the user actions. Several users stated that the number of steps to complete certain tasks felt more than necessary, and the user experience should be improved to reduce the required number of actions. Several possible improvements are adding additional selection methods, implementing shortcut actions, supporting different model navigation methods, and displaying multiple diagrams concurrently to adjust diagrams simultaneously. The lower score of interface quality of PQ and the improvement suggestions to increase resolution and look and feel changes suggest that the user interface can be improved upon.

6.4 Employability in the Industry

The VR and MR prototype, as it stands, can be used in industry as a basic modeling tool. However, it lacks the wide range of language support and certain quality-of-life features that are provided by commercial modeling tools. While the prototype may not become a widespread tool in the industry, for large corporations that have modeling application products, support for VR and MR can be explored following the findings of this thesis. The prototype shows that while it performs worse compared to the desktop environment prototype, on a global scale, the usability and efficiency are above-average products in different fields, and with further investment, the tool may catch up or even surpass the available desktop variants. Any newly developed tool – for utilizing VR or MR environments for MBSE should follow the defined abstract architecture and should be inspired by the capabilities of the developed framework. It can be developed as a standalone product that can exchange information with other products or as a plug-in to the currently existing products.

6.5 General Hesitation Towards VR and MR Devices

Some of the responses to the open-ended questions have highlighted another issue in the adaptation of XR devices for workspaces. Participants, mainly from the SE group, have raised health concerns such as headaches or neck pain and found the devices' price to make them inaccessible. It is clear that these concerns are based on inexperience with the devices. It is not correct to judge the devices based on a single sample. The types of VR or MR devices are vast. There are certain MR glasses that weigh down to around 250 grams, and the prices of certain commercially available devices can be as low as the cost of a base-level office laptop. The prototype can be run on a newer model VR and MR device, which weighs roughly 503 grams and has customized straps for different health problems and long-term usage. Either way, the general hesitation seems to be a barrier to a more widespread adaptation.

CHAPTER 7

CONCLUSION AND FUTURE WORK

The continuously evolving fields of MBSE and XR complement one another with mutual benefits. Systems engineering is shifting focus from document-based methodology to model-based methodology to provide a holistic view in a simpler and more understandable practice, with reduced redundant information and over-engineering problems. The XR devices are expanding to newer fields, and workplaces are one of the key target areas. With the increasing interest in VR and MR devices, systems engineers may be able to utilize these technologies as an alternative to desktop environments. This study proposed a basic framework for using MR or VR devices to conduct MBSE. It includes developing a monolithic MBSE tool prototype supporting SysML–a general-purpose modeling language for systems–running on MR or VR devices named MRSysML and another that runs on desktop environments named 2DSysML. With assistance from 30 participants who have either systems engineering or interactive applications and game development backgrounds, user testing sessions are conducted using these prototypes. With an equal number of participants from both backgrounds, the experience from these sessions' participants evaluated the prototypes based on presence, usability, user experience, and general opinions. The data collected suggest that the MR or VR variant of the prototypes is a useful application for performing MBSE, providing an enjoyable experience with possible improvements to increase efficiency. It should be noted that while the VR or MR variant is acceptable, the desktop variant seems to get more attention and preference. There is a bias towards using desktop environments due to familiarity and experience. It may not be possible to change this in the short term, but for future generations, this may not be an issue.

The defined abstract framework can be applied to many types of prototypes and can be expanded upon with additional features to incorporate similar to simulation, digital twin, or collaborative work. It can be adjusted as a distributed software system with separate applications conforming to each individual component or to utilize the stronger processing powers of servers and use the devices solely for displaying models and obtaining user actions. To the best of the author's knowledge, there is no common framework in academia for designing diagrams on VR or MR devices without requiring any third-party application, but there are for visualization and simulation of the models. The abstract framework can be implemented in a different software architecture, or the MRSysML can be improved upon to build a fully functional framework with additional capabilities.

The groundwork done by this study highlights multiple issues that need to be considered for the widespread utilization of MR and VR devices. The first one is the biases of the people unfamiliar with these devices. There are certain hesitations towards the usability of MR and VR devices for long-term work, which seems to originate from a lack of knowledge. The second one is that the text input feels difficult with virtual keyboards and has to be replaced with an alternative. Speech-to-text seemingly is the best alternative since almost all new technologies seem to support such features.

Additional studies should be conducted to expand the usability of MBSE in VR or MR environments. Due to the high enjoyability, combining gamification with MBSE can be a strong training tool. Systems engineering, being an independent engineering discipline that utilizes information from all engineering disciplines, creates a barrier to entry, and the barrier

can be surpassed with better interactive training methods other than learning by experience, supplementary courses, or postgraduate education.

Collaborative work can be enhanced with the usage of MR devices. Being able to see the avatars of other collaborators in real-time can improve the experience. Findings of Yigitbas et al. [28] can be explored in the MBSE domain to see if any differences exist between software design and systems design. The effectiveness of collaborative work done remotely over MR devices should also be investigated.

REFERENCES

[1] R. Cloutier and M. Pennotti, “A brief history of systems engineering,” Systems Engineering Body of Knowledge (SEBoK), https://sebokwiki.org/wiki/A_Brief_History_of_Systems_Engineering (accessed Oct. 26, 2024).

[2] “MIL-STD-498: SOFTWARE DEVELOPMENT AND DOCUMENTATION.” Department of Defense, Washington D.C., 1994

[3] M. Adedjouma, T. Thomas, C. Mraidha, S. Gerard, and G. Zeller, “From Document-Based to Model-Based System and Software Engineering,” in *Joint Proceedings of EduSymp 2016 and OSS4MDE 2016*, 2016, pp. 27–36

[4] A. W. Wymore, *Model-Based Systems Engineering*, 1st ed. CRC Press, 1993.

[5] J. Hugues, “About AADL,” Open AADL, http://www.openaadl.org (accessed Oct. 26, 2024).

[6] “OMG Systems Modeling Language (OMG SysML).” Object Management Group® Standards Development Organization (OMG® SDO), Nov. 2019

[7] “MIL-STD-499: Military Standard System Engineering Management.” Department of Defense, 1969

[8] “MILL-STD-499A: Military Standard Engineering Management Notice 1.” Department of Defense, 1995

[9] “MIL-STD-499A: Military Standard Engineering Management Notice 2.” Department of Defense, 2017

[10] “About systems engineering,” INCOSE, https://www.incose.org/about-systems-engineering (accessed Oct. 26, 2024).

[11] “Our story,” MITRE, https://www.mitre.org/who-we-are/our-story (accessed Oct. 26, 2024).

[12] Systems Engineering Guide. MITRE, 2014.

[13] G. Rebovich, “The Evolution of Systems Engineering,” in *SysCon 2008 - IEEE International Systems Conference*, 2008

[14] I. G. Vargas and R. T. Braga, “Understanding System of Systems Management: A systematic review and Key Concepts,” *IEEE Systems Journal*, vol. 16, no. 1, pp. 510–519, Mar. 2022. doi:10.1109/jsyst.2020.3018068

[15] A. M. Madni and M. Sievers, “Model-based systems engineering: Motivation, current status, and research opportunities,” *Systems Engineering*, vol. 21, no. 3, pp. 172–190, May 2018. doi:10.1002/sys.21438

[16] K. X. Campo *et al.*, “Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature,” *Systems Engineering*, vol. 26, no. 1, pp. 104–129, Oct. 2022. doi:10.1002/sys.21644

[17] H. Yang *et al.*, “Research on visual simulation for complex weapon equipment interoperability based on MBSE,” *Multimedia Tools and Applications*, vol. 83, no. 5, pp. 13463–13482, Jul. 2023. doi:10.1007/s11042-023-15950-5

[18] B. Cameron and D. M. Adsit, “Model-based systems engineering uptake in engineering practice,” *IEEE Transactions on Engineering Management*, vol. 67, no. 1, pp. 152–162, Feb. 2020. doi:10.1109/tem.2018.2863041

[19] M. Lutfi and R. Valerdi, “Virtual reality in Model Based Systems Engineering: A review paper,” *Communications in Computer and Information Science*, pp. 197–205, 2020. doi:10.1007/978-3-030-60703-6_25

[20] A. Kande, thesis, 2011

[21] M. Lutfi and R. Valerdi, “Framework for integration of virtual reality into model based systems engineering approach,” *Lecture Notes in Networks and Systems*, pp. 131–139, 2021. doi:10.1007/978-3-030-80091-8_16

[22] M. Lutfi and R. Valerdi, “Integration of SysML and virtual reality environment: A ground based telescope system example,” *Systems*, vol. 11, no. 4, p. 189, Apr. 2023. doi:10.3390/systems11040189

[23] R. Oberhauser, “VR-UML: The Unified Modeling Language in virtual reality – an immersive modeling experience,” *Lecture Notes in Business Information Processing*, pp. 40–58, 2021. doi:10.1007/978-3-030-79976-2_3

[24] R. Oberhauser, “VR-SysML: SysML Model Visualization and Immersion in Virtual Reality,” in *MODERN SYSTEMS 2022*, 2022

[25] R. Oberhauser, “VR-SysML+Traceability: Immersive Requirements Traceability and Test Traceability with SysML to Support Verification and Validation in Virtual Reality,” *International Journal On Advances in Software*, vol. 16, no. 2, pp. 23–35, Jun. 2023.

[26] E. Yigitbas, M. Schmidt, A. Bucchiarone, S. Gottschalk, and G. Engels, “Gamification-based UML Learning Environment in virtual reality,” *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, vol. 2145, pp. 27–31, Oct. 2022. doi:10.1145/3550356.3559088

[27] E. Yigitbas, M. Schmidt, A. Bucchiarone, S. Gottschalk, and G. Engels, “GAMOVR: Gamification-based UML Learning Environment in virtual reality,” *Science of Computer Programming*, vol. 231, p. 103029, Jan. 2024. doi:10.1016/j.scico.2023.103029

[28] E. Yigitbas, S. Gorissen, N. Weidmann, and G. Engels, “Collaborative software modeling in virtual reality,” *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pp. 261–272, Oct. 2021. doi:10.1109/models50736.2021.00034

[29] “Unity Engine,” Unity, https://unity.com/products/unity-engine (accessed Oct. 26, 2024).

[30] B. Wagner, Overview - A tour of C#, https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview (accessed Oct. 26, 2024).

[31] What is Mixed Reality Toolkit 2, https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05 (accessed Oct. 26, 2024).

[32] “OMG Unified Modeling Language (OMG UML).” Object Management Group® Standards Development Organization (OMG® SDO), Dec. 2017

[33] “OMG Object Constraint Language (OCL).” Object Management Group® Standards Development Organization (OMG® SDO), Jan. 2012

[34] J. E. McDonough, “Singleton Design Pattern,” in *Object-Oriented Design with ABAP: A Practical Approach*, 2017, pp. 137–145

[35] M. Richards, *Software Architecture Patterns*, 1st ed. O’Reilly Media, 2015.

[36] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Addison-Wesley Professional, 1994.

[37] B. G. Witmer and M. J. Singer, “Measuring presence in virtual environments: A presence questionnaire,” *Presence: Teleoperators and Virtual Environments*, vol. 7, no. 3, pp. 225–240, Jun. 1998. doi:10.1162/105474698565686

[38] “Immersive Tendencies Questionnaire.” UQO Cyberpsychology Lab, Mar. 2013

[39] B. G. Witmer, C. J. Jerome, and M. J. Singer, “The Factor Structure of the presence questionnaire,” *Presence: Teleoperators and Virtual Environments*, vol. 14, no. 3, pp. 298–312, Jun. 2005. doi:10.1162/105474605323384654

[40] J. Brooke, “SUS: A ‘quick and dirty’ usability scale,” *Usability Evaluation In Industry*, pp. 207–212, Jun. 1996. doi:10.1201/9781498710411-35

[41] B. Laugwitz, T. Held, and M. Schrepp, “Construction and evaluation of A user experience questionnaire,” *Lecture Notes in Computer Science*, pp. 63–76, 2008. doi:10.1007/978-3-540-89350-9_6

[42] J. Sauro and J. R. Lewis, *Quantifying the User Experience*. Elsevier Science, 2012.

[43] A. Bangor, P. T. Kortum, and J. T. Miller, “An empirical evaluation of the system usability scale,” *International Journal of Human-Computer Interaction*, vol. 24, no. 6, pp. 574–594, Jul. 2008. doi:10.1080/10447310802205776

[44] A. Bangor, P. Kortum, and J. Miller, “Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale,” *Journal of Usability Studies*, vol. 4, no. 3, pp. 114–123, May 2009.

[45] M. Schrepp, A. Hinderks, and J. Thomaschewski, “Construction of a benchmark for the User Experience Questionnaire (UEQ),” *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 4, p. 40, 2017. doi:10.9781/ijimai.2017.445

[46] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, Dec. 1945. doi:10.2307/3001968

[47] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, Mar. 1947. doi:10.1214/aoms/1177730491

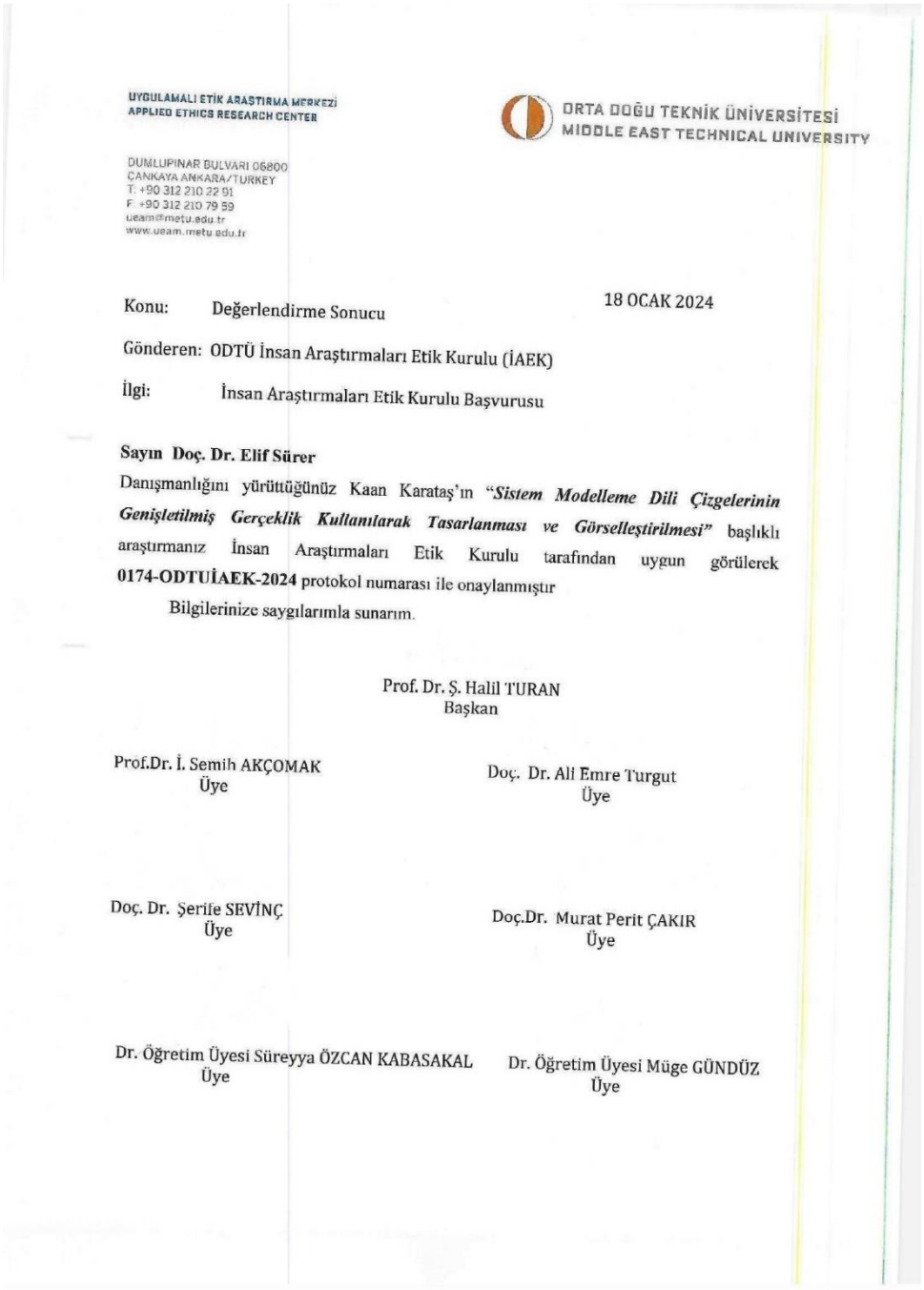
[48] “The Wilcoxon signed-ranks Test Calculator,” Social Science Statistics, https://www.socscistatistics.com/tests/signedranks/default2.aspx (accessed Oct. 26, 2024).

[49] “Mann-Whitney U Test Calculator,” Social Science Statistics, https://www.socscistatistics.com/tests/mannwhitney/default2.aspx (accessed Oct. 26, 2024).

APPENDICES

APPENDIX A

A.1. ETHICAL APPROVAL



UYGULAMALI ETİK ARAŞTIRMA MERKEZİ
APPLIED ETHICS RESEARCH CENTER

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 22 91
F: +90 312 210 79 59
ueam@metu.edu.tr
www.ueam.metu.edu.tr

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

18 OCAK 2024

Konu: Değerlendirme Sonucu

Gönderen: ODTÜ İnsan Araştırmaları Etik Kurulu (İAEK)

İlgi: İnsan Araştırmaları Etik Kurulu Başvurusu

Sayın Doç. Dr. Elif Sürer

Danışmanlığını yürüttüğünüz Kaan Karataş'ın ***"Sistem Modelleme Dili Çizgelerinin Genişletilmiş Gerçeklik Kullanılarak Tasarlanması ve Görselleştirilmesi"*** başlıklı araştırmanız İnsan Araştırmaları Etik Kurulu tarafından uygun görülerek **0174-ODTUİAEK-2024** protokol numarası ile onaylanmıştır

Bilgilerinize saygılarımla sunarım.

Prof. Dr. Ş. Halil TURAN
Başkan

Prof.Dr. İ. Semih AKÇOMAK
Üye

Doç. Dr. Ali Emre Turgut
Üye

Doç. Dr. Şerife SEVİNÇ
Üye

Doç.Dr. Murat Perit ÇAKIR
Üye

Dr. Öğretim Üyesi Süreyya ÖZCAN KABASAKAL
Üye

Dr. Öğretim Üyesi Müge GÜNDÜZ
Üye

APPENDIX B

B.1. SUPPLEMENTARY FIGURES OF THE PROTOTYPES (PART 1)

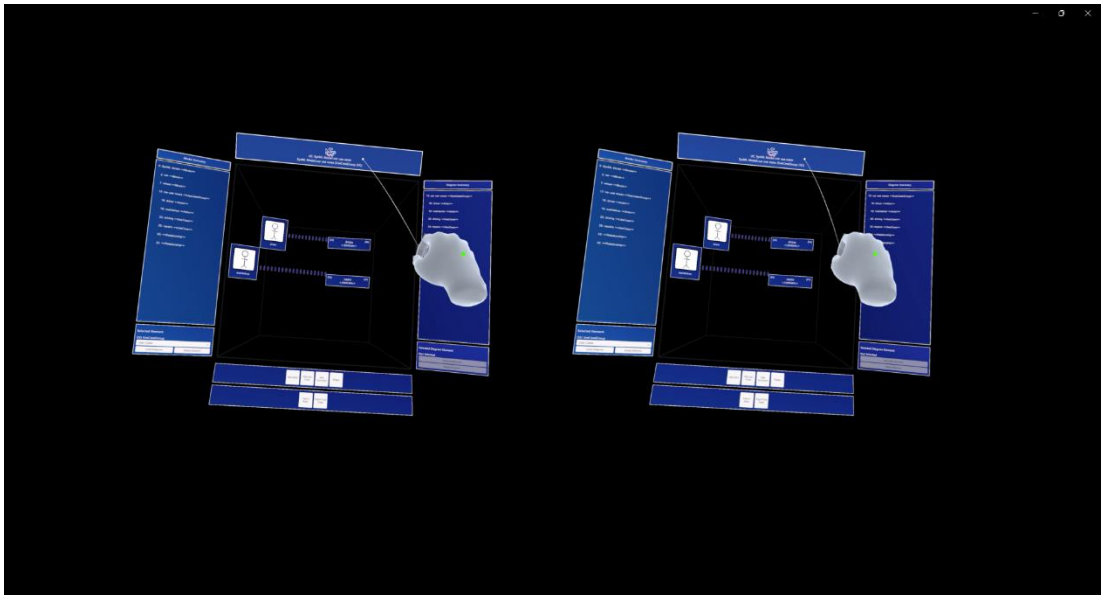


Figure 31 – User using hand interaction to grab the top bar to move it around with view from both eyes.

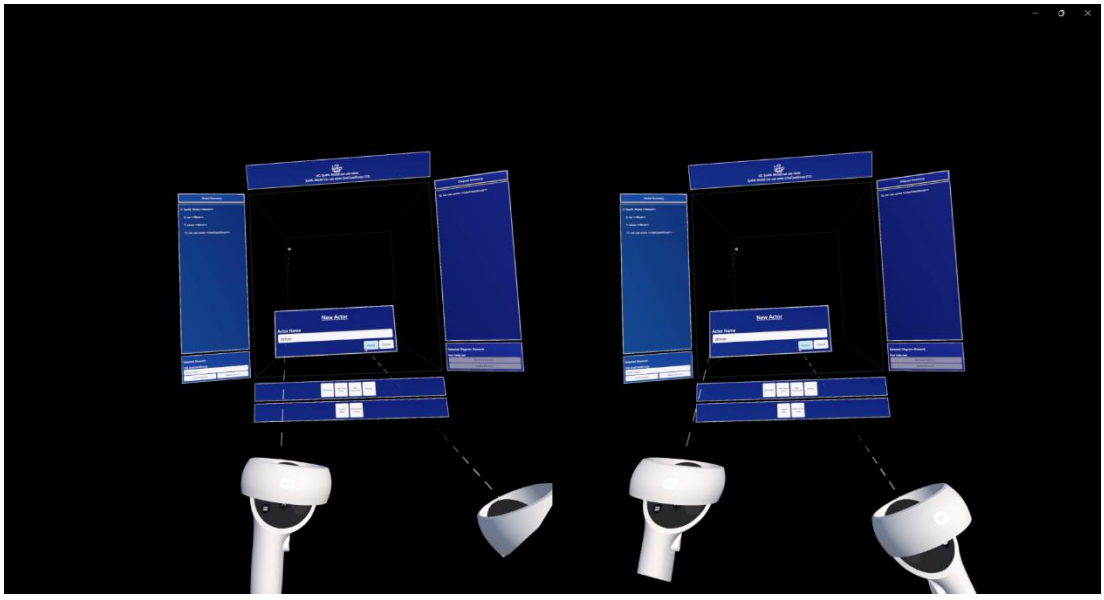


Figure 32 – User defining a new actor diagram element with the name driver with view from both eyes

B.2. SUPPLEMENTARY FIGURES OF THE PROTOTYPE (PART 2)

Figure 33 – Visualization of use case diagram in MRSysML designed by the user with controller interaction method with view from both eyes.

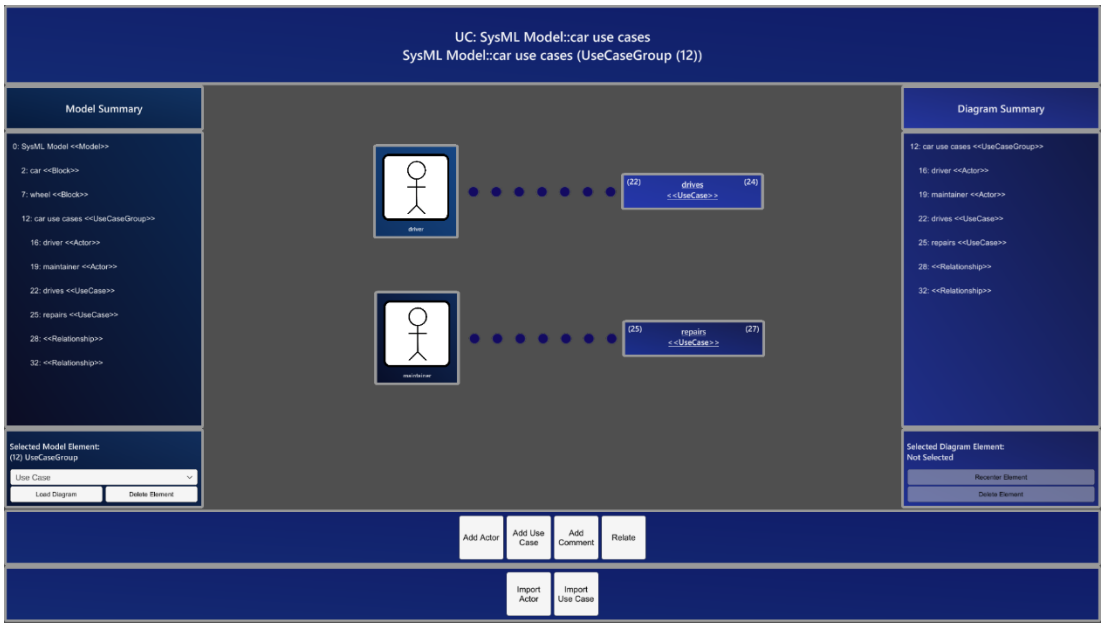


Figure 34 – Visualization of use case diagram in 2DSysML designed by the user.