Design of a Digital Twin for a Smart Meeting Room based on a Model-Driven System Engineering Approach

Sangiliayyah, Harisankar

Award date:
2020

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Design of a Digital Twin for a Smart Meeting Room based on a Model-Driven System Engineering Approach

Final Graduation Thesis Report on Digital Twin systems for a Smart meeting Room

Harisankar Sangiliayyah
Student Number: 1367878
Supervisor:
Dr. Ion Barosan

Graduation thesis report submitted in the partial fulfilment of the requirements of the degree Master of Science in Embedded Systems
version 1.4

Eindhoven, November 2020

Abstract

The recent developments in Digital Twin systems and intelligent buildings have increased the research interest in these domains. The improvements in building infrastructure, construction planning, IoT interface and predictive analysis techniques has to be matched with a suitable approach to design the Digital Twin system for such applications. The testing of product development or life-cycle development of intelligent buildings is usually a time consuming process and therefore it is a blockage in the productivity. However, the main advantage of Digital Twin systems is its efficient operation and improved productivity. Therefore, as a graduation thesis, this project addresses the typical concerns of developing a Smart Meeting Room (SMR) application using a model-driven system engineering approach and designing a Digital Twin system in a 3D virtual space. The objective of this method is to verify that the Digital Twin system can act as a suitable testing method in testing the SMR application using the model-driven system engineering approach.

In this project, firstly, a suitable SMR system is designed and developed in the IBM Rhapsody developer tool. For this purpose, several subsystems were considered and initial assumptions were made. subsequently, the considerations and assumptions are modelled as system requirements. The SMR application is modelled using the SysML approach in the IBM Rhapsody tool. The SMR application work as a stand-alone system and the simulation verifies the working of a typical SMR application by having a GUI within the tool. Secondly, the project involves the design and development of Digital Twin system to be integrated into the SMR system using Unity Game Engine tool. For this purpose, the Digital Twin layered architecture is employed to develop a modern office interior and integrate it with the system behaviour of the SMR application that runs parallelly in IBM Rhapsody tool. However, to enable communication between two different processes, the network socket communication is established. Finally, the model is extended for HIL testing by porting the SMR application to a raspberry pi controller. However, to enable project execution on a remote device certain constraints were met such as creating a linux based profile on IBM Rhapsody tool and automating the process.

In short, the SMR application running in IBM Rhapsody tool together with the virtual model in Unity Game Engine serves as a verification method that demonstrates the working of a Digital Twin system for SMR application. Thereby, the project results in drawing conclusions on the importance of Digital Twin system and stating the future scope of research in this domain.

Contents

List of Figures

Chapter 1

Introduction

The need for an intelligent environment has always been at peak ever since the onset of technology and Internet-of-Things (IoT) revolution. Re-imagining the world around us that behaves as a connected and integrated organism which solves most of the day-to-day mechanical and repetitive tasks, can be something very interesting. In this context, advanced intelligent buildings are definitely a useful area of research. " An intelligent building is one that is responsive to the occupants' needs, satisfies the aims of an organization, and meets the long-term aspirations of society " - (Clements-Croome, 2009).

To define an effective intelligent building, the various life cycle processes including planning, design, construction, commissioning, and facilities management are all vitally important. All these processes are usually time-consuming and costly. Therefore, it is highly essential to devise some common methodologies and standards to adopt so as to establish cost-effective ways of performing the life cycle processes of an intelligent building. Designing and testing of the controllers used in such applications is the first and foremost step. There are various levels of testing in controller development. However, the models created in traditional methods do not always pertain to the practical working of the hardware. The problem of diverging models poses a big challenge. It is therefore necessary to incorporate an additional method that solves the problem of diverging models. The goal of this project is to create a Model-in-loop, Software-in-loop, and a Digital-Twin-in-the loop controller for a smart room application that serves as a testing method to verify the controller.

This document entails the final graduation report of the thesis project on the design and development of a Digital Twin system for a Smart Meeting Room (SMR) based on model-driven system engineering approach. Therefore, the project is divided into four phases. First: model the behaviour of all the sub-systems of the SMR and analyse all the attributes that interact with the subsequent subsystems involved. Second: To create a central intelligence and logic for the overall system and to generate the software code for all the subsystems and the controller. Third: To develop the Digital Twin model in a 3D virtual environment. Fourth: Integrating the models to observe the various scenarios in testing the controller behaviour in simulated virtual cases.

This project focuses on each aspect of designing a SMR system under consideration, the assumptions made, establishing the overall objectives of the project, creating models for each subsystem, identifying the attributes for system communications, and discussing a suitable method of implementation for each of the phases of the project along with the step-by-step implementation tutorial. Furthermore, a literature study on a suitable digital twin systems was made. The purpose, efficiency, and futuristic views of such systems are discussed.

In this report, each phases of the project are documented as follows. After the introduction chapter, chapter 2 establishes the project description along with the objectives and research questions. Next, Chapter 3 deals with the Literature survey on digital twin systems. This is done so as to develop ideas and frame the design in suitable approach followed in previous implementations of Digital Twin systems. Subsequently in chapter 4, the software development methodologies followed for this project are established. Further, chapter 5 introduces to SMR system requirements and SMR system analysis. Next, chapter 6 moves on to designing the SMR system along with the Digital Twin system needed for integration. After portraying the plausible design for SMR system and Digital Twin, Chapter 7 discusses the implementation of the SMR system and integrates it with the Digital Twin developed in Unity Game Engine, followed by chapter 8 which demonstrates the results of implementation. Finally, Chapter 9 shows the conclusions and future scope of improvement in each areas of the project.

Chapter 2

Project Description

In this project, a miniature version such as a smart meeting room inside an intelligent building is considered to represent a Digital Twin for intelligent buildings. The description of the project along with the sub-systems required for suitable implementation are discussed in this chapter. Furthermore, the objectives of this project is stated in the form of three research questions which is discussed below in section 2.2.

2.1 System under Test

As a first step in researching the way of designing and developing a controller used in intelligent buildings the following things are to be take care. A smart room with various drivers (sub-systems) that interact with the central system (SMR) is considered. For instance, a typical Smart room has the following devices: a HVAC system to control and monitor the room temperature, The security and access system to avoid any unauthorized access, the safety system to alert the occupants of any possible hazards such as fire and increased CO2 levels, an automatic lighting system that provides adjustable lighting so as to maintain comfort and an audio system with speakers that has inbuilt functionalities for making audio settings etc. The software controller takes the inputs from all the simulated sensors and intelligently takes decision to actuate the devices in an efficient way such that power consumption is minimal and also offers comfort to user.

This project mainly focuses on creating a Digital Twin as a testing method to verify the software control and processes of a smart meeting room. The research questions here will be about developing a Digital Twin model that best replicates the scenario of a meeting room and also serves as a representational model for any intelligent building. It is therefore necessary to analyse the models from individual sub-systems that are required to formulate an ideal smart meeting room system. The models are to be created with Unity Game Engine with the help of Unity's asset store package that helps in developing models from templates that are suitable for office interior environments.

2.2 Question 1

How to design the SMR system using a model-driven system engineering approach?

The Requirement, Design, Analysis and Specification of the SMR model is developed in the V-model project development approach. Section 5.1 discusses the analysis of SMR system for modelling it in IBM Rhapsody tool.

2.3 Question 2

How to design and implement a Digital Twin for Smart Meeting Room using a Digital Twin approach and integrate it with the SMR system?

To implement a Digital Twin for SMR model, we make use of the Digital Twin layered architecture for building levels discussed in chapter 2. The input/output variables from each of the layers is described in section 6.3. The integration of Digital Twin with the SMR system is discussed in section 7.2.

2.4 Question 3

How to couple the Digital Twin with a real system? How to test the functionality of the real system using the Digital Twin?

To test the SMR system with a real system, we couple it with a Raspberry Pi controller. The simulated user inputs are now fed into the controller. The responses of the Digital Twin system and the Raspberry Pi controller are analysed in parallel. This is demonstrated by implementing it in section 7.4.

Additionally, various sub-systems have to be tested by creating multiple versions of similar models independently. Testing for basic functionalities of individual subsystems allow us to understand the behaviour in a real-life scenario. Furthermore, Creating multiple scenarios and testing the interaction of various sub-systems will help us realize the goals of this project.

Chapter 3

Literature Survey on Digital Twin systems

Digital Twin serves as a testing method in different fields of engineering. The recent developments in Digital Twin market, a literature survey of the Digital Twin System, and the system architecture of a Digital Twin System are discussed in this chapter.

3.1 Related Information

The Digital Twin is a leading technology driving attention in fields like manufacturing, automobile, healthcare, retail, smart cities, and industrial IoT. It is an integration of all modern intelligence technologies, including Big Data, artificial intelligence, machine learning, and IoT used for predictive analysis of any system or equipment [3]. The Digital Twin combines real-time data and predictive data from existing software products. Another important element is the technology required to visualize the information that comes from the Digital Twin Systems. It helps in a simple and effective way of improving cost efficiency, power optimization, speed and time discreet sensors at different locations, collects data, and consolidate under one hub, making it possible for everyone to access the data, which acts as a physical environment for digital space [4]. Digital

Twining has the ability to take the virtual representation of the elements and dynamics of how the internet of things operates and works. The combination of physical elements and virtual elements aids better manufacturing, high tolerance, determination of product efficiency through a different environment. In Digital Twins, the physical data influences the product design for better performance and better manufacturing before taking the data in a Digital space, the dynamics of the information is analysed through various analyses such as video processing, natural language processing, acoustic analysis, video analysis, etc., to check the variance and tolerances [7]. In a nutshell, Digital Twin is a complex data-driven model which has a purpose in almost all fields.

Latest advancements and developments in Digital Twins include:

- Digital Twin in product design.
- Digital Twin in smart manufacturing.
- Digital Twin job-shop.
- Modelling and simulation of Digital Twin.
- Smart interconnection and inter-operation of Digital Twin.
- Digital Twin in human-machine collaboration.
- Digital Twin in product life-cycle data management.

3.2 Methods of modelling a Digital Twin System (DTS)

The context of the operation of Digital Twin involves an instrumented test-bed in which model-based systems engineering (MBSE) tools(e.g., system modelling and verification tools) and operational scenario simulations (e.g., discrete simulation and agent-based simulations) are used. Insights from the operational environment are used to modify the system models used in the virtual prototype [5]. Data Supplied by the Physical System (PS) are used by the Virtual prototype to instantiate the Digital Twin System (DTS) [2].

The MBSE tool suite provides a means to model the processes of a system. The modelling methods include Design Structure Matrix, Process dependency structure matrix, Probabilistic models such as Partially-Observable Markov Decision Process (POMDP), Discrete event simulation, agent-based simulation, Model-based storytelling, an MBSE knowledge base, and systems engineering life cycle process models [5].

The recent industry trends have raised the extent of research attempts in the field of Digital Twin development in building and city levels. The existing definitions related to DTS are discussed in this section. New insights can be learned through the review of current literature discussing limitations related to research efforts on the concepts of DTS in the Architecture, Engineering and Construction, and Facilities Management industry (AEC/FM sector). This section aims at providing an introduction to the system architecture followed in the development of Digital Twins.

3.2.1 Digital Twin architectures

There are various multi-tier architectures to support heterogeneous environments (e.g., multi-function and a large amount of data). It can be classified as Cyber-Physical Systems (CPS), IoT platform architectures, and smart cities architectures. Digital Twins can support many different applications such as energy management, security, and health monitoring. Data requirements differ for each application. Therefore, it becomes a problem when data comes from different systems, because the system may have a different intended use of these data that does not fully match the requirements of all those applications.

3.2.2 A Digital Twin System Architecture for Building and City levels

In this section, the system architecture used for DTs, which are specifically designed at both the building and city levels is discussed based on the DT demonstrator developed for the West Cambridge site (Developing a Digital Twin at Building and City Levels: A Case Study of West Cambridge Campus - Qiuchen Lu) [6]. The figure represents the DT model for building level, implemented at the West Cambridge campus.

A city is a comprehensive system that connects various physical, social, and business aspects. A city can be considered as an asset that integrates different sub-assets such as buildings, bridges, utilities, infrastructure, and people (sub-Digital Twin). A typical building level Digital Twin consists of the cluster of sensors, the sensor manager platform, manager/user access, services, Asset manager platform and the cloud database connection. A representation of the Digital Twin model for building level is depicted in figure 3.1. Similarly, in this project, various sub-systems such as the HVAC system, Audio system, safety system, access and security system, and lighting system can be considered as sub-Digital Twins. Each sub-Digital Twins have a child-parent relationship of Digital Twins at different levels.

The system architecture of Digital Twin development in a city is a layered architecture. This architecture is comprised of 5 layers: Data acquisition layer, Transmission layer, a Digital modelling layer, Data/Model integration layer, and Service layer. Each layer in the system architecture is depicted in figure 3.2.

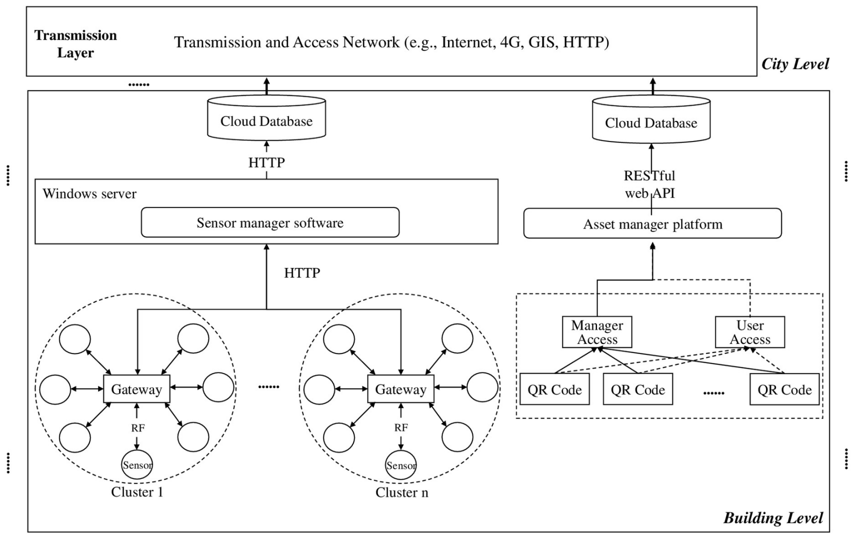


Figure 3.1: Digital Twin model for Building level

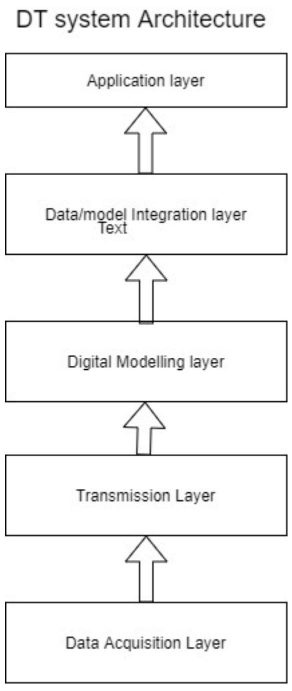


Figure 3.2: DT system Architecture

Data Acquisition layer

Data Acquisition layer is the foundation of each Digital Twin. The design of the data acquisition mechanism and approach is a foremost and challenging task due to the large volume and heterogeneous nature of data in city levels. Examples of data collection techniques include contact-less data collection (RFID, Image-based techniques), distributed sensor systems, wireless communication, and mobile access (e.b., WiFi environment).

Transmission layer

The transmission layer aims at transferring the acquired data to the higher layers for modelling and analysis. Communication technologies such as short-range coverage, access network technologies (e.g., WiFi, Zigbee, near field communication (NFC), M2M, and Zwave) and some of the wider coverage (i.e., 3G, 4G, 5G) can be used in this layer. For developing DTs in building and city levels, the light fidelity (Li-Fi) and LP-WAN was considered as a promising means for wide-range coverage. (Saini 2016; Silva et al. 2018).

Digital Modelling layer

This layer actually contains all the digital models of the physical assets (e.g., BIM, City Information Modelling (CIM)) and supplements information (e.g., weather information, cultural backgrounds) that supports the upper layers. Different models/model types can be used for different purposes in Digital Twins. Examples of these are: real-time status/control, managing assets (e.g., asset management model), planning infrastructure/cities (e.g., CIM), modelling scenarios and decision support (Bolton et al., 2018; Kim et al., 2018).

Data/Model Integration layer

This layer aims at integrating all the data resources based on the designed data structure. This layer also contains the functions required for data and model, manipulating, storing, analysing, processing. In this Architecture, real-time data analysis and processing functions would update as-is condition of the city assets (including transportation conditions and energy consumption). Here, cloud storage and computing, and data/model visualization can be used to achieve dynamic and effective data management in a city and building level. In addition, intelligent functions can keep updating their embedded algorithms and supporting continuous applications in future development.

Service layer

The service layer is the top and the implementation layer of the Digital Twin architecture that interprets the knowledge from Kernel and enables the interaction between people/society and the data/model integration layer. This layer provides services for different functions, evaluates performances for the constructed Digital Twins.

Chapter 4

Software development methodologies for SMR system and Digital Twin

The objectives of the project include the demonstration of the development process through Model-in-Loop, Software-in-Loop, Digital-Twin-in-Loop and Hardware-in-Loop [8]. We know that it is a challenge to create models that perfectly aligns with the actual hardware testing. The Digital-Twin-in-Loop acts as a new interface that solves the problem of diverging models. Also, most of the companies after Software-in-Loop testing move on to Hardware testing, which is more expensive, slow and prone to error. Therefore, it becomes necessary to include Digital Twin models. So, the new development process through various stages can be seen in figure 4.1. The various Development Stages and Project Development Method are discussed in the below sections.

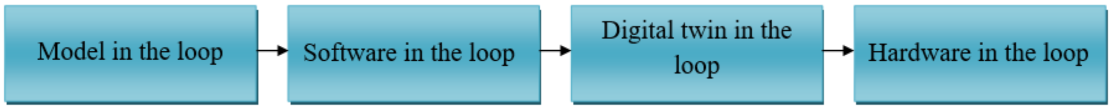


Figure 4.1: Development process

In this project, the MIL development is done in IBM Rhapsody. Once the models of the devices and drivers are developed, the software code is generated and a meta model of the SMR application is created to manage the functions of the smart room. The SMR application is then linked to the Unity Game engine via C# scripting. In HIL phase, The signals from the software controller will further be tested with a raspberry pi controller for conformity. The controller can further be modelled in Simulink and made to interact with the IBM Rhapsody models. As shown in Figure 4.1 the development process is tested sequentially in each of the four phases. The research question here is to find out the ways of interaction between the 3D models developed in the Unity game engine and the SysML profiling done via IBM Rhapsody using the Prespective software package.

Fig 4.2 shows the modular approach to software designing [8]. The modular approach is so as to segregate I/O nodes for the communication between various blocks. One functional layer on top of other functional layers allows us to alter the logic and functionality of the system [8]. The functional layer is mainly responsible for making decisions, Processing data, dynamic scaling to fulfil the functional requirements of the system. In case if there are any future updates, the particular block can be modified and tested, which avoids redundant model checking and updates.

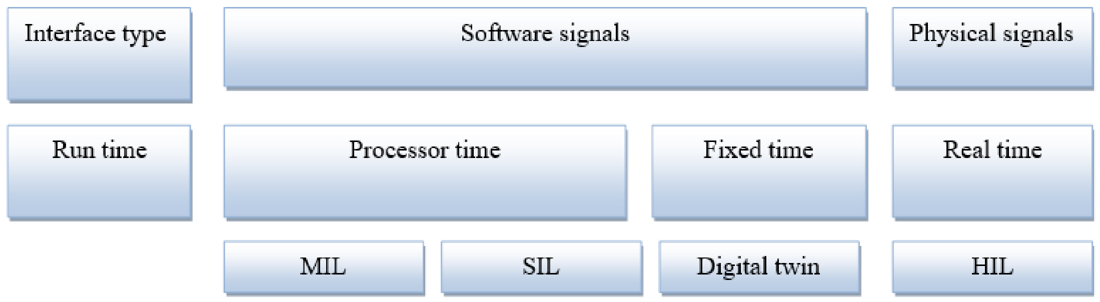


Figure 4.2: Modular approach to software designing

4.1 Development stages in creating a Smart Meeting Room System (SMR)

4.1.1 Model-in-Loop (MIL)

In this stage, a model for the controller and individual devices are designed. The MIL development is done in IBM Rhapsody and it does not require any kind of physical signals. The main goal of MIL is to create a control algorithm and to ensure that the interactions between the controller and the models do not violate the System Requirements.

The requirements are identified as functional, general, and specific for each of the sub-systems. Satisfying all the derived requirements proves the validity of MIL testing.

SysML modelling

SysML is an enabling technology for Model-Based Systems Engineering (MBSE). SysML is a dialect of UML 2 and is defined as a UML 2 Profile. A UML Profile is a UML dialect that customizes the language via three mechanisms: Stereotypes, Tagged Values, and Constraints.

SysML approach is the cost-effective method of application of modelling systems to explore and test system characteristics. Early testing and validation of system characteristics facilitate timely learning about properties and behaviours, enabling fast feedback on requirements and design decisions.

4.1.2 Software-in-Loop (SIL)

After meta-modelling is done, the models are made to generate C# code for the software controller. The software code is coupled with the unity scripts to feed inputs to the 3D environment developed in unity. Moreover, The software signals are essential for further HIL testing. The software signals from IBM Rhapsody are ported to the Raspbian environment via a python interface so as to test with the Raspberry Pi controller for HIL testing. The software also acts as an interface to connect to the Unity engine via C hash scripting. This helps in serving the Digital-Twin.

4.1.3 Digital-Twin-in-Loop (DTIL)

A digital Twin represents a physical object virtually, which can be used to analyse the performance of a developed system. The design for digital Twin modelling has to be done via Prespective software package which has inbuilt models. The smart meeting room which consists of a lighting system, audio and visual systems, security and access systems, and safety systems are to be modelled in unity scripts. The system interactions and controller behaviour which was modelled in IBM Rhapsody is now fed to the 3D virtual environment. In this way, it enables us to actually observe the responses of the smart room for multiple scenarios, use cases, and user inputs.

The structure of the overall development process is depicted in figure 4.3. To facilitate scenario-based testing, blocks are identified as Driver, controller, and virtual.

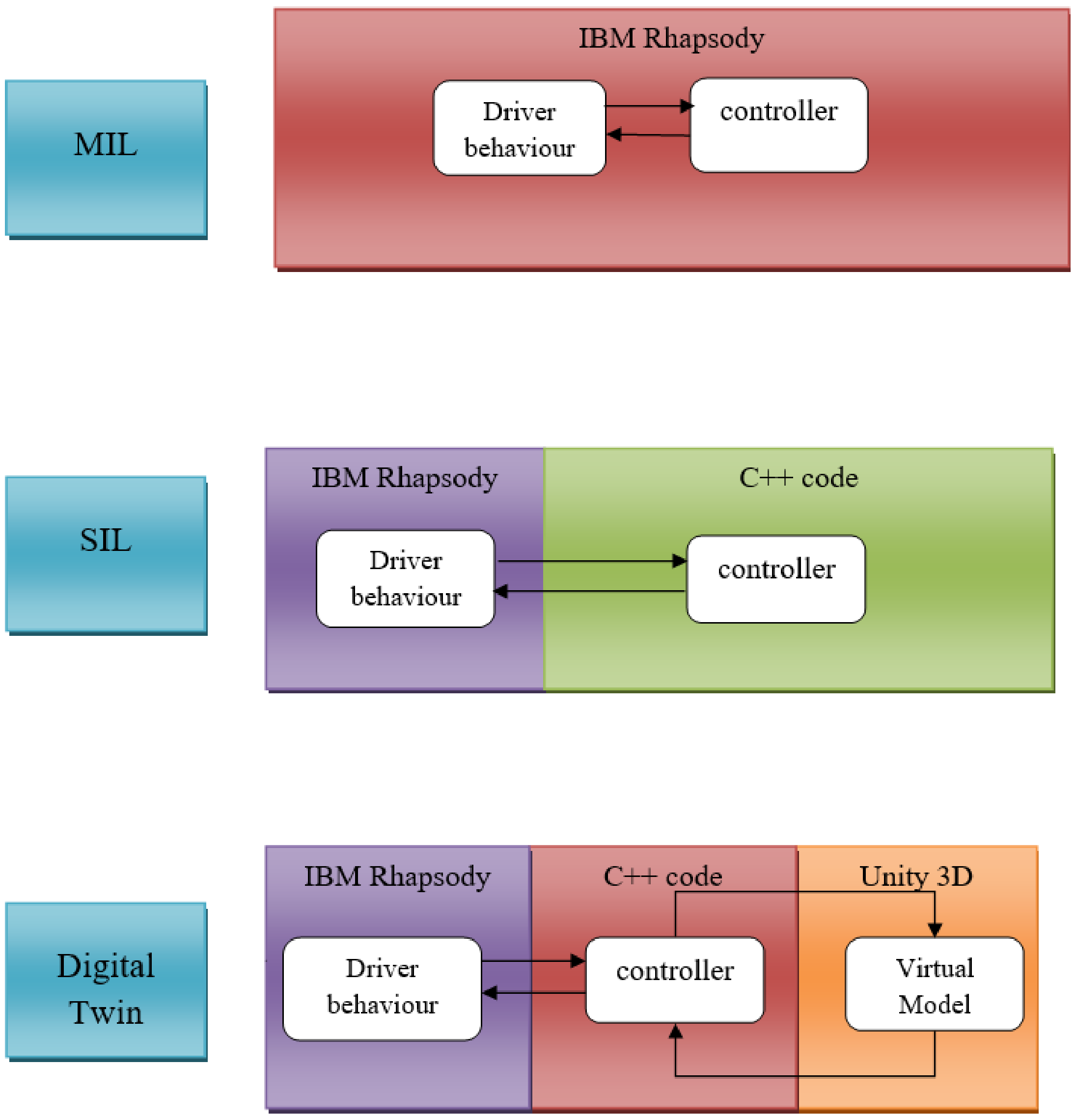


Figure 4.3: Testing Procedure through various stages

4.2 V-Model Project Development Method

The research and development method planned to be followed in this project is the V-Model continuous engineering process depicted in figure 4.4. In this method, a thorough study is performed to understand the feasibility of this project. The Requirements are analysed and captured in Requirement Diagrams as per the SysML model development Method. System Analysis and Design consist of creating use cases for various scenarios of the Smart Meeting Room System (SMR). In the Virtual Analysis and Integration stage, the 3D models created with Unity Scripts are coupled with the SysML profiling and tested for functionality and use cases. The Physical signals from the controller are then integrated with the simulated SMR system. In the Implementation stage, the system is further validated for acceptance. Once, the module is integrated, tested, and implemented on a Raspberry Pi controller, it is now ready for deployment purposes.

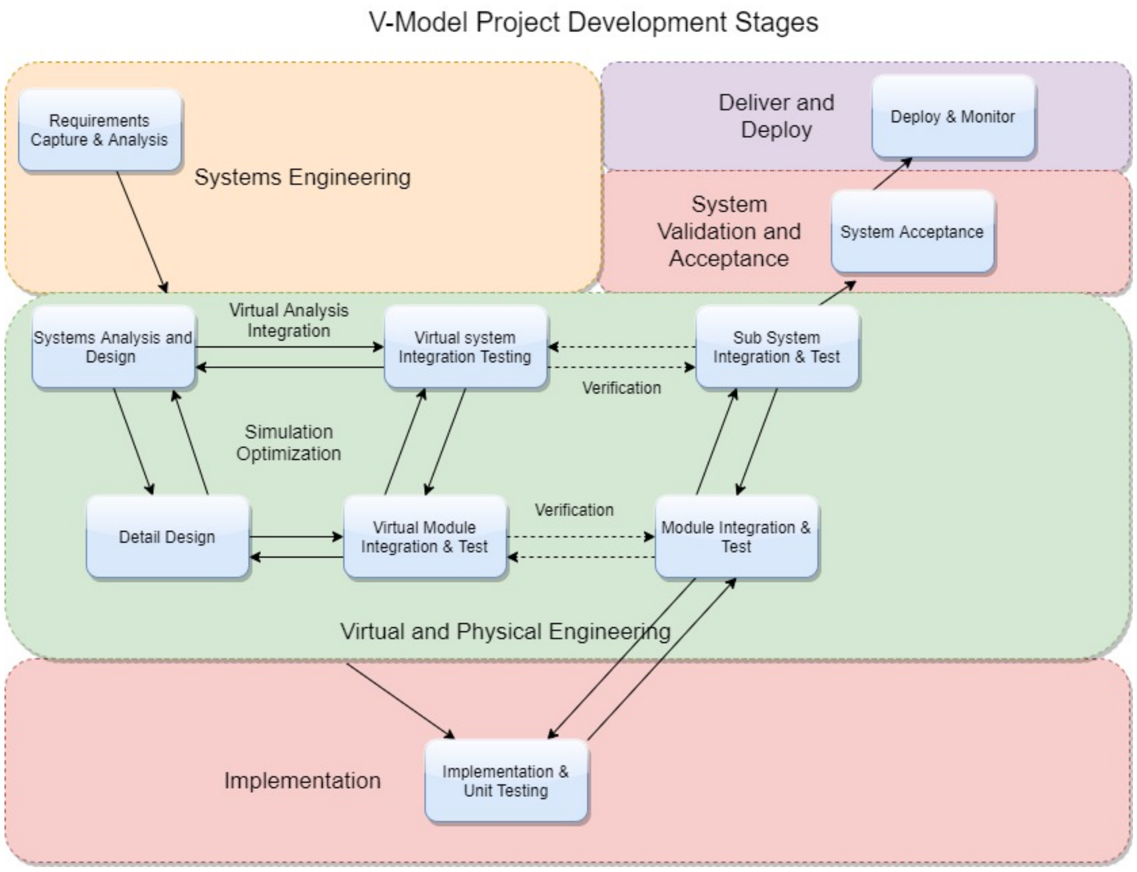
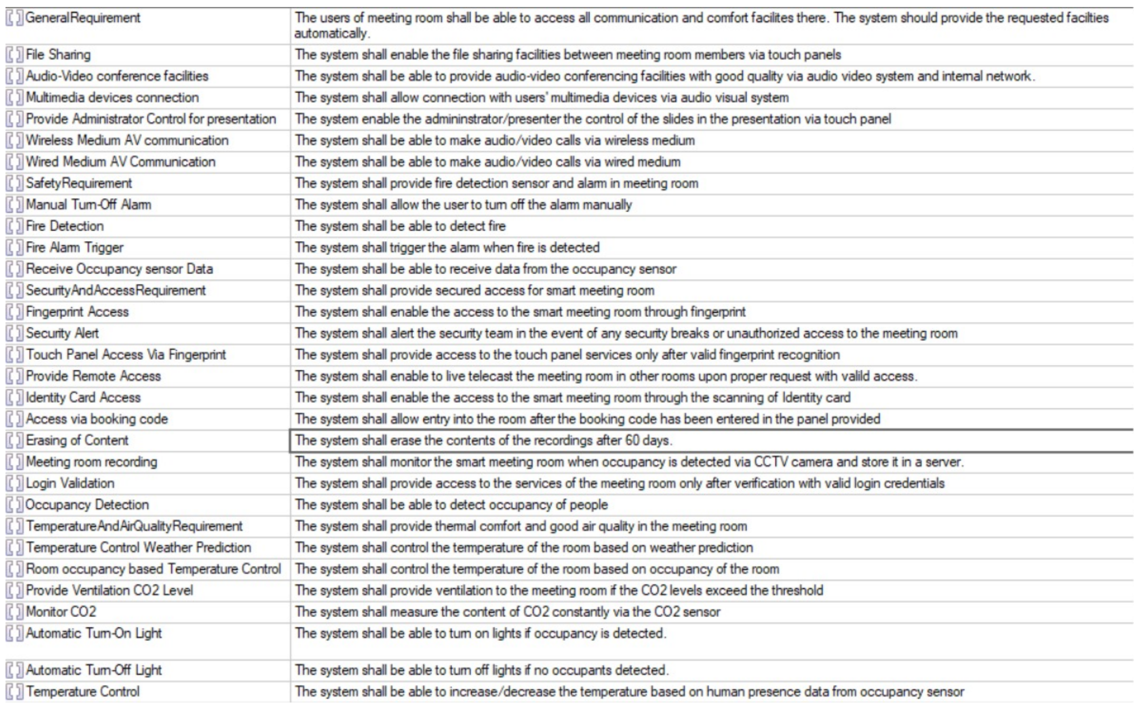

V-Model Project Development Stages
Requirements Capture & Analysis
Systems Engineering
Deliver and Deploy
Deploy & Monitor
System Validation and Acceptance
System Acceptance
Systems Analysis and Design
Virtual Analysis Integration
Virtual system Integration Testing
Sub System Integration & Test
Verification
Simulation Optimization
Detail Design
Virtual Module Integration & Test
Verification
Module Integration & Test
Virtual and Physical Engineering
Implementation
Implementation & Unit Testing

Figure 4.4: V-Model Project Development Stages

Chapter 5

Requirements and system analysis of SMR

As a first step in development process, the project requires SMR system analysis and requirement capturing. In this chapter the SMR system is analysed to capture the suitable requirements. In accordance to the project description discussed in chapter 3, certain system requirements were developed and are listed in the following figure 5.1. Also, the requirements are engineered as per the SysML modelling which answers the research question 1. Furthermore, the additional requirements for SMR system can be found in the appendix A section.



GeneralRequirement	The users of meeting room shall be able to access all communication and comfort facilites there. The system should provide the requested facilties automatically.
File Sharing	The system shall enable the file sharing facilities between meeting room members via touch panels
Audio-Video conference facilities	The system shall be able to provide audio-video conferencing facilities with good quality via audio video system and internal network.
Multimedia devices connection	The system shall allow connection with users' multimedia devices via audio visual system
Provide Administrator Control for presentation	The system enable the admininstrator/presenter the control of the slides in the presentation via touch panel
Wireless Medium AV communication	The system shall be able to make audio/video calls via wireless medium
Wired Medium AV Communication	The system shall be able to make audio/video calls via wired medium
SafetyRequirement	The system shall provide fire detection sensor and alarm in meeting room
Manual Turn-Off Alarm	The system shall allow the user to turn off the alarm manually
Fire Detection	The system shall be able to detect fire
Fire Alarm Trigger	The system shall trigger the alarm when fire is detected
Receive Occupancy sensor Data	The system shall be able to receive data from the occupancy sensor
SecurityAndAccessRequirement	The system shall provide secured access for smart meeting room
Fingerprint Access	The system shall enable the access to the smart meeting room through fingerprint
Security Alert	The system shall alert the security team in the event of any security breaks or unauthorized access to the meeting room
Touch Panel Access Via Fingerprint	The system shall provide access to the touch panel services only after valid fingerprint recognition
Provide Remote Access	The system shall enable to live telecast the meeting room in other rooms upon proper request with valild access.
Identity Card Access	The system shall enable the access to the smart meeting room through the scanning of Identity card
Access via booking code	The system shall allow entry into the room after the booking code has been entered in the panel provided
Erasing of Content	The system shall erase the contents of the recordings after 60 days.
Meeting room recording	The system shall monitor the smart meeting room when occupancy is detected via CCTV camera and store it in a server.
Login Validation	The system shall provide access to the services of the meeting room only after verification with valid login credentials
Occupancy Detection	The system shall be able to detect occupancy of people
TemperatureAndAirQualityRequirement	The system shall provide thermal comfort and good air quality in the meeting room
Temperature Control Weather Prediction	The system shall control the temperature of the room based on weather prediction
Room occupancy based Temperature Control	The system shall control the temperature of the room based on occupancy of the room
Provide Ventilation CO2 Level	The system shall provide ventilation to the meeting room if the CO2 levels exceed the threshold
Monitor CO2	The system shall measure the content of CO2 constantly via the CO2 sensor
Automatic Turn-On Light	The system shall be able to turn on lights if occupancy is detected.
Automatic Turn-Off Light	The system shall be able to turn off lights if no occupants detected.
Temperature Control	The system shall be able to increase/decrease the temperature based on human presence data from occupancy sensor

Figure 5.1: List of all requirements for SMR model

5.1 The analysis of Smart Meeting Room system (SMR) and its requirements

The main focus of the Smart Meeting Room model will be on expressing and recording requirements, design, analysis, and verification information. The SysML approach of modelling the Smart Meeting Room consists of the following steps. a) Classifying the requirements based on the use cases and capturing them in a requirement diagram b) Implementing of use case diagrams for further requirement analysis c) Designing of simulation enabled Sequence diagrams and State machine diagrams to model the dynamic behaviour of the system d) Implementing package diagrams to clearly specify the structure of the system. The SMR system can be represented in a context diagram as follows in figure 5.2.

Components needed in the SMR model

- Intelligent lamps, HVAC system, fire sensors, CO2 sensors and occupancy sensors.
- An audio system that has its speakers enabled for different group settings.
- Security and Access system that grants access with a 3 digit PIN number.

Services needed in the SMR Model

- The system must monitor and control the temperature in the room in an intelligent way.
- The temperature and the lighting in the room must be based on the human presence in the room.
- The user can remotely access all the services provided by the Smart Meeting Room

Classification of Requirements for the SMR model

Based on the purposes and use cases, the requirements can be classified in the following way. All the requirements that follow the below classification are added in the appendix section.

- General Requirements
- Communication Requirements
- Safety Requirements
- Security and Access Requirements
- Temperature and Air Control Requirements

The individual subsystems, its composition of sensors and its association with the central system can be depicted by a domain model diagram as shown in figure 5.3.

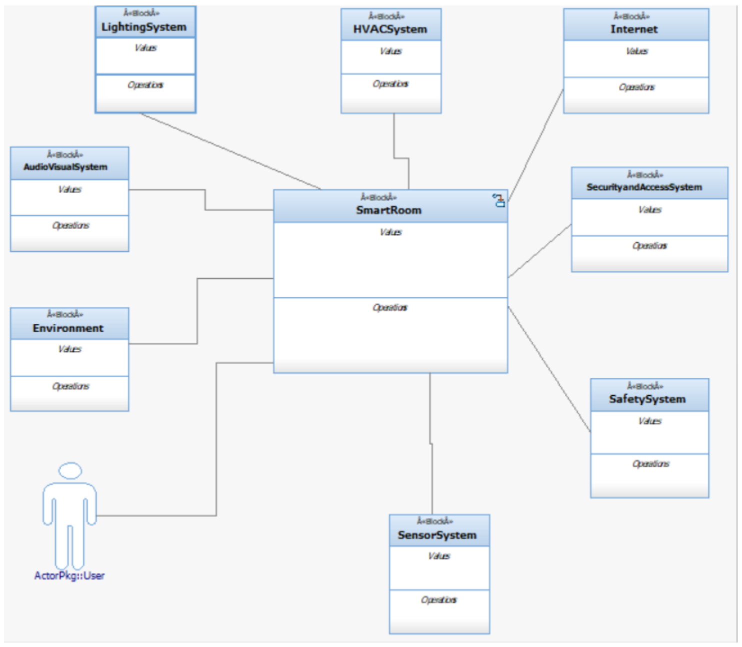


Figure 5.2: Context Diagram for the SMR system with its sub-systems

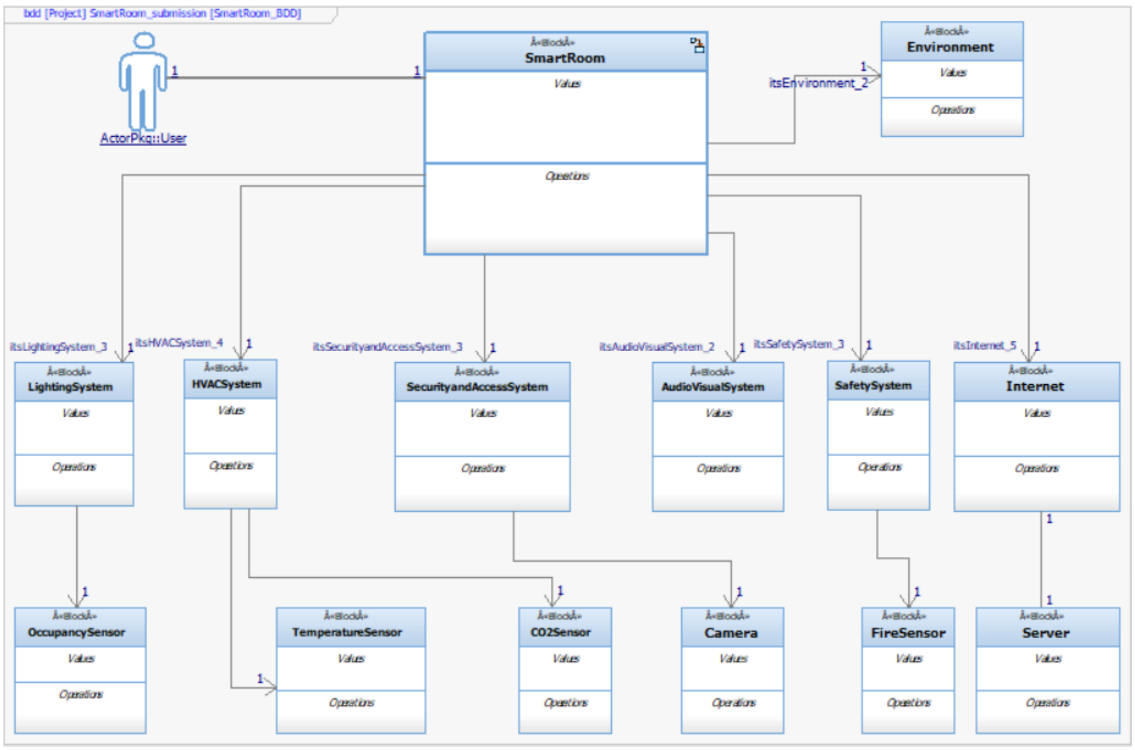


Figure 5.3: Domain model Diagram of the SMR system

5.2 Subsystems of SMR system

5.2.1 HVAC system

An HVAC system is designed to control the environment in which it works. It achieves this by controlling the temperature of a room through heating and cooling. It also controls the humidity level in that environment by controlling the movement and distribution of air in the room. More advanced models will offer variable fan speeds to reduce power usage, however, they remain inefficient when compared to multi-stage systems, and are thus more expensive to run over the long term. HVAC systems can also be built to offer humidity control, and both humidifiers and Dehumidifiers can be added as options for heating and cooling systems. People that live in very dry environments or the tropics find these additions to the system essential. With that being said, some people prefer to install a separate humidifier or dehumidifier systems, so that they can manage the humidity of their environment without also having to turn on the air conditioner.

The three main functionalities of an HVAC system are heating, cooling, and ventilation. The objective of the HVAC system is to ensure that the meeting room environment is both safe and comfortable for humans. Safety mainly concerns with indoor air quality (IAQ). The indoor air should have enough oxygen and be free of noxious gases and this is achieved with the help of inbuilt sensor systems. The sensors that are typically used in an HVAC system are CO2 sensors, temperature sensors, humidity sensors, etc., specifically applying the DT technology and capability to the world of HVAC, Digital Twins can be helpful with [9]:

- Tracking asset performance in real time to plan for maintenance and replacement;
- Training staff to learn key aspects of operation and maintenance much more quickly;
- Real-time decision-making, using repeated simulations and making adjustments to parameters;
- Decommissioning resources using detailed performance records for end-of-life analyses.

5.3 Security and access system

Since the information from the environment and various multimedia are fused together with futuristic technologies such as smart cities and autonomous cars, the demand for higher value-added spatial information contents and the necessity of technology for spatial information security are increasing. However, there is a problem with unreliable or costly to modify security policy. Such problems occur frequently in the process of coordination or integration of the information management systems that are used in public institutions and private companies. Therefore, it is necessary to maintain an updated security and access system that can adapt to future modifications. The security and access system considered in this project has the following features for proper maintenance and provide security. The entry and exit from the room are detected by an occupancy sensor. It authorizes the users after logging in to their personal account. The CCTV camera helps record data for security purposes.

5.4 Lighting system

Lighting controls are the means to provide lighting functionality and saves lighting energy. The strategies mentioned below (or a combination of strategies) are often incorporated to realize a compliant design [10].

Time/astronomical scheduling:

Lighting in a defined area turns on or off, or dims, based on a predetermined, customizable schedule.

Occupancy/vacancy control:

In each area of the building, lighting is turned on, off, or to a predetermined level based on occupancy level detected. With "vacancy" control lights are automatically turned off when space becomes vacated. The users no longer need to manually turn on the light. Often, room conditions may not require electric lighting, as in the case of a day-lit office. In this type of situation, vacancy sensors are able to save energy that would have otherwise been automatically consumed if an occupancy sensor forced the lights on. For this reason, vacancy sensors are generally considered more efficient than, therefore preferable to, occupancy sensors, driving the inclusion of prescriptive vacancy sensors in many versions of updated electrical codes.

Daylight harvesting:

Light levels are manually and/or automatically adjusted based on the amount of natural light in a space. Appropriate light levels are provided for functional purposes, and total illumination is maintained throughout the space. Different ways of harvesting daylight include continuous dimming of the lighting systems (the most efficient option), bi-level or multi-level zone dimming or switching, or simple on/off controls provided for lighting zones where ample daylight is expected.

5.5 Safety system

A typical fire alarm system has a number of devices that work synchronously to warn and alert people through visual and audio appliances when smoke, fire, carbon monoxide, or other emergencies are detected. The system can be activated automatically from smoke detectors and heat detectors or may also be activated alternatively using manual fire alarm activation devices such as manual call points or pull stations. Functionalities of the safety system include alerting the users of a potential fire hazard and when the Co2 levels have reached beyond the permissible limit. The alarms can be of the form of speakers that sound an alarm and gives an evacuation message to not use the elevators. Fire sensors and Co2 sensors are inbuilt in this system.

5.6 Audio System

The audio system for the meeting room needs to be designed in a way that satisfies the users and moreover complies with the comfort and efficiency concerns. The automatic control for speakers are made available using a central intelligence that suits for different kinds of group settings. Additionally, there is manual control to turn on/off, increase/decrease volume, place video calls, and share screen. The room has facilities to conduct meetings that require video conferencing to a small group of people. It requires internet connectivity.

Chapter 6

Design of SMR system and its Digital Twin

6.1 System dynamics and control

The smart meeting room as a system operates in different modes that suit the conditions of the environment, adjusting automatically with the ambience.

The control of the system has to be clearly defined to ensure that the system meets the safety and comfort needs of the users. The various sub-systems interact and function together in an intelligent and efficient way. To define a central intelligence and determine the nature of the interaction between multiple functionalities and subsystems, it is essential to identify and segregate the different states of the system based on the needs of the users.

6.1.1 Modes of the SMR system

Based on the requirements, the states of the SMR system can be classified as idle, sleep, meeting, fire alarm. Each of the states is explained and depicted pictorially in this section. The figures show at each state of the system, the corresponding sub-system/component behaviour.

Idle

When the meeting room is in the Idle state the lights, speakers, CCTV cameras are all turned off and remains off until the state changes. The doors remain closed during this period. However, the HVAC system needs to be running in order to maintain the temperature level in the room. It is not possible to control the HVAC system in this state, hence the functioning of the HVAC system will be based on some predefined schedule or according to the weather prediction data. The system enters idle mode automatically when no human presence is detected for a specific duration of time. Alternatively, the meeting room can be manually controlled to enter into the Idle state by a remote controller. The overall behaviour and the state of the subsystems in Idle mode is depicted in figure 6.1.

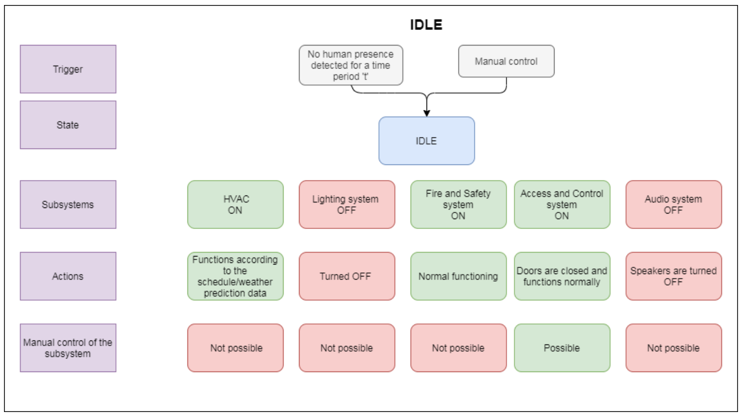

IDLE
Trigger
State
Subsystems
Actions
Manual control of the subsystem
No human presence detected for a time period 't'
Manual control
IDLE
HVAC ON
Lighting system OFF
Fire and Safety system ON
Access and Control system ON
Audio system OFF
Functions according to the schedule/weather prediction data
Turned OFF
Normal functioning
Doors are closed and functions normally
Speakers are turned OFF
Not possible
Not possible
Not possible
Possible
Not possible

Figure 6.1: The state/behaviour of sub-systems in the control mode - Idle

Sleep

When the meeting room is in sleep state all the devices and subsystems are turned off including the HVAC system. If the users decide to lock down the meeting room with all its functionalities being turned off, they can do so manually using the remote controller. The meeting room will automatically go into the Sleep state if it remains in the idle state for a long period of time. The temperature in the room is no longer need to be maintained, based on the assumption that the room needs no cooling/heating/ventilation and the room remains in the ideal temperature. The overall behaviour and the state of the subsystems in Sleep mode is depicted in figure 6.2.

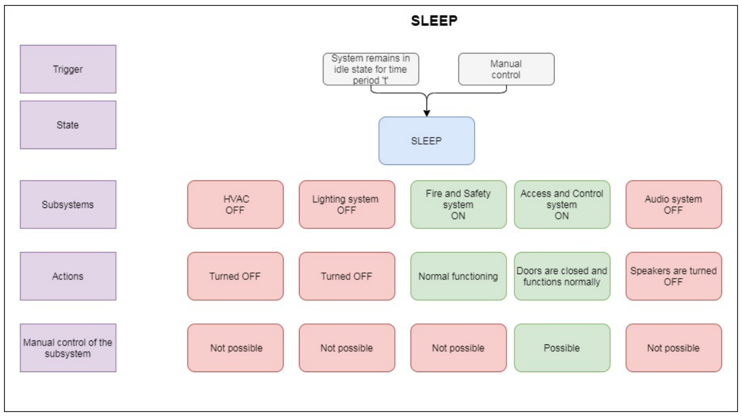


Figure 6.2: The state/behaviour of sub-systems in the control mode - sleep

Meeting

If the room needs to be prepared for the meeting, the users can manually enter into Meeting state. To facilitate the meetings, the lighting of the room needs to be set in an appropriate way. By default, the lighting setup goes into meeting mode. However, it can be toggled between intelligent, normal and meeting. The HVAC system functions as per the schedule/ weather prediction data, the doors remain closed, the CCTV cameras remain turned ON, the and speakers are turned on for any audio/video output. The overall behaviour and the state of the subsystems in Meeting mode is depicted in figure 6.3.

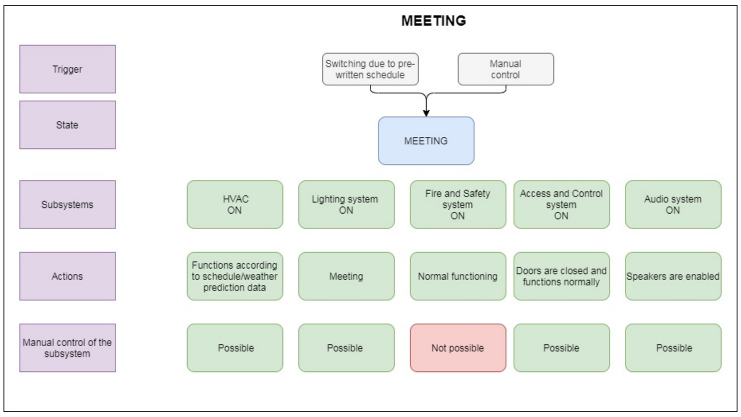

MEETING
Trigger
State
Subsystems
Actions
Manual control of the subsystem
Switching due to pre-written schedule
Manual control
MEETING
HVAC ON
Lighting system ON
Fire and Safety system ON
Access and Control system ON
Audio system ON
Functions according to schedule/weather prediction data
Meeting
Normal functioning
Doors are closed and functions normally
Speakers are enabled
Possible
Possible
Not possible
Possible
Possible

Figure 6.3: The state/behaviour of sub-systems in the control mode - Meeting

Fire alarm

When the fire sensors/CO2 sensors have detected an emergency situation, the meeting room enters into a Fire alarm state. In this state, a warning message is announced through the speakers as an alarm followed by alarm sounds. The doors will remain open, the HVAC system will be turned off, the lighting system will be turned off after a short period of time to allow users to leave the room. The CCTV cameras and the touch panels are turned off. All the sub-systems will immediately be turned off in this state. The Fire alarm state of the meeting room cannot be triggered manually. It is to be noted that the fire alarm and CO2 sensors will remain online throughout all of the states of the system. The overall behaviour and the state of the subsystems in Fire alarm mode is depicted in figure 6.4.

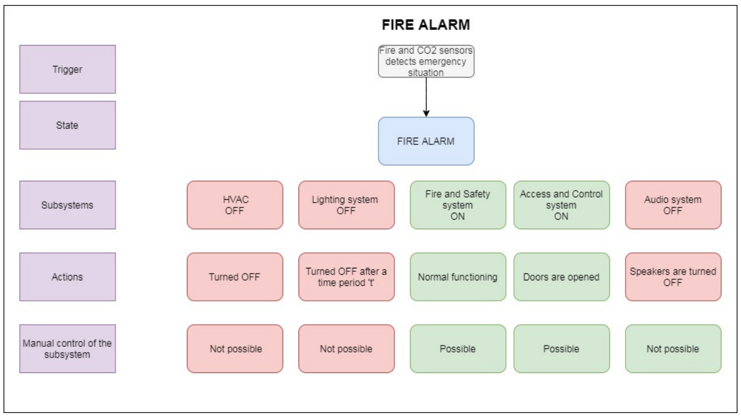


Figure 6.4: The state/behaviour of sub-systems in the control mode - Fire alarm

6.2 Behaviour of the subsystems (devices and drivers)

A representation of the behaviour of the HVAC system, lighting System, Safety System, Security and Access system, and the Audio Visual System are depicted as services using use-case diagrams. This is a part of the SysML agile development of models for the Smart Meeting Room. Some of the subsystems mentioned below are clubbed together to create a generic use case that serves multiple purposes.

6.2.1 HVAC and Safety System

In figure 6.5, we see the use case diagram for the service of HVAC and safety system combined. The user can turn on/off the HVAC system and manually control the room temperature. The HVAC system has functions of heating, Air conditioning, and providing ventilation to the room. The central system which has inputs from the occupancy sensor detects the increased human presence and provides ventilation and air conditioning automatically. Whenever the safety system detects fire through its fire sensor it triggers the fire alarm and therefore the HVAC system is turned off automatically. The Carbon-dioxide sensors in the safety system check for the permissible limits of CO2 and triggers a warning message whenever the threshold is reached.

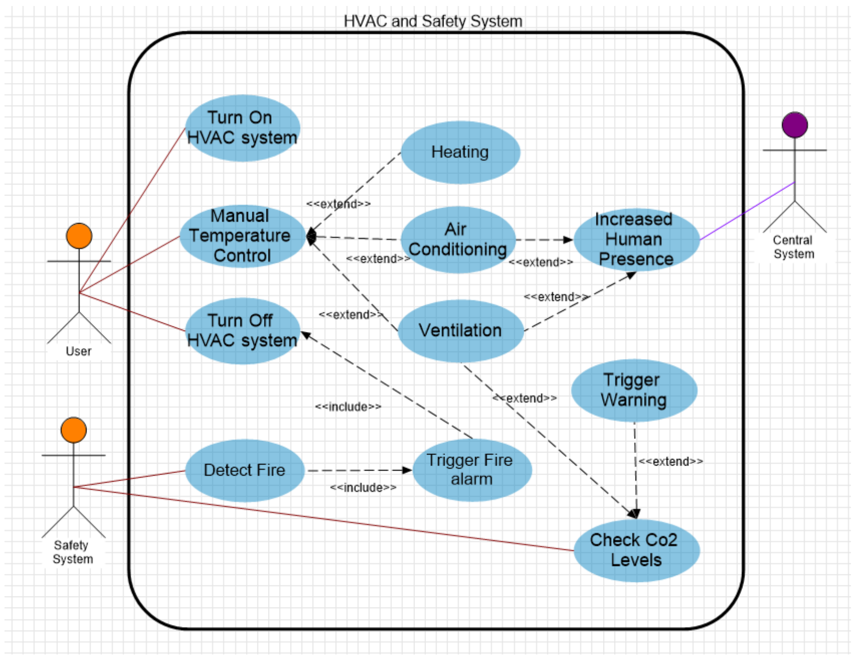


Figure 6.5: HVAC and safety system

6.2.2 Lighting System

The lighting system service is represented in a use case diagram in figure 6.6. The user can manually turn on the lights and adjust the brightness or the central system can detect the entry and exit at the door through its occupancy sensor and turn on/off the lights whenever needed. Whenever the audio and the visual system does the screen sharing function, the brightness of the room has to go down so as to provide a better display and viewing for the occupants.

6.2.3 Security and Access System

The security and access system of the smart room provides an option for the users to login with a password. The entry to the room can be restricted to a limited number of people. The Central system constantly monitors for entry at the door. Whenever there is an entry at the door, the CCTV camera is turned on automatically. This is done to monitor the room events for security purposes. The service of a security and access diagram is represented as a use case diagram in figure 6.7.

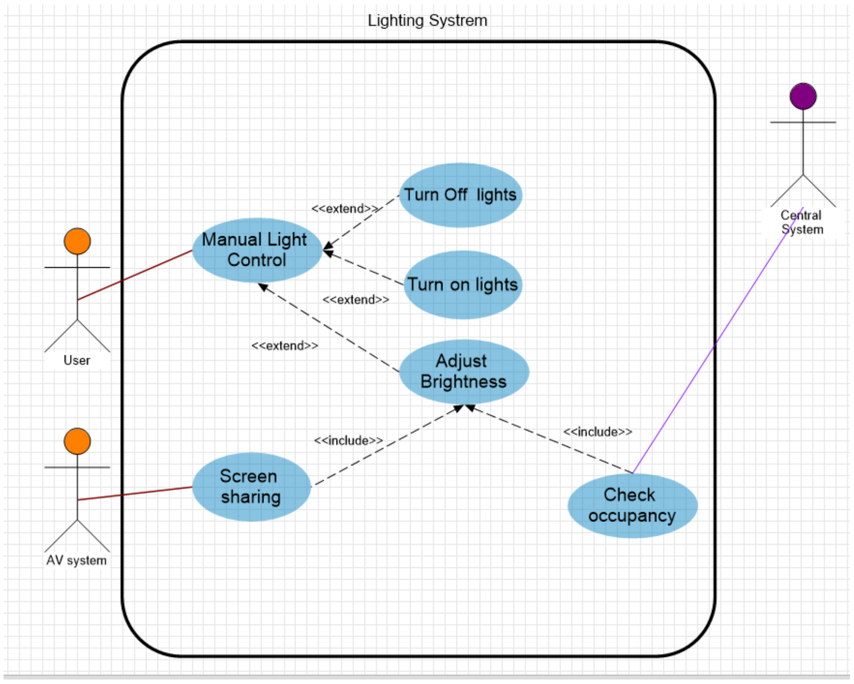


Figure 6.6: Lighting System

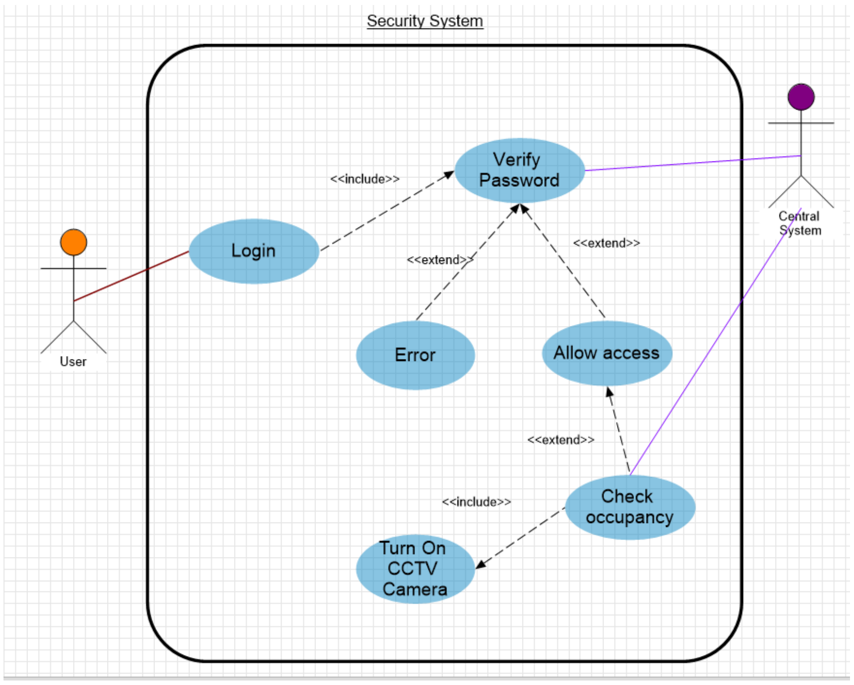


Figure 6.7: Security and access system

6.2.4 Audio and visual system

In figure 6.8, the service of Audio and Visual system is depicted. The touch panel in the smart room allows users to interact by logging in to their personal account. The touch panel has options to make conference calls and allow screen sharing.

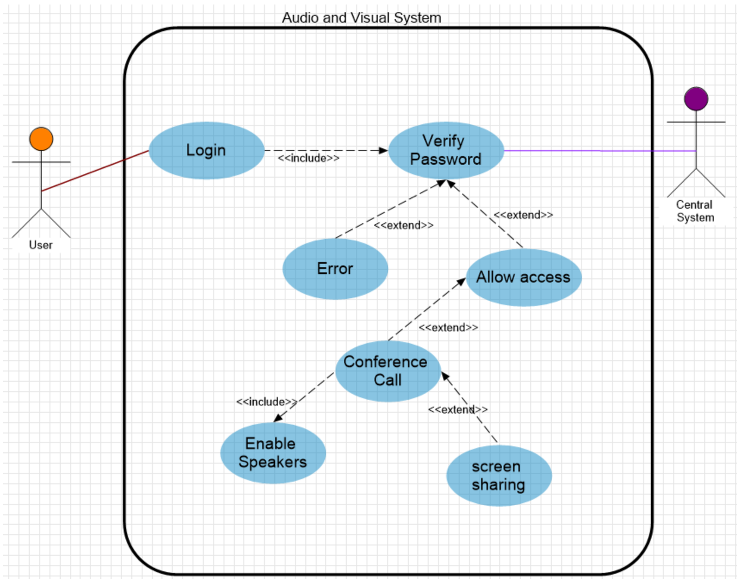


Figure 6.8: Audio and Visual System

6.3 The Layered Architecture for the Digital Twin of Smart Meeting Room system (SMR)

The layered architectural approach previously discussed in section 3.2.2 can be established in this project to answer the research question 2. In this context, developing a virtual space in Unity Game Engine can be accomplished with the help of the principles of Digital Twin layered architecture. In this section, adopting the SMR system to each layers of Digital Twin architecture is discussed.

6.3.1 *Data Abstraction Layer*

All the Data that are collected from different components and its sensors are encapsulated in this layer. The SysML models created using IBM Rhapsody contain all the data in this layer.

General Input Data variables of the SMR model

- Movement detection flag
- Occupancy detection flag

Input Data variables from HVAC system

- Current Room Temperature
- Current outside Temperature
- Current CO2 level
- Current Humidity level
- Weather prediction data

Input Data variables from Safety system

- Fire Alarm flag
- CO2 alarm flag

Input Data variables from the Lighting system

- Room Illumination index
- Outside illumination index
- Window panel adjustment

Input Data variables from Security and Access system

- PIN number
- CCTV flag

Input Data variables from Audio and Visual system

- Conference flag
- Speaker flag
- User ID

6.3.2 *Transmission Layer*

All the data from the sensors and sub-systems which are collected in the Data acquisition layer is now transmitted to the controller domain. In this domain, the data is manipulated to offer the services of the SMR system. Although, the physical inputs from the sensors are simulated using SysML models and user inputs, there is need for a transmission layer because, it acts as a bridge between the IBM Rhapsody model and 3D Virtual model. The Data values from the IBM model is transmitted to the Unity Game Engine using a suitable communication protocol. Therefore, a separate service that transmits the input data from the SysML model is necessary to control the SMR system in an organised way.

6.3.3 *Digital Modelling Layer*

The Smart Meeting Room with all its components are modelled in this layer. The free asset store in Unity game engine is used for this purpose. The 3D models of Meeting Room background and components are to be developed from available templates. Furthermore, the C# scripts help in developing a game like scenario to generate player movements inside the SMR system application.

6.3.4 *Data/Model Integration Layer*

In this layer, the C# scripts for the virtual model in Unity is designed as per the system behaviour and bounded together with the C++ code that is auto generated from the IBM Rhapsody models. The integration should be in such a way that it offers flexibility and adaptability to the code.

6.3.5 *Application/Service Layer*

Once the SysML model is integrated with the 3D model, the data needs to be manipulated and accessed in an intelligent way. The sequencing of the services needs to be in a proper way to facilitate real case scenarios.

Chapter 7

SMR system and Digital Twin Implementation

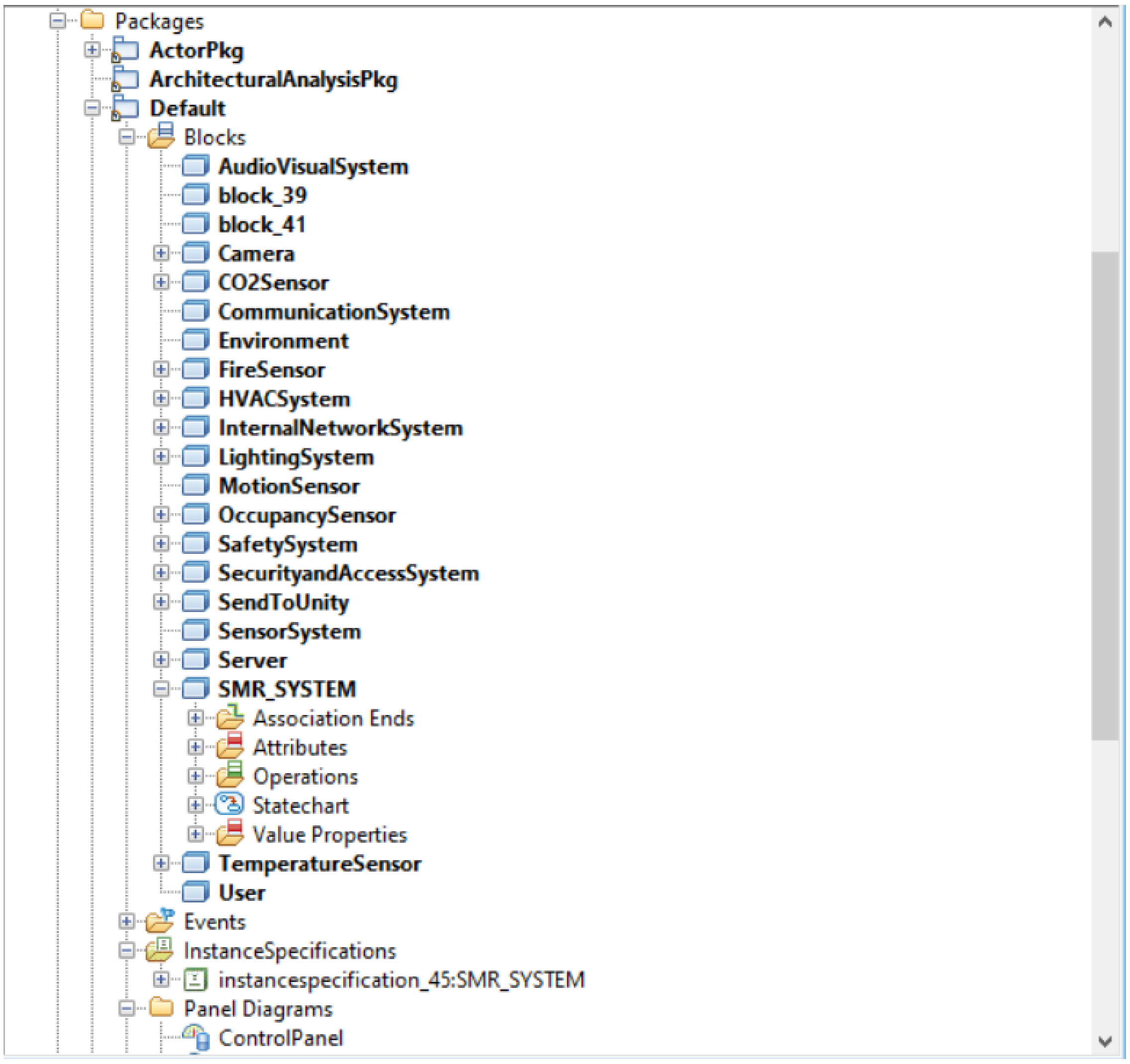
The project is implemented in the following 3 stages. Firstly, The SMR system is modelled in IBM Rhapsody followed by Virtual model integration in Unity Game Engine. Finally, the coupled model is made portable to a linux environment for HIL implementation on a raspberry pi device connected remotely. In this chapter, section 7.1 discusses the IBM model implementation followed by its communication to Unity and Implementation of Digital Twin in Unity Game Engine in subsequent sections. Finally, section 6.3 discusses the raspberry pi adaptation of the integrated SMR system and Digital Twin.

7.1 IBM Model Implementation

IBM model implementation is the first and foremost step towards the project. The project description established in chapter three provides us with necessary information to design a suitable SMR system in consideration. The IBM Rhapsody developer 8.4 edition together with Visual Studio 2017 Community edition is used as the modelling tool for this project. Further, In the IBM tool, a new C++ project is created with the default project configurations and compilation instructions specific to a windows system that uses x86 processor.

Firstly, The requirements are modelled in a separate package named 'RequirementAnalysisPkg' which is added in the appendix A - SMR system Requirements. Secondly, the system context and block definitions diagram together with the class diagram of the SMR model are created as shown previously in figures 5.2 and 5.3. These diagrams help in defining the blocks and components needed for a Typical SMR system. Moreover, the diagrams establishes a relationship between the subsystems and its respective components. As a next step, we maintain a file structure such that the individual components are made as object blocks for the package named 'default'. This is represented in the following figure 7.1. each of these objects are listed in the appendix section. Thus, in an effort To make use of the object oriented programming in C++, the subsystems and its components are made available as objects which further contains associations, attributes, operations, state transition charts and value properties to execute and simulate the application for different use case scenarios.

Once the objects together with its respective attributes and operations are defined, we move on to implementing the system dynamics and control. The SMR system is operated in five modes namely, Idle, Online, Sleep, Meeting, Fire/Emergency as mentioned in section 6.1.1. However, to implement these modes, it is essential to make use of the state transition diagrams and events. For instance, in the event of absence of humans inside the room for a time period of 't' through



```
Packages
  ActorPkg
  ArchitecturalAnalysisPkg
  Default
    Blocks
      AudioVisualSystem
      block_39
      block_41
      Camera
      CO2Sensor
      CommunicationSystem
      Environment
      FireSensor
      HVACSystem
      InternalNetworkSystem
      LightingSystem
      MotionSensor
      OccupancySensor
      SafetySystem
      SecurityandAccessSystem
      SendToUnity
      SensorSystem
      Server
      SMR_SYSTEM
        Association Ends
        Attributes
        Operations
        Statechart
        Value Properties
      TemperatureSensor
      User
    Events
    InstanceSpecifications
      instancespecification_45:SMR_SYSTEM
    Panel Diagrams
      ControlPanel
```

Figure 7.1: Project File structure in IBM Rhpasody Tool

the motion detection sensors in the system, the system goes into 'Idle' state. similarly, each of the mode has its own trigger and set of actions to perform and therefore the corresponding design of state transition diagram is shown in the figure 7.2. Furthermore, The state change must be designed in such a fashion that it not leads to a deadlock situation. In addition, The overall process of the SMR system can be depicted with the help of a sequence diagram as shown in figure 7.3.

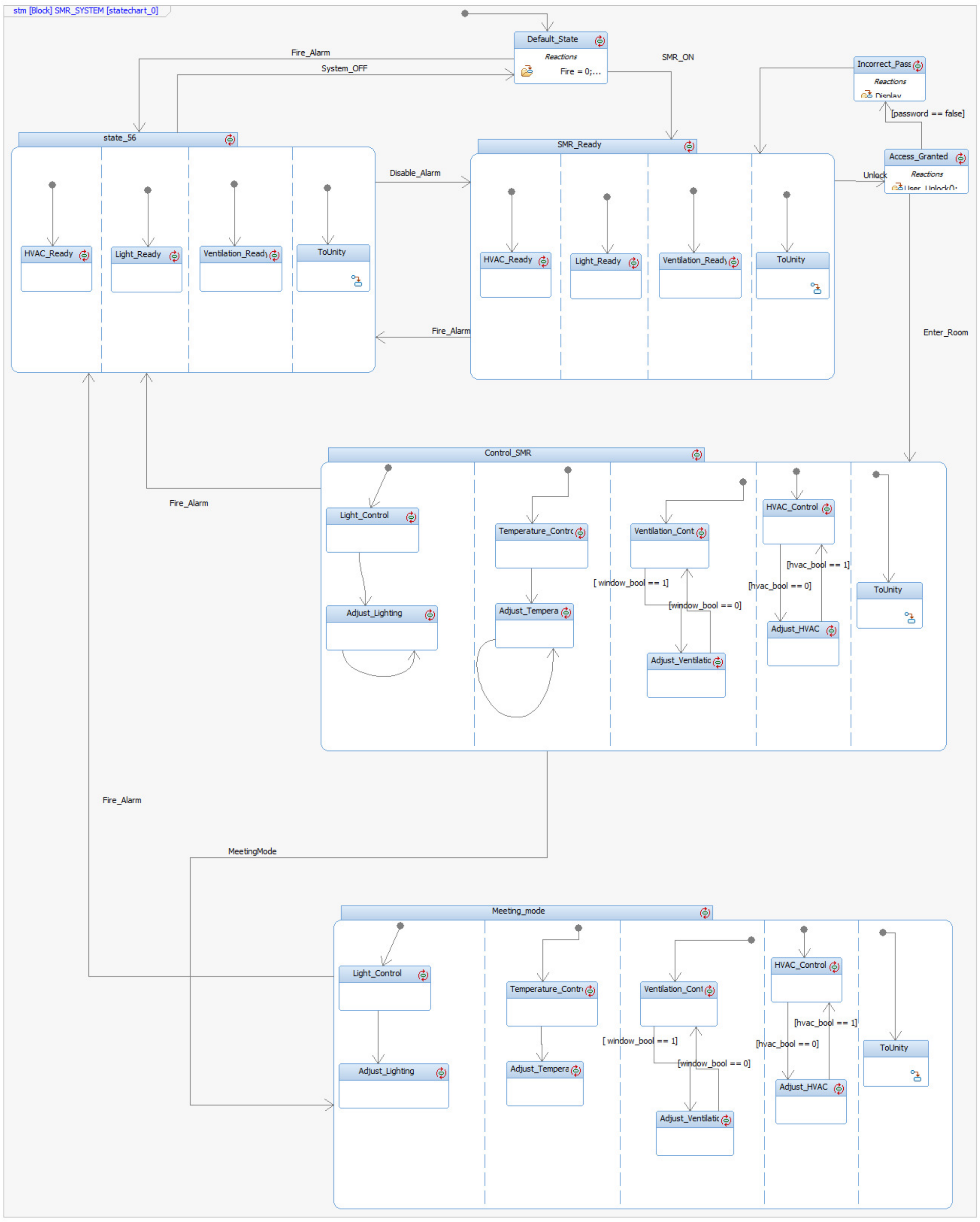


Figure 7.2: SMR system state machine diagram

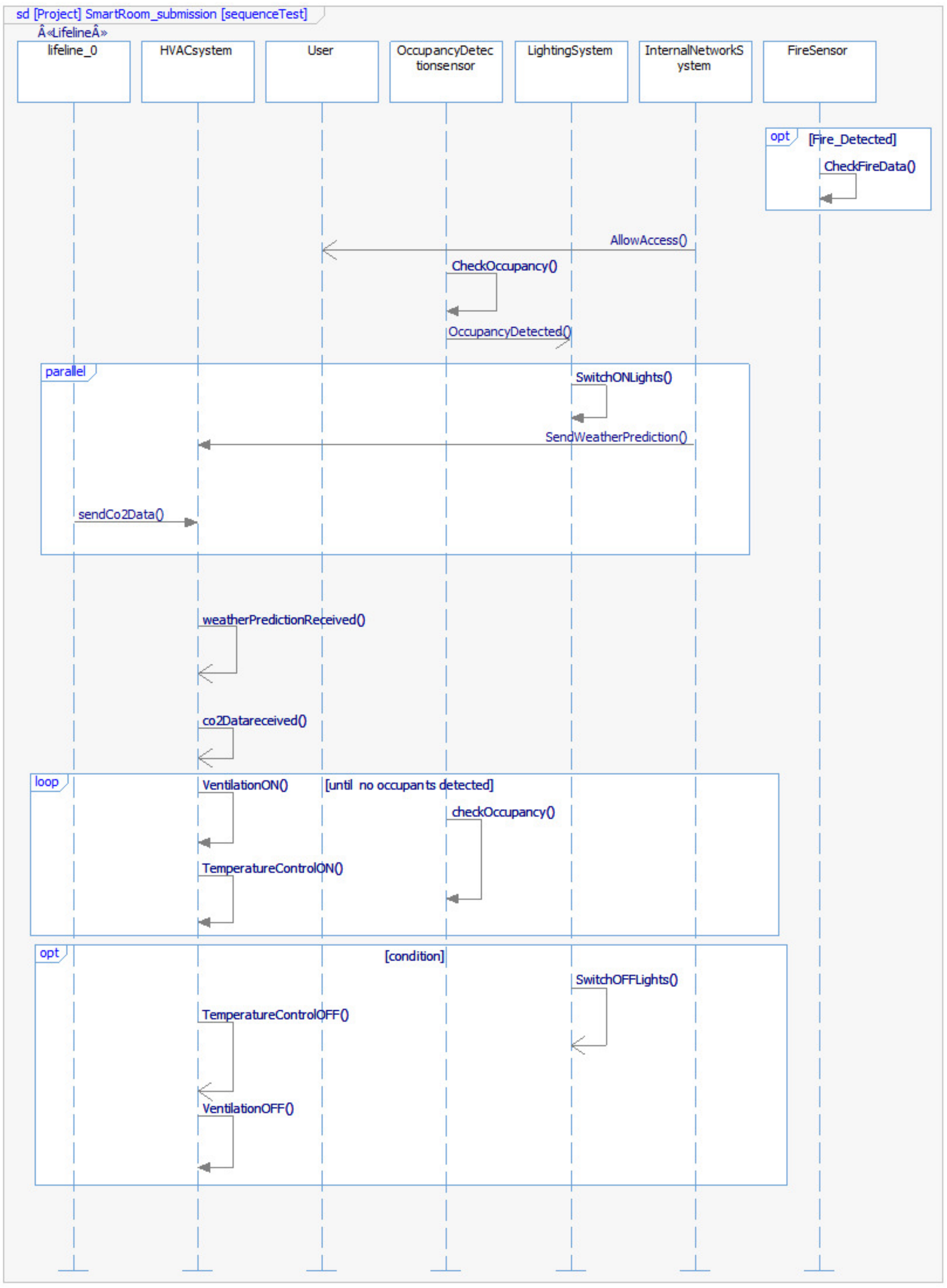


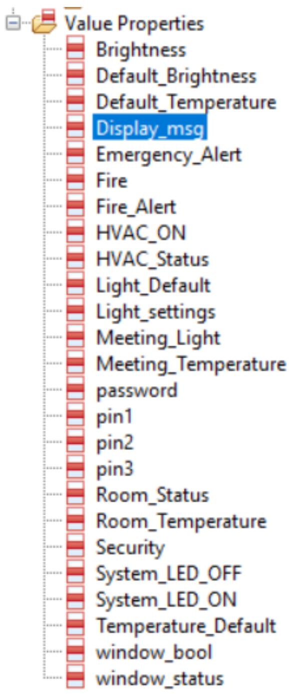
Figure 7.3: SMR system sequence diagram

7.2 Communication to Unity

The SMR model implemented in IBM Rhpasody needs to communicate with the virtual model of the SMR system in Unity Game Engine. For this purpose, the individual subsystems such as the HVAC system, Fire and safety system, lighting system, access and control system, and audio and visual system needs to update its status variable and actuate a set of actions in the Unity Game Engine. These set of actions are based on the user input that determines the scenario and the control state in which the SMR operates. As far as the communication method is concerned, network socket programming enables sufficient and flexible communication of raw data between applications. In this section, the GUI display of SMR system, the block that implements the socket communication in the SMR model -'SendToUnity', and the attributes and data values needed for successful communication are further discussed in detail.

7.2.1 Modelling the User Input and display

The touch panel acts as a GUI for the SMR system. every buttons, knobs, display and switches in the control panel are bound to an event or an attribute in the instance specification of the SMR system. The Touch panel diagram that takes input from the user is modelled as shown in figure 7.5. the value properties that bind to the GUI are shown in the below figure 7.4.



Value Properties
- Brightness
- Default_Brightness
- Default_Temperature
- Display_msg
- Emergency_Alert
- Fire
- Fire_Alert
- HVAC_ON
- HVAC_Status
- Light_Default
- Light_settings
- Meeting_Light
- Meeting_Temperature
- password
- pin1
- pin2
- pin3
- Room_Status
- Room_Temperature
- Security
- System_LED_OFF
- System_LED_ON
- Temperature_Default
- window_bool
- window_status

Figure 7.4: The value properties that bind to the Touch panel diagram

7.2.2 Modelling the 'SendToUnity' block

The 'SendToUnity' block of the SMR model is the most essential part of this project and helps in accomplishing the fundamental objective by integrating the SMR system with the Digital Twin. This feature answers the second part of the research question 2. Since 'SendToUnity' block is a salient feature that communicates with the virtual model back and forth, it is an important step in the model-driven system engineering approach. It offers the necessary output ports enabled in

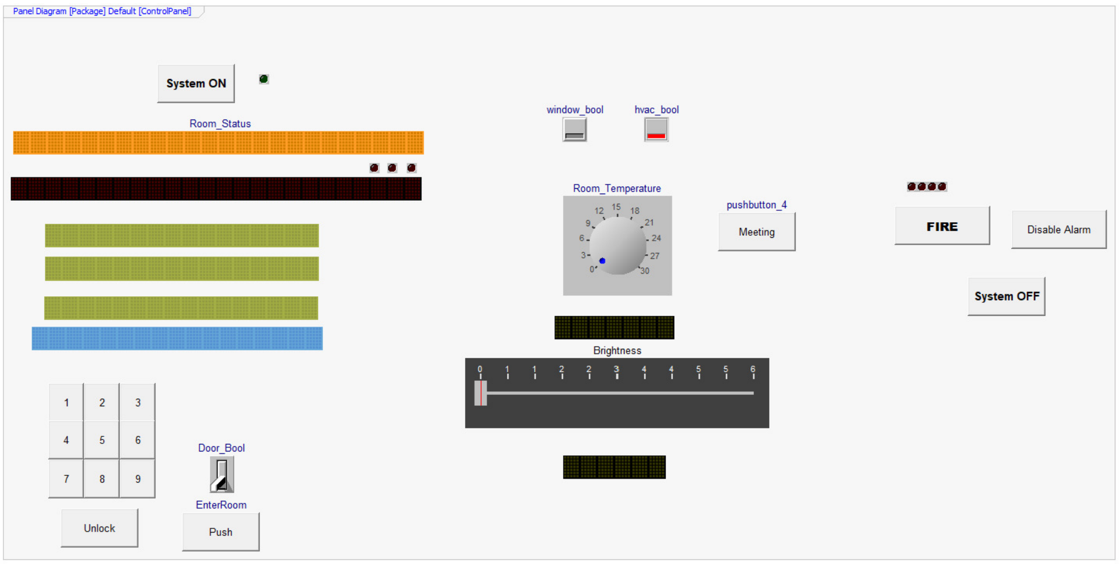


Figure 7.5: Touch Panel Diagram acting as GUI display for SMR system

Unity Game Engine and acts as a testing method to verify the SMR model in the Virtual space. The state transition diagram of the 'SendToUnity' block is shown in the below figure 7.5. The variables in the block are auto updated with a time interval of 5000ms as portrayed in fig 7.6. However, the 'CallMain()' function that receives and sends data to and from Rhapsody tool is called recursively with a time delay of 1000ms.

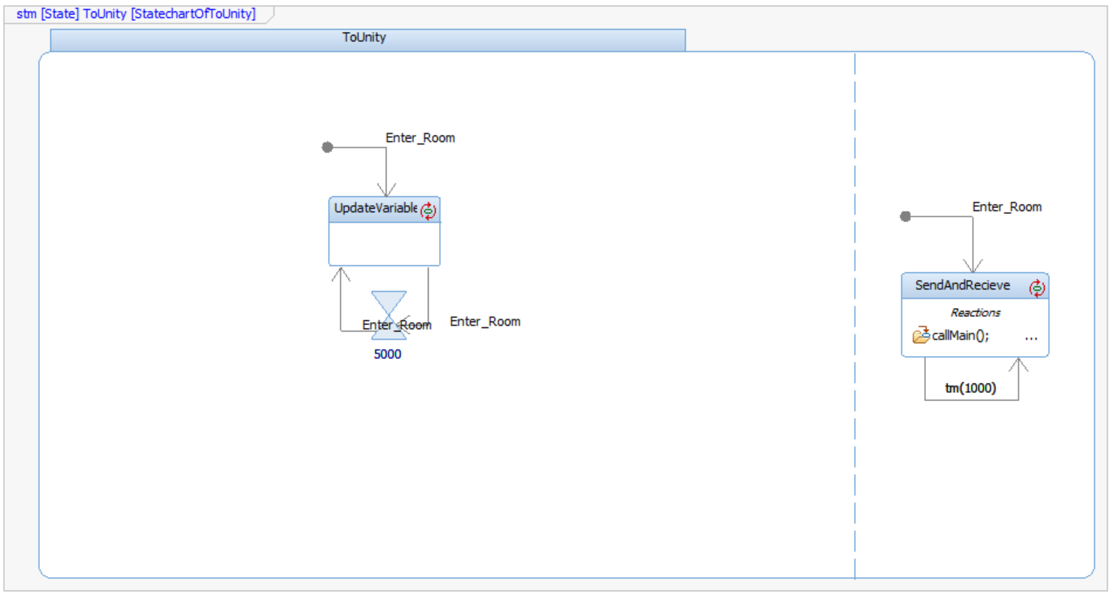
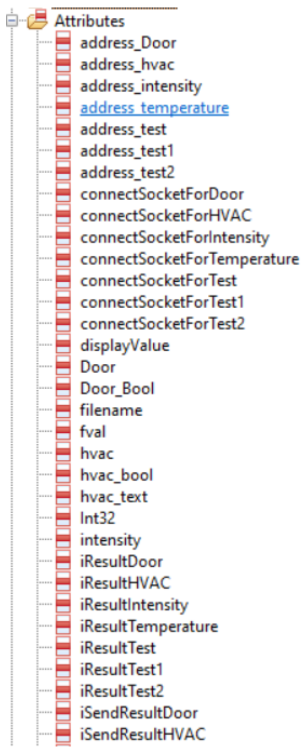


Figure 7.6: State transition diagram of 'SendToUnity' block

The SMR model in the IBM Rhapsody and virtual model in Unity Game Engine do run on the same windows machine in this implementation that has been demonstrated in this report. However, it is important to note that it is possible to establish such communication between two processes even while running on two different machines provided they operate in the same network.

Since network socket communication makes use of TCP/IP communication protocol, it becomes easier to implement the functionality in the IBM Rhapsody tool. The 'WINSOCK' library present in the C++ software development environment has the structure and predefined functions for enabling Socket communication in IBM Rhapsody tool. The raw data and string values that needs to be sent/received are given separate individual ports. In addition, The 'SendToUnity' block essentially acts as a client that wants to connect to specific ports on the server that runs on Unity Game Engine. The code for TCP client functionality is attached in the appendix B section. The attributes that are needed for establishing the network socket communication are shown in following figures 7.7 and 7.8.



Attributes
- address_Door
- address_hvac
- address_intensity
- address_temperature
- address_test
- address_test1
- address_test2
- connectSocketForDoor
- connectSocketForHVAC
- connectSocketForIntensity
- connectSocketForTemperature
- connectSocketForTest
- connectSocketForTest1
- connectSocketForTest2
- displayValue
- Door
- Door_Bool
- filename
- fval
- hvac
- hvac_bool
- hvac_text
- Int32
- intensity
- iResultDoor
- iResultHVAC
- iResultIntensity
- iResultTemperature
- iResultTest
- iResultTest1
- iResultTest2
- iSendResultDoor
- iSendResultHVAC

Figure 7.7: Attributes needed for communication in 'SendToUnity' block

7.2.3 Model compilation and execution

Once the SMR model is designed, it is necessary to check for model errors. The model free from all errors is able to auto generate the C++ source files, header files and makefile required for compiling and executing the project. The framework for the default c++ configuration needs to be built and compiled. The framework will build the debug information files for the library. Running the executable file enables the animation toolbar. The demonstration and execution of the project will be discussed stepwise in chapter 8.

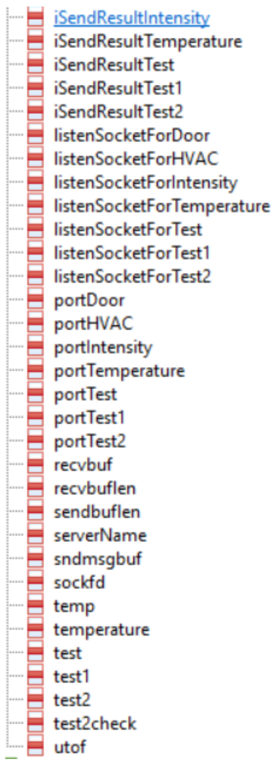

iSendResultIntensity
iSendResultTemperature
iSendResultTest
iSendResultTest1
iSendResultTest2
listenSocketForDoor
listenSocketForHVAC
listenSocketForIntensity
listenSocketForTemperature
listenSocketForTest
listenSocketForTest1
listenSocketForTest2
portDoor
portHVAC
portIntensity
portTemperature
portTest
portTest1
portTest2
recvbuf
recvbuflen
sendbuflen
serverName
sndmsgbuf
sockfd
temp
temperature
test
test1
test2
test2check
utof

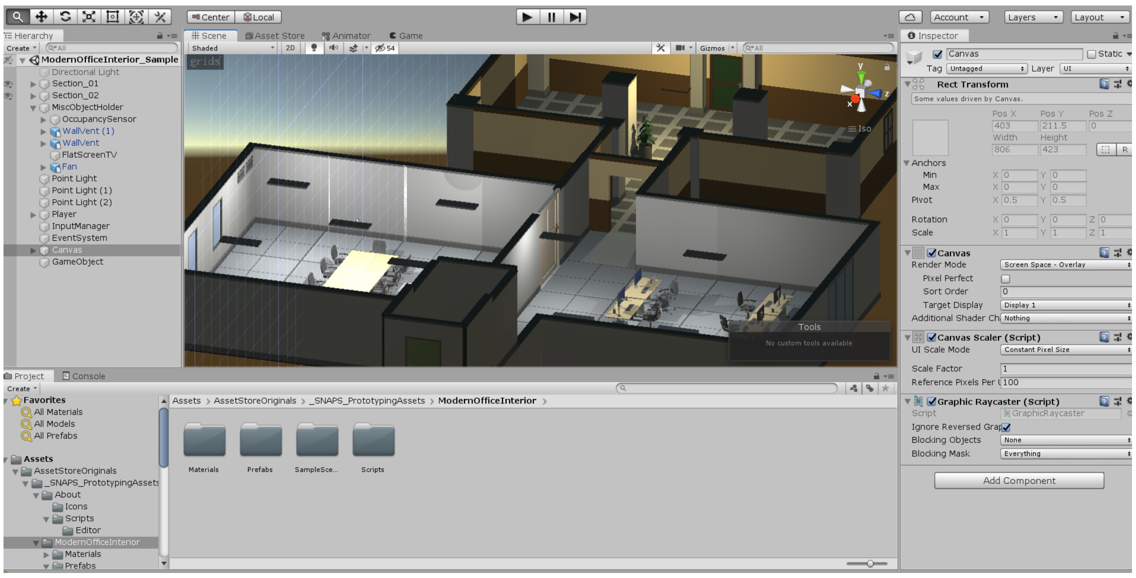
Figure 7.8: Attributes needed for communication in 'SendToUnity' block

7.3 Implementation of Digital Twin system in Unity Game Engine

The virtual models required for visualising the SMR system is developed in the Unity Game Engine 2019.1.0f2. This tool allows the user to develop 3D models either from scratch or directly from the asset store. Moreover, The asset store in unity software consists of a lot of 3D models which is freely available and is open source. The assets are imported as a package to the project and it is possible to import assets that contain animations and behaviours predefined. The following subsections describe the details of creating a SMR system from scratch namely, the development of 3D models in Unity and the Digital Twin layered architecture.

7.3.1 Development of 3D models

The 'ModernOfficeInterior' package is imported from the asset store and contains almost all the necessary components of SMR system. As a consequence, the scene that shows the three dimensional visualisation of the office interior after importing to the project is shown in figure 7.9. As seen in the figure 6.9., the package consists of necessary subsystems and components such as the occupancy sensors, 'WallVent' for symbolising the HVAC system, Flat screen TV, Walls, Floor, Fan, Ceiling lights, etc. However, certain objects and animations need to be added separately. For instance, the Meeting room needs a door that opens only when the user gives the correct PIN number in the GUI of IBM Rhapsody. In addition, a fire animation is required to depict the emergency situation inside the meeting room. For this purpose, the package 'Tim'sAssets_Door' and the prefab 'FlamesParticleEffect' are used. Both the aforementioned components require further detailing of its animations so as to make it look fitting with the SMR environment. Therefore, in case of the door, the state behaviour of the door is defined so as to animate it. The state behaviour of the door is depicted in the figure 7.10. The state transition A has no triggers or conditions, so the door will remain in the wait state without any action. However, the state transitions B and D require the condition that the flag variable is set to 'true'. finally, since, fire explosion model has predefined animation embedded, it can be used as such. However, the flame size and colour can be changed using the Inspector toolbar in unity Game Engine. Thus, in this way, the 3D models are developed, animated and visualised for the SMR system.



```
ModernOfficeInterior_Sample
  Directional Light
  Section_01
  Section_02
  MiscObjectHolder
    OccupancySensor
    WallVent (1)
    WallVent
    FlatScreenTV
    Fan
  Point Light
  Point Light (1)
  Point Light (2)
  Player
  InputManager
  EventSystem
  Canvas
  GameObject
```

```
Favorites
  All Materials
  All Models
  All Prefabs
Assets
  AssetStoreOriginals
    _SNAPS_PrototypingAssets
      About
        Icons
        Scripts
          Editor
      ModernOfficeInterior
        Materials
        Prefabs
```

Assets > AssetStoreOriginals > _SNAPS_PrototypingAssets > ModernOfficeInterior >

```
Canvas                         Static
Tag Untagged                   Layer UI
Rect Transform
Some values driven by Canvas.
            Pos X    Pos Y    Pos Z
            403      211.5    0
            Width    Height
            806      423
Anchors
  Min       X 0      Y 0
  Max       X 0      Y 0
Pivot       X 0.5    Y 0.5
Rotation    X 0      Y 0      Z 0
Scale       X 1      Y 1      Z 1
Canvas
Render Mode              Screen Space - Overlay
  Pixel Perfect
  Sort Order             0
  Target Display         Display 1
Additional Shader Ch     Nothing
Canvas Scaler (Script)
UI Scale Mode            Constant Pixel Size
Scale Factor             1
Reference Pixels Per U   100
Graphic Raycaster (Script)
Script                   GraphicRaycaster
Ignore Reversed Grap     ✓
Blocking Objects         None
Blocking Mask            Everything
```

Figure 7.9: Scene view of the 'ModernOfficeInterior' package from asset store

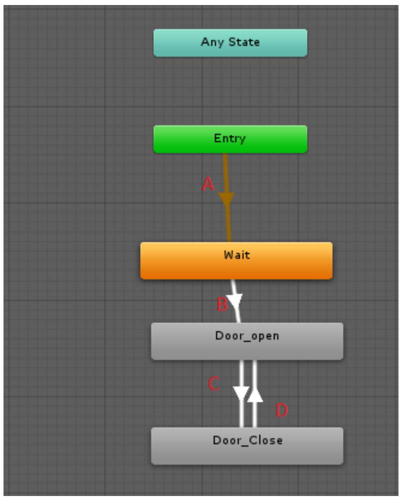


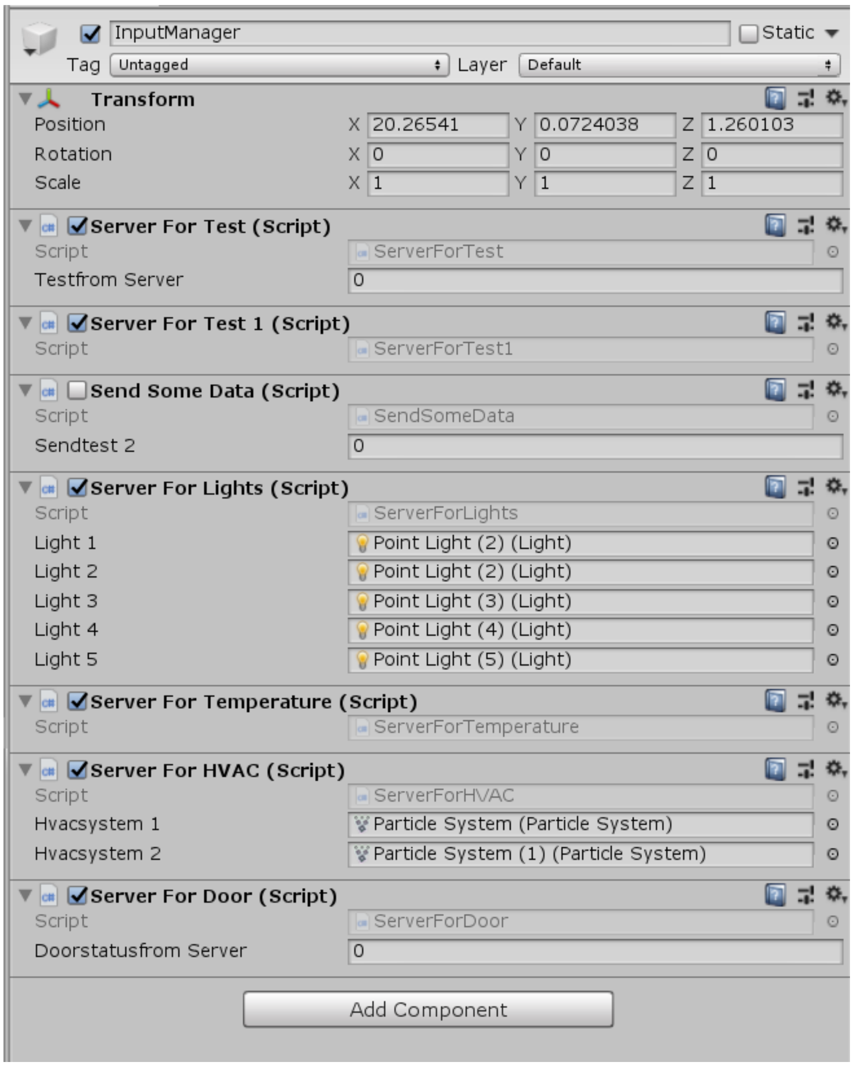
Figure 7.10: State Behaviour of the door used in SMR system

7.3.2 Implementation of Digital Twin layered architecture

As discussed in section 6.3, the digital twin of the SMR system is developed in a layered architecture. The first and the basic layer is the data abstraction layer followed by transmission layer, digital modelling layer, data/model integration layer and finally the application/service layer. As far as this project is considered, it is not in the scope to implement physical data collection and abstraction from hardware sensors and components. The sensor values and user inputs are collected from IBM Rhapsody model. However, the transmission layer, Data/Model integration layer, and the application/service layer are implemented in separate C# scripts and are discussed in detail in the below sections.

Transmission layer

The transmission layer helps in communicating the sensor values and user inputs from Rhapsody model to the Unity Game Engine. The client-server communication using windows socket programming concept is already discussed in the previous section 6.2. While IBM Rhapsody acts as the client with relevant sensor data and user information, the Unity game engine acts as the server that enables the ports and therefore establishes communication to IBM Rhapsody. The server side source code implementation in c# language is attached in the appendix B section for reference.



InputManager ☐ Static
Tag Untagged Layer Default

Transform

	X	Y	Z
Position	20.26541	0.0724038	1.260103
Rotation	0	0	0
Scale	1	1	1

Server For Test (Script)
Script ServerForTest
Testfrom Server 0

Server For Test 1 (Script)
Script ServerForTest1

Send Some Data (Script)
Script SendSomeData
Sendtest 2 0

Server For Lights (Script)
Script ServerForLights
Light 1 Point Light (2) (Light)
Light 2 Point Light (2) (Light)
Light 3 Point Light (3) (Light)
Light 4 Point Light (4) (Light)
Light 5 Point Light (5) (Light)

Server For Temperature (Script)
Script ServerForTemperature

Server For HVAC (Script)
Script ServerForHVAC
Hvacsystem 1 Particle System (Particle System)
Hvacsystem 2 Particle System (1) (Particle System)

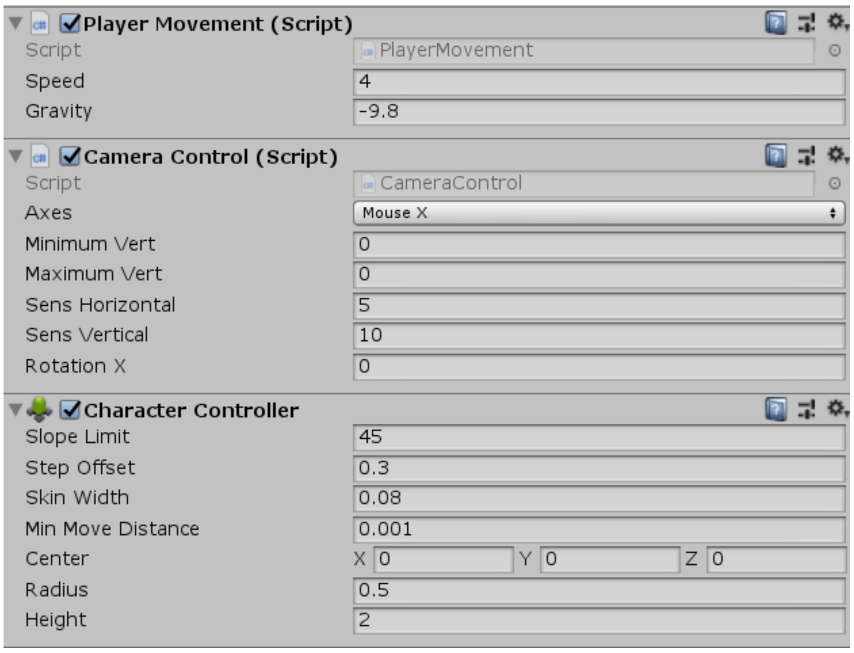
Server For Door (Script)
Script ServerForDoor
Doorstatusfrom Server 0

Add Component

Figure 7.11: Input Manager that binds the server scripts in transmission layer of Digital Twin

Digital Modelling layer

In this layer, the scene view of the SMR system has to be modelled in a game like fashion. For this purpose, the player movement inside the office interior and the respective camera control and mouse positioning are designed in this layer. The scripts for player movement control, camera control and mouse position control are bound to the game object 'Player' in the unity project. The character controller tutorial helps in realising a simple player movement [11]. In addition, after testing the player movement inside the office interior, the final character control settings are shown in the figure 7.12.



Player Movement (Script)	
Script	PlayerMovement
Speed	4
Gravity	-9.8
Camera Control (Script)	
Script	CameraControl
Axes	Mouse X
Minimum Vert	0
Maximum Vert	0
Sens Horizontal	5
Sens Vertical	10
Rotation X	0
Character Controller	
Slope Limit	45
Step Offset	0.3
Skin Width	0.08
Min Move Distance	0.001
Center	X 0 Y 0 Z 0
Radius	0.5
Height	2

Figure 7.12: Camera, player and character control settings in Unity Game Engine

Data/Model integration layer

As far as Data/model integration layer is considered, the individual subsystem behaviour are modelled by integrating with the game objects inside the office interior package. For instance, the collision and trigger script helps in realising the function of the occupancy sensor by turning on the lights whenever the player has entered the room. Similarly, the light adjustment script helps in modifying the intensity of the lights used. It is important to note that it even is possible to change the colour of the lights used. Furthermore, the 'Flamechange' and 'door_opening' scripts helps in realising the functionality of the fire explosion scenario and animation of the door opening respectively. The integration layer scripts are added in the appendix B section for reference.

Application layer

In this layer, the overall SMR system control is established. The multiple modes of SMR system are modelled in a script named 'ChangeText'. The system values such as HVAC status, Temperature, lighting status, Meeting room message are displayed in a text format in the screen along with the audio output. The script takes the mode input value from the IBM Rhapsody model and changes the system behaviour accordingly.

7.4 Implementation on a Raspberry pi controller

The application created by the IBM Rhapsody models and Unity Game Engine can be ported to a remote device and its functionality can be tested for a real system. In fact, the answer to the research question 3 is based on coupling the SMR system with a raspberry pi controller that is connected remotely. Typically a microcontroller has limited resources and no operating system. Therefore, building a cross compiler and setting its configuration for a linux OS is required so as to implement a project on the remote device. In such a case, the generated executable file is then ported to the remote device for running the application. However, the raspberry pi controller has its own operating system - the Raspbian GNU/Linux 10. Therefore, through capitalizing the powerful CPU of raspberry pi, it is possible to eliminate the time consuming process of setting up of a cross compiler platform in IBM Rhapsody project. In this way, the model can build project files directly inside the remote target and execute the application remotely. In this section, the workflow followed in automating the creation of the application executable for raspberry pi OS environment is described [12].

7.4.1 Buidling the Rhapsody 0XF framework

Firstly, the linux configuration hex files are needed to be ported to the remote target device. In order to port the project files to a remote device, the remote secure scripting concept is utilised. Since the IBM Rhapsody user interface supports the development of remote device deployment, the concept of automated build, compile and run process is preferred. However, for this purpose, the linux profile requires extensive usage of putty as SSH client. Therefore, it requires putty, plink.exe, and pscp.exe in the tool directory. As a prerequisite, successful connection to the remote target and permission to the remote target's SSH key is ensured. This is a one time operation per local host.The plink command with the appropriate user-id and password will SSH connect to the remote device and create a project work-space directory inside the raspberry pi memory. Further, the pscp command will help transfer the linux tar files from host machine to target device. Once the linux tar files are downloaded on the target system, the plink command initiates the extraction process inside the target system to convert it into necessary hex files. Finally, the file access permissions are modified and the command to build framework is processed. The overall procedure is depicted in the figure 7.13.

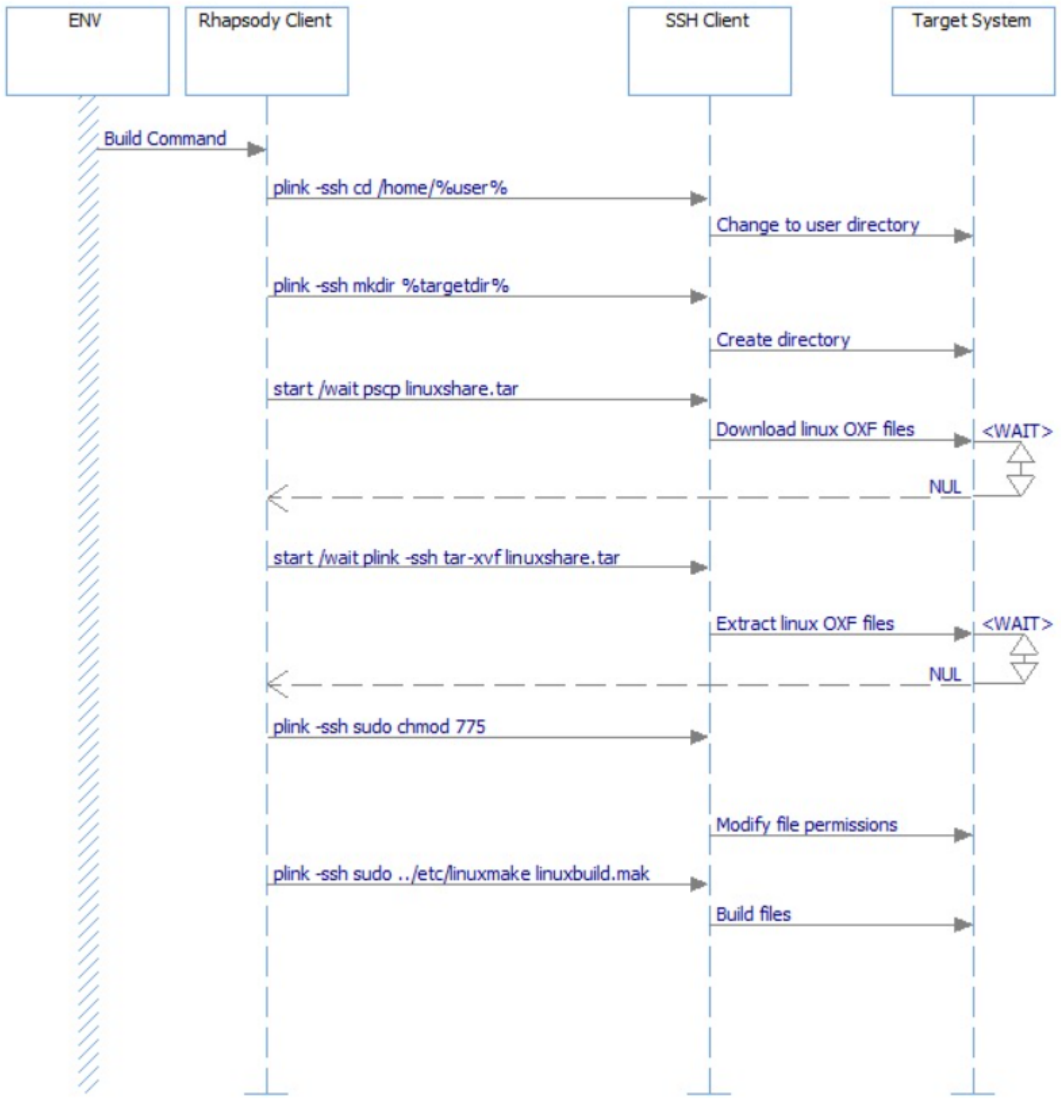

ENV
Rhapsody Client
SSH Client
Target System
Build Command
plink -ssh cd /home/%user%
Change to user directory
plink -ssh mkdir %targetdir%
Create directory
start /wait pscp linuxshare.tar
Download linux OXF files
<WAIT>
NUL
start /wait plink -ssh tar-xvf linuxshare.tar
Extract linux OXF files
<WAIT>
NUL
plink -ssh sudo chmod 775
Modify file permissions
plink -ssh sudo ../etc/linuxmake linuxbuild.mak
Build files

Figure 7.13: Workflow for automating the framework build process in Rhapsody

7.4.2 Building the project on a remote device

The make command in the Rhapsody interface initiates the project build process. However, to initiate the build process in a remote device that uses linux OS, the following tool-chain is preferred. Firstly, the plink makes an ssh connection to the remote device and creates a target directory inside the device. subsequently, the pscp command executes the consecutive file transfer of c++ source files, header files and make files to the target system. finally, the plink initiates the make build process which will compile the project source files and outputs the object files in the target directory followed by compiling the makefile which outputs the project executable in the same folder. The overall process is depicted in a sequence diagram in the figure 7.14.

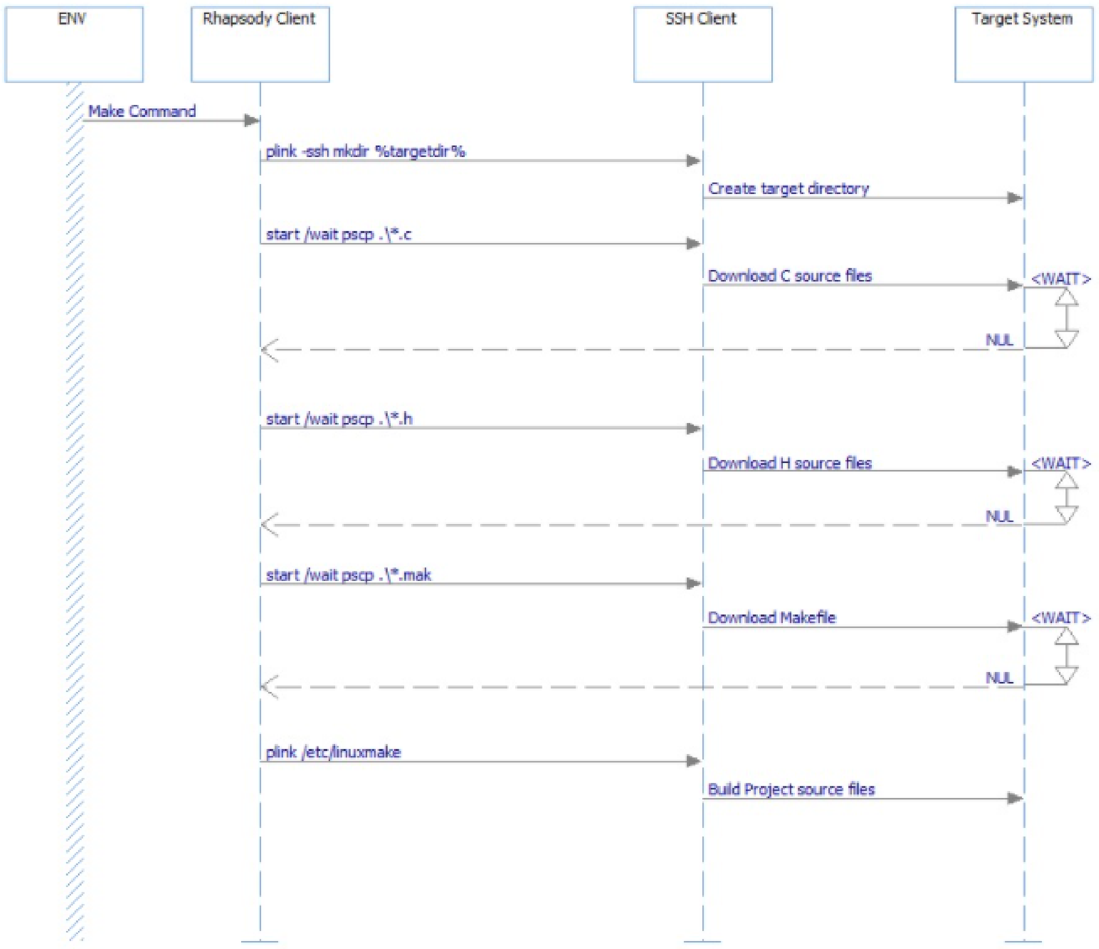


Figure 7.14: Workflow for automating the project build and compile process in Rhapsody

7.4.3 Executing the Project

Typically, when the run command is issued in a project that runs in the host machine, the executable file present in the project workspace directory is directly invoked and the output is presented in the Rhapsody client interface. However, when working on a remote device, the project executable is present on the remote device's memory. Therefore, the plink makes an SSH connection to remote device and then issues the run command over the executable file. The overall workflow is depicted as a sequence diagram in figure 7.15.

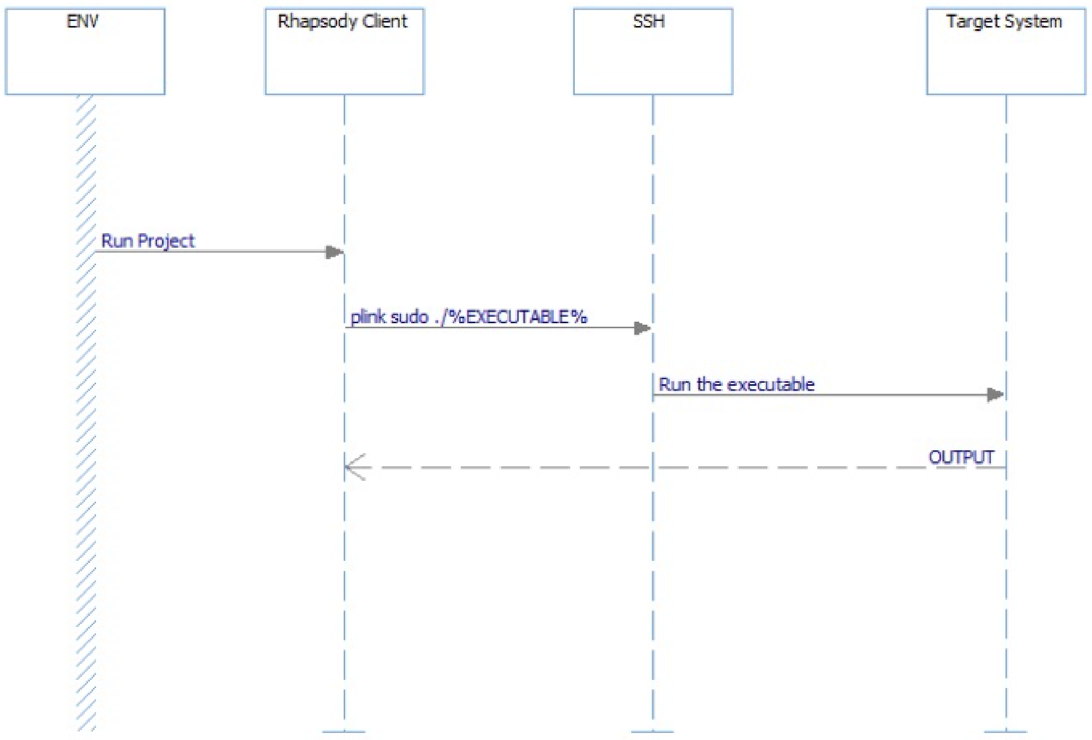


Figure 7.15: Workflow for automating the project execution process in Rhapsody

Chapter 8

Results

In this chapter, the results obtained upon successful completion of the project is described in detail. The following series of pictures demonstrates the working of SMR system in different scenarios when Rhapsody application is coupled with Unity Game Engine. The results observable in each state verifies the working of SMR system using the Digital Twin approach. The following steps 1 to 8 were performed to demonstrate the results displayed.

8.1 Demonstration of SMR system

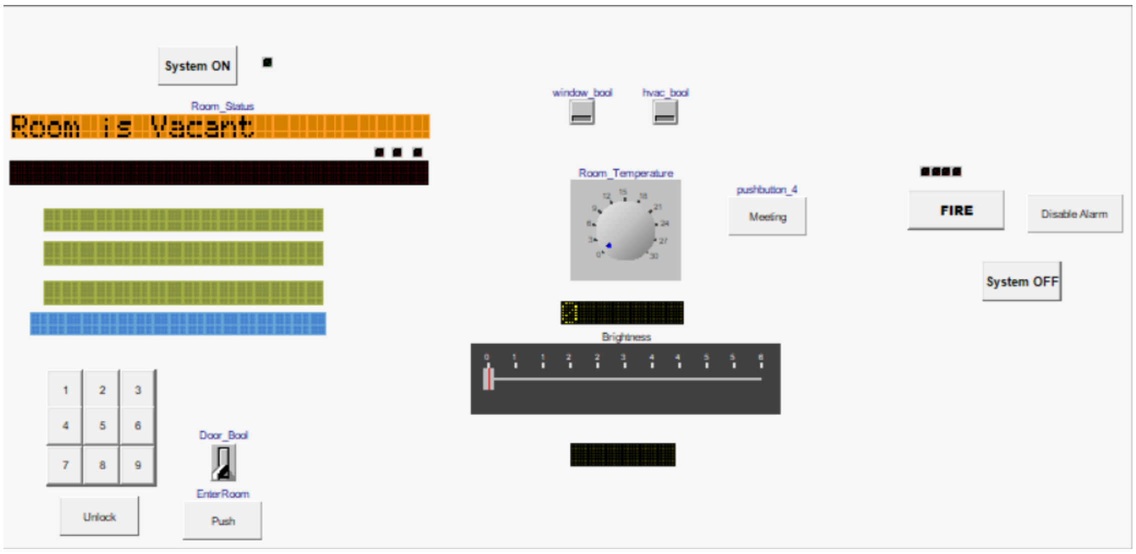


Figure 8.1: Step 1a: Turn on Rhapsody GUI

1. Run the project execution in Rhapsody. The animation toolbar is automatically turned ON. As, a first step, make the executable go into idle state and then press Go in the animation toolbar.The room is vacant initially as observed in Figure 8.1. The system On button in the control panel needs to be toggled for enabling the SMR system.

2. Once, the SMR system is turned on in the Rhapsody GUI, the Unity Project can be executed. As per the implementation, the The display at Unity will welcome the user into the SMR room as observable in figure 8.2. The user can now move around the office interior to verify the player movements and camera control.

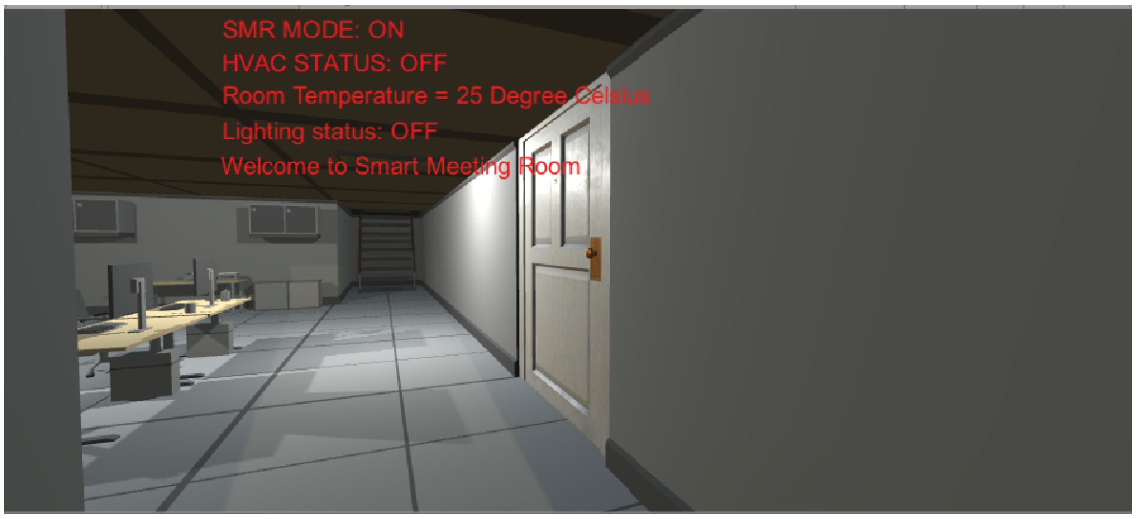

SMR MODE: ON
HVAC STATUS: OFF
Room Temperature = 25 Degree Celsius
Lighting status: OFF
Welcome to Smart Meeting Room

Figure 8.2: Step 1b: Run Game sequence in Unity

3. Secondly, the user can now enter the PIN number to gain access inside the room. The PIN number ”159” provides access into the room and the display message in Rhapsody GUI suggests that the door is now unlocked. The control panel has has display messages designed which helps in step-by-step model verification as shown in figure 8.3.

4. After gaining access inside the room by providing the correct credentials in Rhapsody GUI, the Door for SMR in Unity will automatically unlock and open itself. The user can now move inside the room and the SMR system is found to be in sleep state as shown in figure 8.4. The lighting level, HVAC status and Room Temperature were also found to be displaying proper values as per design.

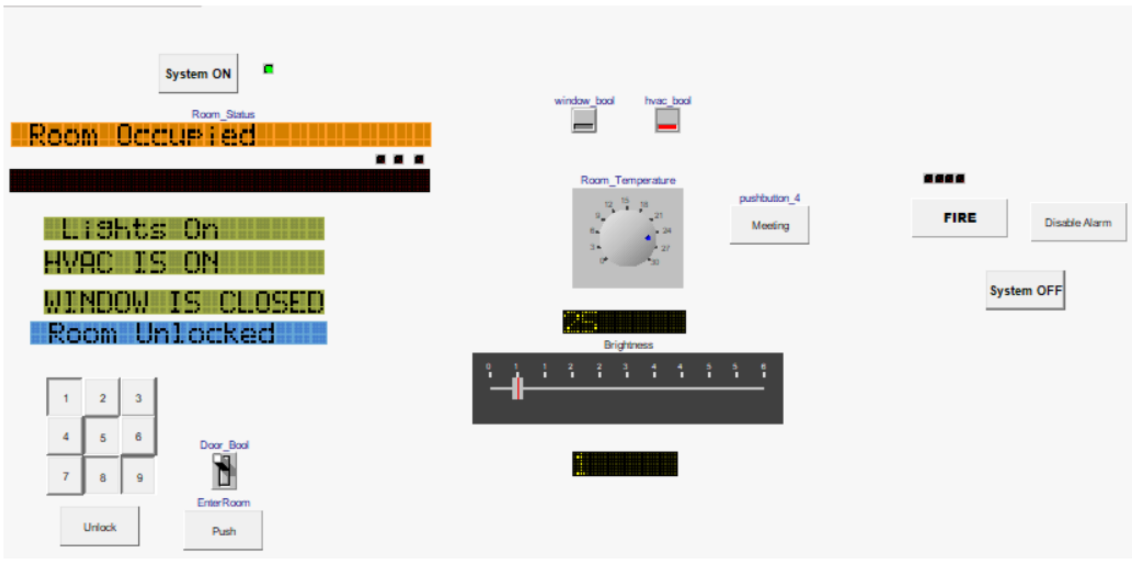


Figure 8.3: Step 2a: Give access to the SMR in Rhapsody GUI

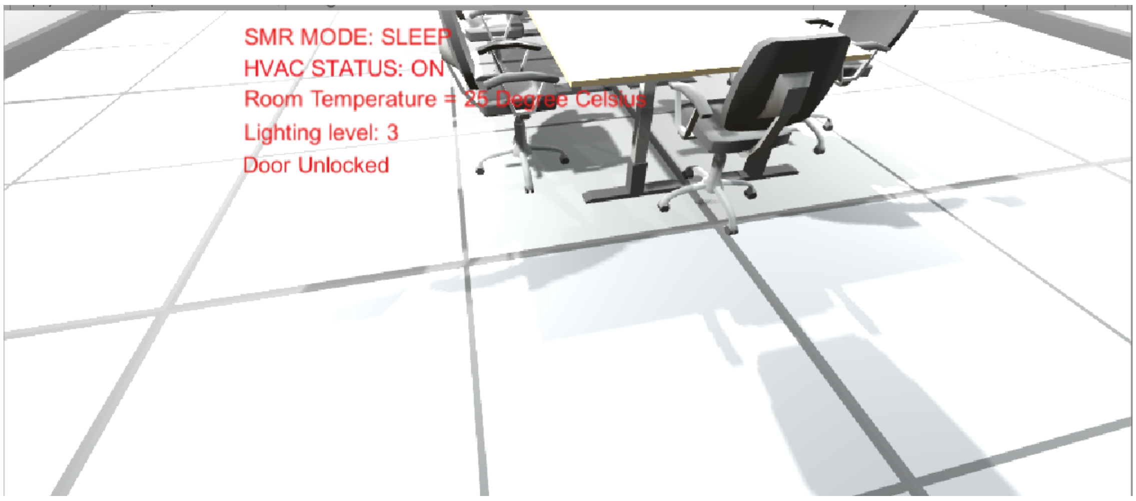

SMR MODE: SLEEP
HVAC STATUS: ON
Room Temperature = 25 Degree Celsius
Lighting level: 3
Door Unlocked

Figure 8.4: Step 2b: Enter into the Meeting room in Unity

5. Now the system can be controlled to modify the temperature and Lighting level by the user. Additionally, the user can now switch to the meeting mode. As per the design, the meeting mode has predefined set of values which can be verified in the Rhapsody GUI. Figure 8.5. shows the Rhapsody GUI that displays the SMR in meeting mode with its corresponding subsystem states.

6. The meeting mode of SMR system in unity aligns perfectly synchronous to the Rhapsody application. The unity model in figure 8.6 shows that the lighting level and temperature of the SMR system are changed as per the data sent by the rhapsody application. Additionally, the speakers are also enabled and its output is audible in this mode. This verifies the working of SMR system in meeting mode.

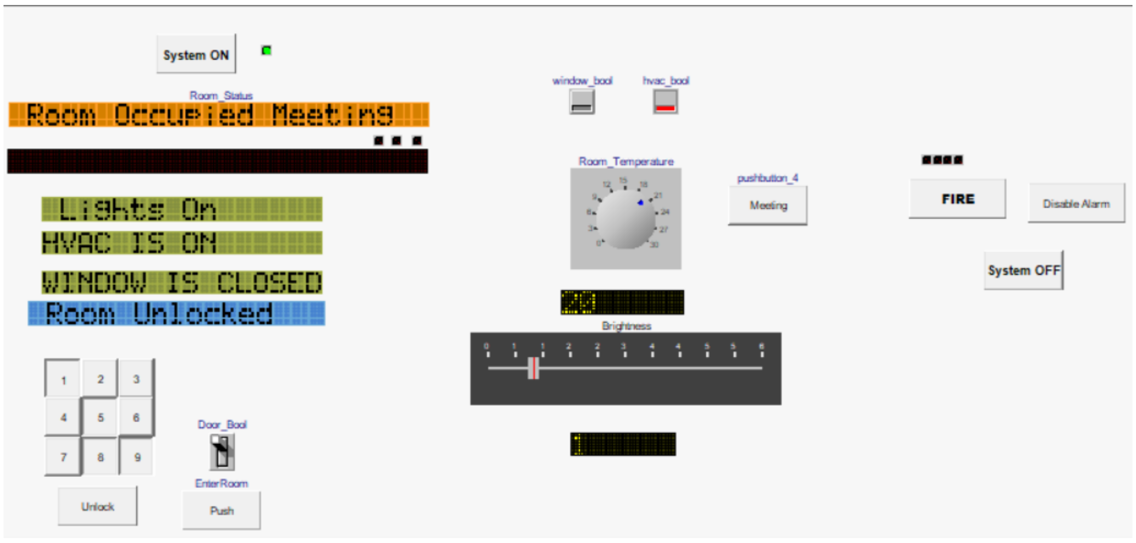


Figure 8.5: Step 3a: switch the SMR mode to Meeting

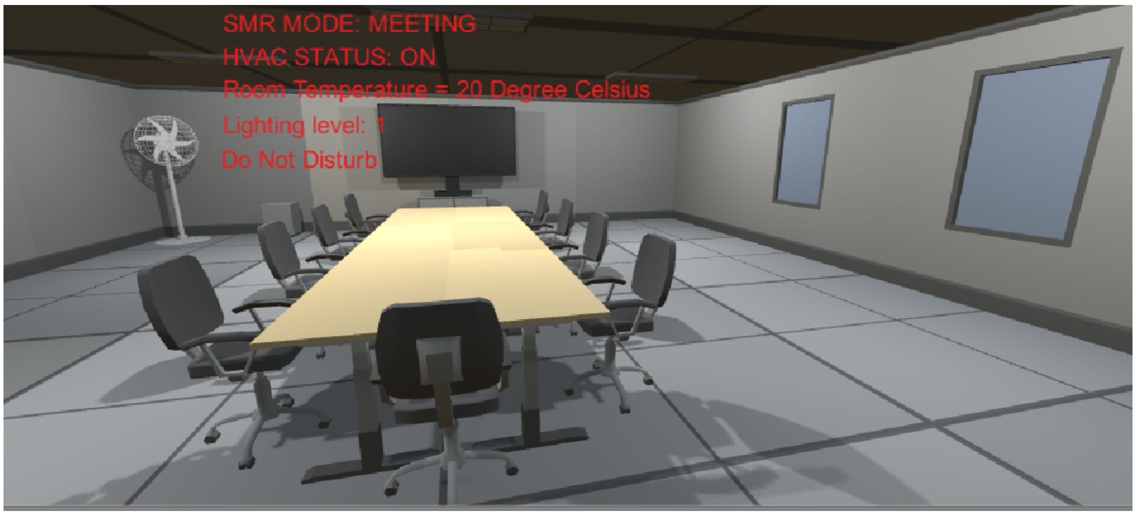
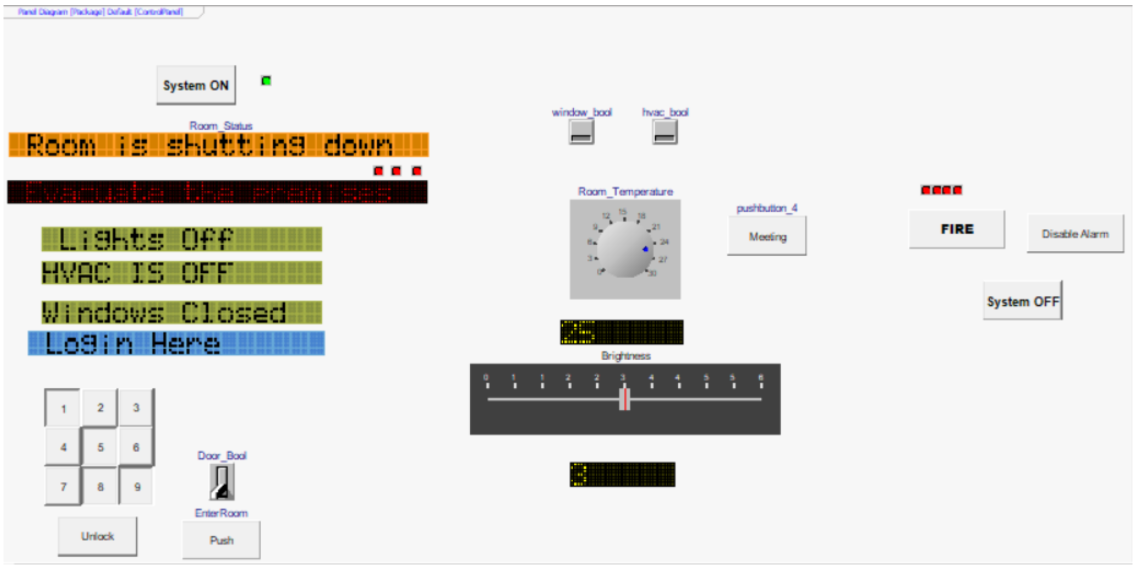


Figure 8.6: Step 3b: verify the changes in the unity

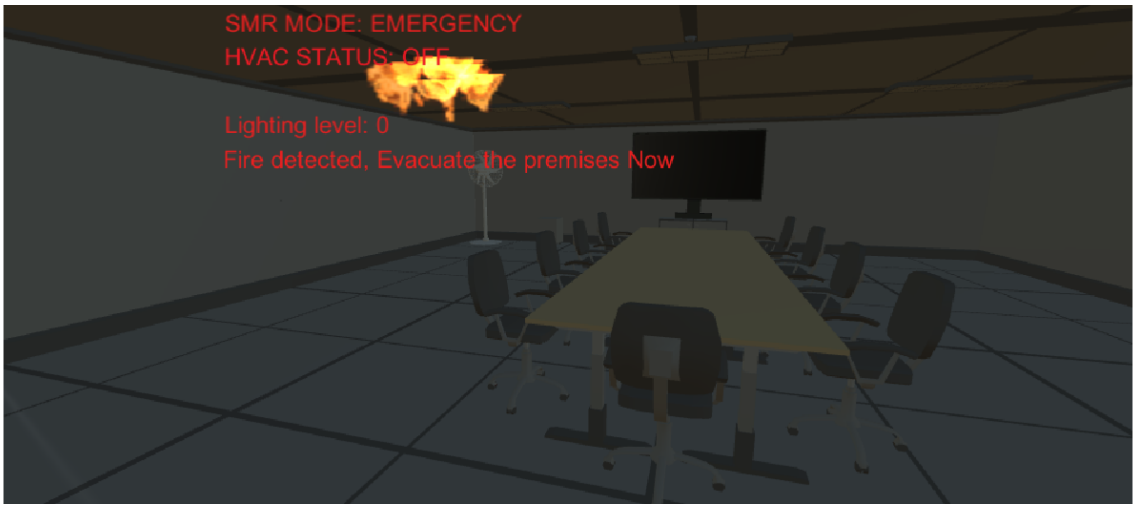
7. The scenario of fire emergency can be simulated by using the Fire button modelled in the control panel. Pressing the fire button, simulates the SMR beahviour and displays the results in the display as shown in figure 8.7. We can observe that the HVAC system, Lighting system are all turned off with an evacuation message.

8. The fire simulation yields the following result in Unity game engine as shown in figure 8.8. We can observe that the Rhapsody application enables the SMR system in unity by performing the necessary controls and generating the observed system results. It is possible to disable the Fire alarm and the SMR system will reiterate back to the sleep state.



Room is shutting down
Evacuate the premises
Lights Off
HVAC IS OFF
Windows Closed
Login Here

Figure 8.7: Step 4a: switch the SMR mode to Fire emergency in Rhapsody GUI



SMR MODE: EMERGENCY
HVAC STATUS: OFF

Lighting level: 0
Fire detected, Evacuate the premises Now

Figure 8.8: Step 4b: Verify the environment change in unity

Chapter 9

Conclusion and future scope

In conclusion, the developed SMR application satisfies the fundamental objectives of this graduation thesis and answers the research questions effectively. This is accomplished by answering the research question 1 in section 5.1, research question 2 in section 6.3 and 7.2, and finally the research question 3 in section 7.4. During the course of this project, there were a number of learning experiences, challenges, personal improvement and shortcomings. However, the shortcomings are minimal and less significant in comparison to the rest of the aspects of this project.

Firstly, on evaluation of the SMR system implementation, it is observed that it satisfies the basic set of requirements listed in section 5.1. However, certain subsystems were not implemented as per the design considerations and use case specifications mentioned in chapter 6.2. For instance, the CCTV camera operations in security system, the touch panel activation in audio visual system, the conferencing and video calling features in the TV, the motion detection sensors and window panel activation are removed from SMR system implementation. This is mainly because of the complexity in the virtual realisation of the actors and components in the Unity Game Engine. Moreover, it does not serve any additional purpose to contribute to the research questions. Therefore, these use cases and and its related actors were removed out of the SMR system implementation.

Secondly, in regards to the third research question, testing the use cases of SMR system to communicate with Digital Twin was unable to be performed. This is mainly because the animation enabled in IBM Rhapsody does not allow to perform the operations in linux ports on Raspberry Pi controller. Although, The SMR system can be executed successfully on a remote device, the functionalities of SMR system were not fully tested on Raspberry Pi controller. However, since there are no changes in the model implementation, it can be argued that the test cases will function similar to the windows implementation. Moreover, since testing and evaluation of HIL implementation is not part of the research question, it can be left for future scope of development. A suitable workaround for this situation can be by exporting the control panel diagram to a web interface and accessing the webpage from the Raspberry Pi controller to give user inputs to the Digital Twin. In addition, new insights and the future scope of improvement related to the design of a digital twin based on a model-driven system engineering approach are found and discussed in the below sections.

9.1 Application interoperability

The SMR application developed in the project has the functionality and advantage of being interoperable between different processes. The SysML modelling in IBM Rhapsody tool helps in auto generating the C++ source files required for the successful execution of the project. The advantage of IBM Rhapsody as an interoperable tool between a different application has been exploited to a great extent in this project. Indeed, the virtual model in 3D space developed in Unity Game

Engine is successfully coupled with the SMR application. In addition, Since C++ offers portability of the code in various OS environments, it is easier to execute the project on remote devices with the help of suitable profiling of the models for the required OS environment in IBM Rhapsody tool. This has been demonstrated with the help of linux profiling of the SMR application and thereby executing the project in a remotely connected raspberry pi controller (Raspbian OS - a linux based OS).

Moreover, the application can be extended to have additional controller and stand-alone components that runs parallelly either on same or different host machines. For instance, an additional control layer can be adopted in MATLAB/simulink software to allow multi agent simulations. Although, this has not been implemented in this project, it is a useful concept that can be exploited in the future.

9.2 Flexiblity in development approach

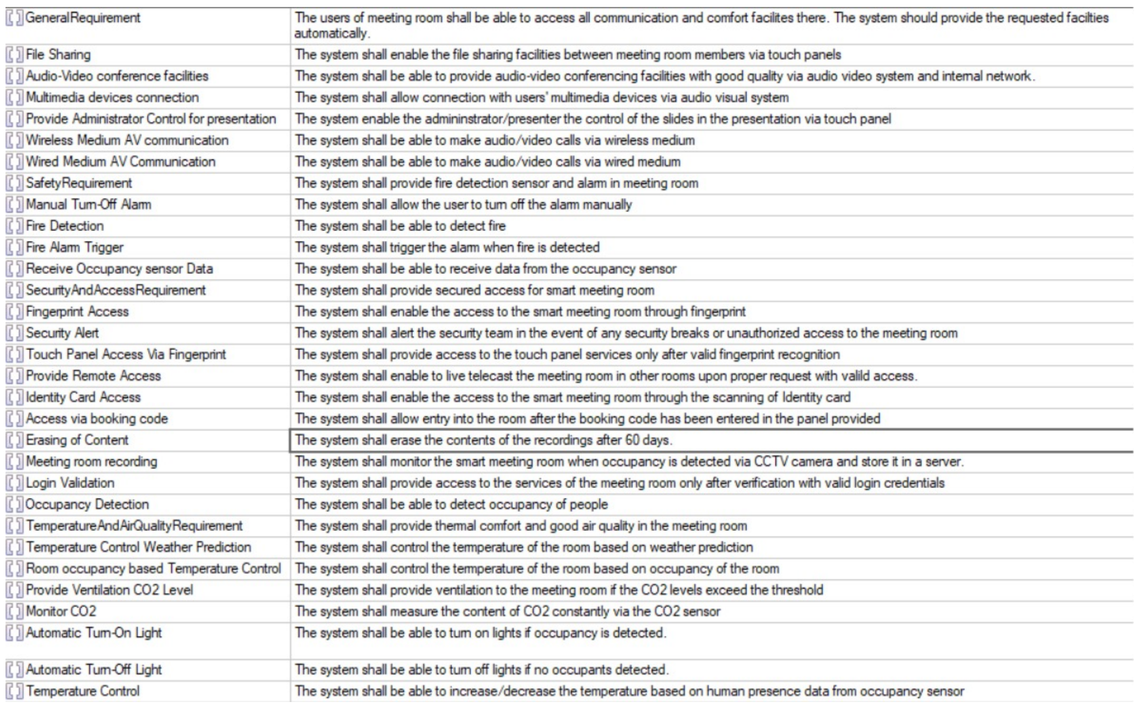
The model-driven system engineering approach has the advantage of being flexible to development at different stages. It was found during the project that the flexibility of SysML modelling offers new feature additions as well as modification of existing attributes of features and services in an easy and less redundant method. For instance, adding a new feature such as an animated door in the 3D space requires software signals to actuate it. These software signals has to be generated from the SMR application through the SysML models. Therefore, it requires additional ports for socket communication and subsequently appropriate designing in the model integration layer as well as application/service layer. However, these changes do not in any way interfere with the working of pre-existing components or services as they are all segregated by individual layers. Therefore, there is minimum rework in flexible model-driven system engineering approach which improves productivity and thus increases the cost effectiveness of the life cycle development.

Further, The SMR application can be interfaced with IoT application through the interaction with cloud environments and servers. Since there are numerous advantages to enable cloud services to SMR application, this extension is a plausible future scope of development. For instance, the HVAC control can be fed directly from cloud based servers based on weather forecast. Additionally, new prediction analysis modles can be incorporated with the system to improve performance of the SMR application in terms of energy consumption and comfort provided.

Appendix A

SMR System Requirements

All the requirements are captured and given a Requirement ID as per SysML modelling standards. The following figure A.1 consists of the requirements captured under SMR model. The Requirement diagrams for each package namely General, Communication, Temperature and Air Control, safety, Security and Access package are shown in subsequent diagrams.



GeneralRequirement	The users of meeting room shall be able to access all communication and comfort facilites there. The system should provide the requested facilties automatically.
File Sharing	The system shall enable the file sharing facilities between meeting room members via touch panels
Audio-Video conference facilities	The system shall be able to provide audio-video conferencing facilities with good quality via audio video system and internal network.
Multimedia devices connection	The system shall allow connection with users' multimedia devices via audio visual system
Provide Administrator Control for presentation	The system enable the admininstrator/presenter the control of the slides in the presentation via touch panel
Wireless Medium AV communication	The system shall be able to make audio/video calls via wireless medium
Wired Medium AV Communication	The system shall be able to make audio/video calls via wired medium
SafetyRequirement	The system shall provide fire detection sensor and alarm in meeting room
Manual Turn-Off Alarm	The system shall allow the user to turn off the alarm manually
Fire Detection	The system shall be able to detect fire
Fire Alarm Trigger	The system shall trigger the alarm when fire is detected
Receive Occupancy sensor Data	The system shall be able to receive data from the occupancy sensor
SecurityAndAccessRequirement	The system shall provide secured access for smart meeting room
Fingerprint Access	The system shall enable the access to the smart meeting room through fingerprint
Security Alert	The system shall alert the security team in the event of any security breaks or unauthorized access to the meeting room
Touch Panel Access Via Fingerprint	The system shall provide access to the touch panel services only after valid fingerprint recognition
Provide Remote Access	The system shall enable to live telecast the meeting room in other rooms upon proper request with valild access.
Identity Card Access	The system shall enable the access to the smart meeting room through the scanning of Identity card
Access via booking code	The system shall allow entry into the room after the booking code has been entered in the panel provided
Erasing of Content	The system shall erase the contents of the recordings after 60 days.
Meeting room recording	The system shall monitor the smart meeting room when occupancy is detected via CCTV camera and store it in a server.
Login Validation	The system shall provide access to the services of the meeting room only after verification with valid login credentials
Occupancy Detection	The system shall be able to detect occupancy of people
TemperatureAndAirQualityRequirement	The system shall provide thermal comfort and good air quality in the meeting room
Temperature Control Weather Prediction	The system shall control the temperature of the room based on weather prediction
Room occupancy based Temperature Control	The system shall control the temperature of the room based on occupancy of the room
Provide Ventilation CO2 Level	The system shall provide ventilation to the meeting room if the CO2 levels exceed the threshold
Monitor CO2	The system shall measure the content of CO2 constantly via the CO2 sensor
Automatic Turn-On Light	The system shall be able to turn on lights if occupancy is detected.
Automatic Turn-Off Light	The system shall be able to turn off lights if no occupants detected.
Temperature Control	The system shall be able to increase/decrease the temperature based on human presence data from occupancy sensor

Figure A.1: List of all requirements for SMR model

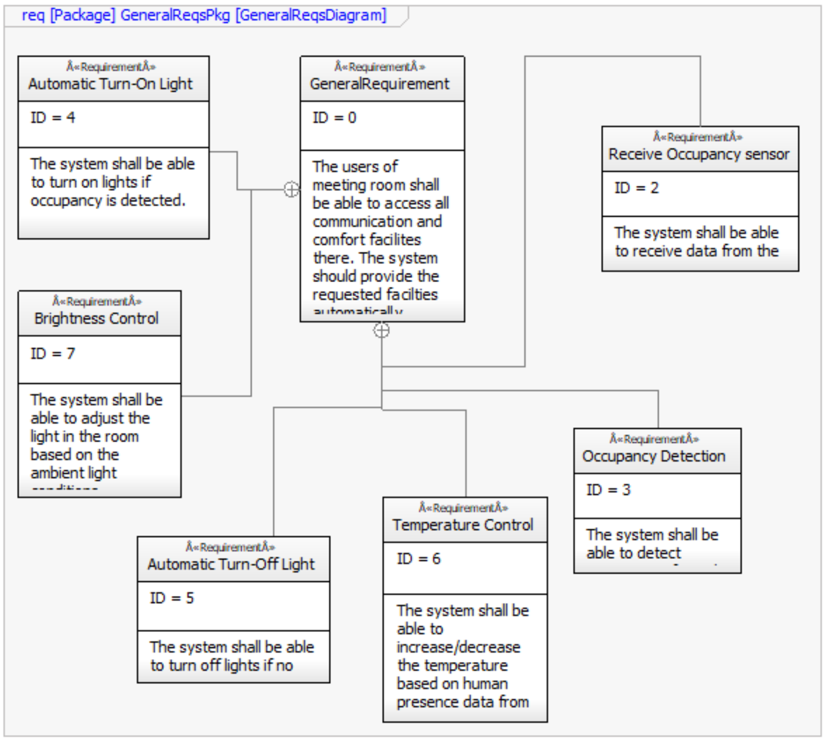

req [Package] GeneralReqsPkg [GeneralReqsDiagram]
Â«RequirementÂ» Automatic Turn-On Light
ID = 4
The system shall be able to turn on lights if occupancy is detected.
Â«RequirementÂ» GeneralRequirement
ID = 0
The users of meeting room shall be able to access all communication and comfort facilites there. The system should provide the requested facilties
Â«RequirementÂ» Receive Occupancy sensor
ID = 2
The system shall be able to receive data from the
Â«RequirementÂ» Brightness Control
ID = 7
The system shall be able to adjust the light in the room based on the ambient light
Â«RequirementÂ» Occupancy Detection
ID = 3
The system shall be able to detect
Â«RequirementÂ» Temperature Control
ID = 6
The system shall be able to increase/decrease the temperature based on human presence data from
Â«RequirementÂ» Automatic Turn-Off Light
ID = 5
The system shall be able to turn off lights if no

Figure A.2: Requirement Diagram for General Requirements Package

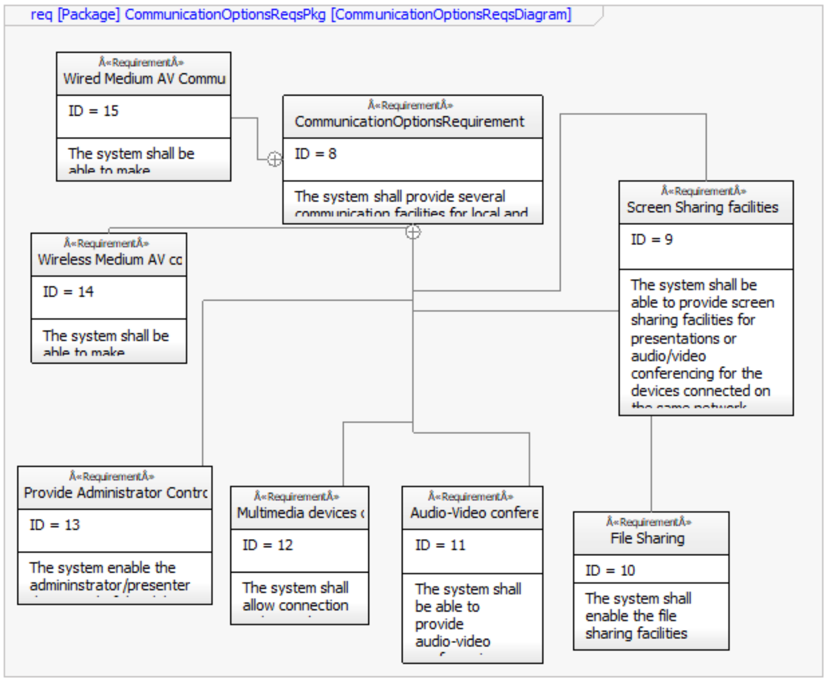

req [Package] CommunicationOptionsReqsPkg [CommunicationOptionsReqsDiagram]
Â«RequirementÂ» Wired Medium AV Commu
ID = 15
The system shall be able to make
Â«RequirementÂ» CommunicationOptionsRequirement
ID = 8
The system shall provide several communication facilities for local and
Â«RequirementÂ» Screen Sharing facilities
ID = 9
The system shall be able to provide screen sharing facilities for presentations or audio/video conferencing for the devices connected on the same network
Â«RequirementÂ» Wireless Medium AV cc
ID = 14
The system shall be able to make
Â«RequirementÂ» Provide Administrator Contrc
ID = 13
The system enable the admininstrator/presenter
Â«RequirementÂ» Multimedia devices c
ID = 12
The system shall allow connection
Â«RequirementÂ» Audio-Video confere
ID = 11
The system shall be able to provide audio-video
Â«RequirementÂ» File Sharing
ID = 10
The system shall enable the file sharing facilities

Figure A.3: Requirement Diagram for Communication system Package

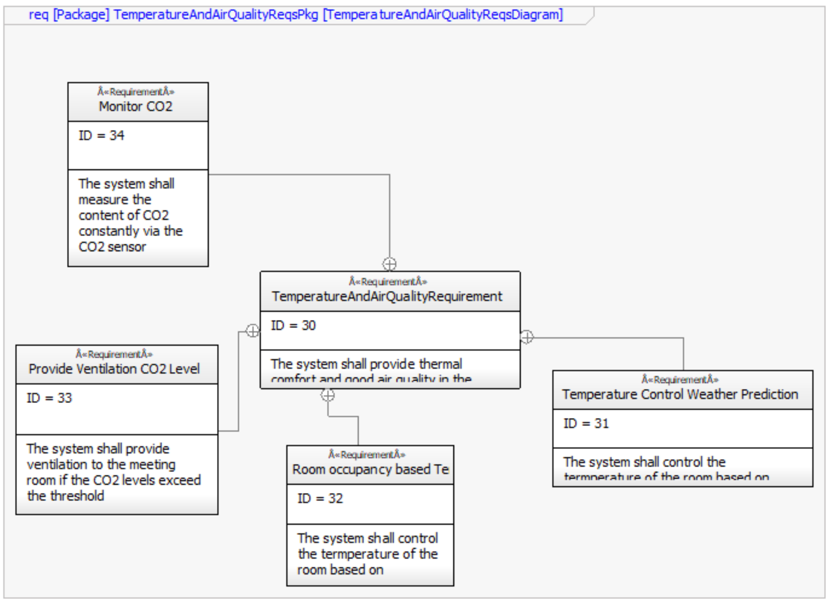

req [Package] TemperatureAndAirQualityReqsPkg [TemperatureAndAirQualityReqsDiagram]
Â«RequirementÂ» Monitor CO2
ID = 34
The system shall measure the content of CO2 constantly via the CO2 sensor
Â«RequirementÂ» TemperatureAndAirQualityRequirement
ID = 30
The system shall provide thermal comfort and good air quality in the
Â«RequirementÂ» Provide Ventilation CO2 Level
ID = 33
The system shall provide ventilation to the meeting room if the CO2 levels exceed the threshold
Â«RequirementÂ» Temperature Control Weather Prediction
ID = 31
The system shall control the termperature of the room based on
Â«RequirementÂ» Room occupancy based Te
ID = 32
The system shall control the termperature of the room based on

Figure A.4: Requirement Diagram for Temperature and Air control system Package

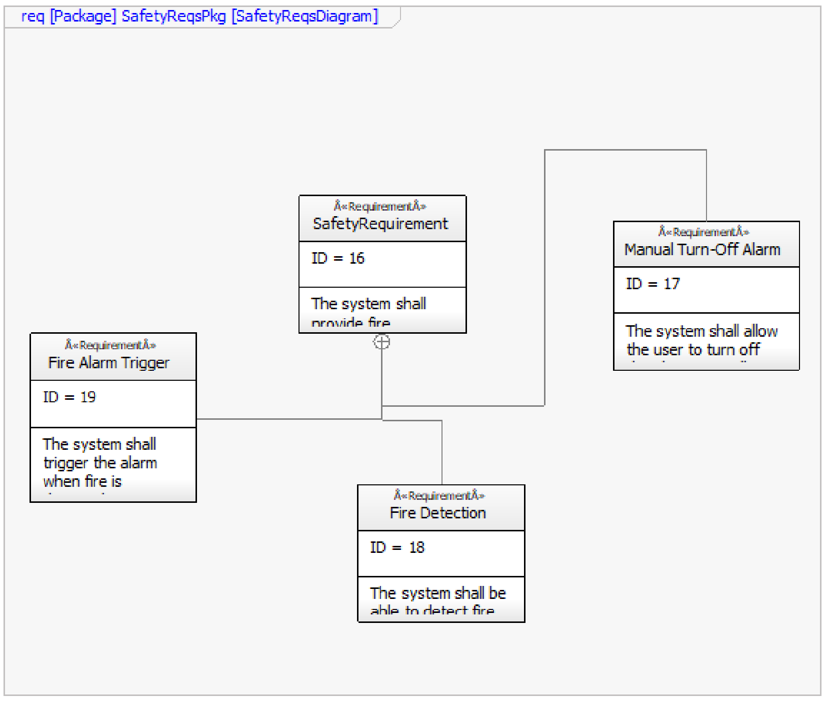

req [Package] SafetyReqsPkg [SafetyReqsDiagram]
Â«RequirementÂ» SafetyRequirement
ID = 16
The system shall provide fire
Â«RequirementÂ» Manual Turn-Off Alarm
ID = 17
The system shall allow the user to turn off
Â«RequirementÂ» Fire Alarm Trigger
ID = 19
The system shall trigger the alarm when fire is
Â«RequirementÂ» Fire Detection
ID = 18
The system shall be able to detect fire

Figure A.5: Requirement Diagram for Safety system Package

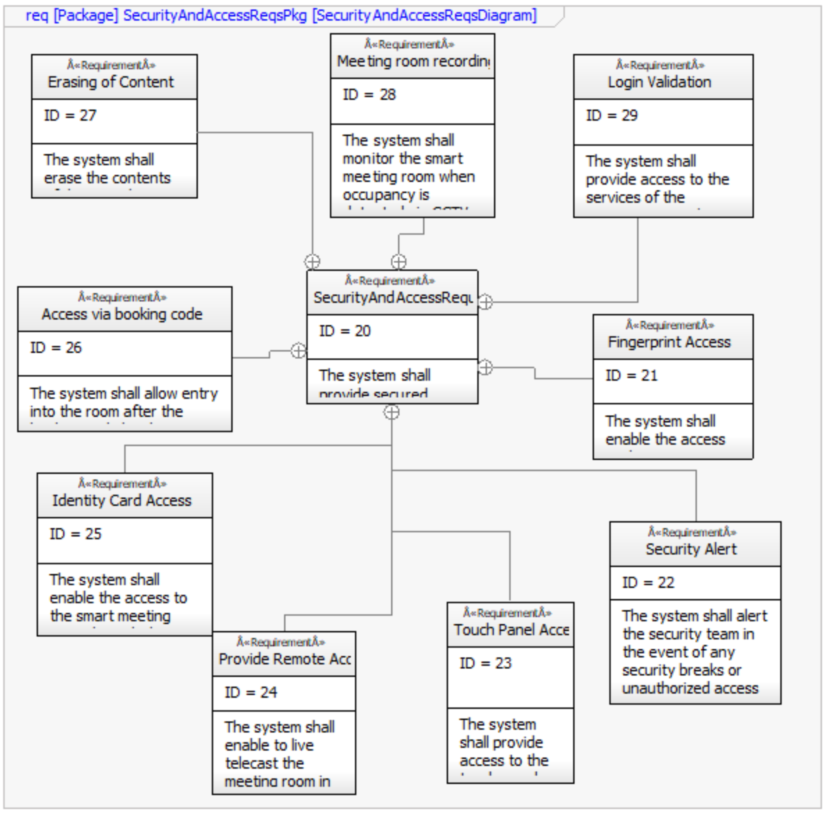

req [Package] SecurityAndAccessReqsPkg [SecurityAndAccessReqsDiagram]
Â«RequirementÂ» Erasing of Content
ID = 27
The system shall erase the contents
Â«RequirementÂ» Meeting room recordin
ID = 28
The system shall monitor the smart meeting room when occupancy is
Â«RequirementÂ» Login Validation
ID = 29
The system shall provide access to the services of the
Â«RequirementÂ» Access via booking code
ID = 26
The system shall allow entry into the room after the
Â«RequirementÂ» SecurityAndAccessRequ
ID = 20
The system shall provide secured
Â«RequirementÂ» Fingerprint Access
ID = 21
The system shall enable the access
Â«RequirementÂ» Identity Card Access
ID = 25
The system shall enable the access to the smart meeting
Â«RequirementÂ» Security Alert
ID = 22
The system shall alert the security team in the event of any security breaks or unauthorized access
Â«RequirementÂ» Touch Panel Acce
ID = 23
The system shall provide access to the
Â«RequirementÂ» Provide Remote Acc
ID = 24
The system shall enable to live telecast the meeting room in

Figure A.6: Requirement Diagram for Security and Access control system Package

Appendix B

Digital Twin Layered approach

B.1 Transmission layer

B.1.1 TCP client

```
#ifndef WIN32_LEAN_AND_MEAN
#define WIN32_LEAN_AND_MEAN
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#endif

#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

char out_string_for_hvac [1024];

//HVAC
memset(&address_hvac, 0, sizeof(address_hvac));
address_hvac.sin_family = AF_INET;
address_hvac.sin_port = htons(portHVAC);
address_hvac.sin_addr.s_addr=inet_addr(serverName);

//INITIALIZE THE SOCKET
listenSocketForHVAC = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);

//CONNECT  THE SOCKET
iResultHVAC = connect(listenSocketForHVAC,(struct sockaddr *)&address_hvac, sizeof(
    address_hvac));


//ASSIGN VALUES TO SEND
sprintf(out_string_for_hvac, "%f", hvac);

//SEND VALUE
iSendResultHVAC = sendto(listenSocketForHVAC, (char *)&out_string_for_hvac ,sizeof(
    uint32_t), 0
,(sockaddr *) & address_hvac, sizeof (address_hvac) );
//SHOW SENT VALUES IN CONSOLE
printf("HVAC Status value sent: %s \n",out_string_for_hvac);

//RECIEVED VALUE OF HVAC
iResultHVAC = recv(listenSocketForHVAC, recvbuf, recvbuflen, 0);
printf("HVAC value recieved: %f\n",atof(recvbuf));
int hvactocheck = atof(recvbuf);
```

```
if(hvactocheck==1){
  hvac_text = "HVAC is ON";
}
else{
  hvac_text = "HVAC is OFF";
}
//CLOSE THE SOCKET
close(listenSocketForHVAC);


return 0;
```

B.1.2 TCP server

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using System.Net;
using UnityEngine;
using UnityEngine.UI;
using System.Net.Sockets;
using System.Text;
using UnityEngine.SceneManagement;

public class ServerForTest : MonoBehaviour
{
    [SerializeField]
    //private Canvas panelPayment;

    public TcpListener server;

    //Default host and port
    string host = "127.0.0.1";
    int portTest = 52051;
    private string displaymessage = "";
    //public Text connectionStatus;
    //public Text variable;
    public float testfromServer = 0;

    Thread tcpListenerThread;

    // Use this for initialization
    public void Start()
    {
        tcpListenerThread = new Thread(() => ListenForMessages());
        tcpListenerThread.Start();
        displaymessage = "Not Connected!";
    }

    public void Update()
    {
        //displayConnectionStatus();
        //displayVariable();
    }

    public void ListenForMessages()
    {

        try
        {
            // Set the TcpListener on port 13000.
            IPAddress localAddr = IPAddress.Parse(host);
            // TcpListener server = new TcpListener(port);
            server = new TcpListener(localAddr, portTest);

            // Start listening for client requests.
            server.Start();

            // Buffer for reading data
            Byte[] bytes = new Byte[1024];      //The byte array containing the
                sequence of bytes to decode.
            String data = null;
            int counter = 0;
            // Enter the listening loop.
            while (true)
            {
                if (counter == 0)
                {
                    //counterForScene = 0;
                    displaymessage = "Waiting for a connection...";
                    Debug.Log(displaymessage);
                }

                // Perform a blocking call to accept requests.
                // You could also user server.AcceptSocket() here.
                using (TcpClient client = server.AcceptTcpClient())
                {

                    if (counter == 0)
                    {
                        displaymessage = "Connected!";
                        Debug.Log(displaymessage);
                        counter = 1;
                        //counterForScene = 1;
                    }

                    data = null;

                    // Get a stream object for reading and writing
                    NetworkStream stream = client.GetStream();

                    int i;      //The number of bytes to decode

                    // Loop to receive all the data sent by the client.
                    while ((i = stream.Read(bytes, 0, bytes.Length)) != 0)
                    {
                        // Translate data bytes to a ASCII string.
                        data = System.Text.Encoding.ASCII.GetString(bytes, 0, i);
                            // 0 is The index of the first byte to decode.
                        float testfromRhapsody = (float)Convert.ToDouble(data);

                        testfromServer = testfromRhapsody;

                        // Process the data sent by the client.
                        data = data.ToUpper();

                        byte[] msg = System.Text.Encoding.ASCII.GetBytes(data);

                        // Send back a response.
                        stream.Write(msg, 0, msg.Length);
                        //Debug.Log("Test: " + testfromServer.ToString());
                    }
                    // Shutdown and end connection
                    client.Close();
                }


            }
        }
        catch (SocketException e)
        {
            Debug.LogError(String.Format("SocketException: {0}", e));
        }
        finally
        {
            // Stop listening for new clients.
            server.Stop();
        }
    }


}
```

B.2 Digital Modelling layer

B.2.1 'PlayerMovement' script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{   //on player

    public float speed = 6.0f;
    public float gravity = -9.8f;
    private CharacterController _charCont;

    // Use this for initialization
    void Start()
    {
        _charCont = GetComponent<CharacterController>();
    }

    // Update is called once per frame
    void Update()
    {
        float deltaX = Input.GetAxis("Horizontal") * speed;
        float deltaZ = Input.GetAxis("Vertical") * speed;
        Vector3 movement = new Vector3(deltaX, 0, deltaZ);
        movement = Vector3.ClampMagnitude(movement, speed);     //Limits the
            maximum speed of the player

        movement.y = gravity;

        movement *= Time.deltaTime;
        movement = transform.TransformDirection(movement);

        _charCont.Move(movement);
    }
}
```

B.2.2 'CameraControl' script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraControl : MonoBehaviour
{

    public enum RotationAxis
    {
        MouseX = 1, //on player
        MouseY = 2  //on camera
    }

    public RotationAxis axes = RotationAxis.MouseX;

    public float minimumVert = -45.0f;
    public float maximumVert = 45.0f;
    public float sensHorizontal = 10.0f;
    public float sensVertical = 10.0f;

    public float _rotationX = 0;

    // Update is called once per frame
    void Update()
    {
        if (axes == RotationAxis.MouseX)
        {
            transform.Rotate(0, Input.GetAxis("Mouse X") * sensHorizontal, 0);
        }
        else if (axes == RotationAxis.MouseY)
        {
            _rotationX -= Input.GetAxis("Mouse Y") * sensVertical;
            _rotationX = Mathf.Clamp(_rotationX, minimumVert, maximumVert);     //
                Clamps the vertical angle within mix and max limits (45 degrees)
            float rotationY = transform.localEulerAngles.y;

            transform.localEulerAngles = new Vector3(_rotationX, rotationY, 0);
        }
    }
}
```

B.3 Data and Model integration layer

B.3.1 'Door_opening' script

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.SceneManagement;

public class Door_opening : MonoBehaviour
{
    Animator anim;
    // Start is called before the first frame update
    ServerForDoor p1;
    ServerForTest T1;

    void Start()
    {
        anim = GetComponent<Animator>();

        p1 = GameObject.Find("InputManager").GetComponent<ServerForDoor>();
        T1 = GameObject.Find("InputManager").GetComponent<ServerForTest>();
    }

    // Update is called once per frame
    void Update()
    {
        //Animator anim = GetComponent<Animator>();
        if (p1.DoorstatusfromServer == 1)
        {
            anim.SetBool("open", true);
            Debug.Log("Door opened");
        }

        else if (p1.DoorstatusfromServer == 0 && T1.testfromServer == 6)
        {

                //Fireflame.Fire = true;
            StartCoroutine(ExampleCoroutine());
        }
    }

    IEnumerator ExampleCoroutine()
    {
        yield return new WaitForSeconds(20);

        anim.SetBool("open", false);
        Debug.Log("Door closed");
    }
}
```

B.3.2 'FlameChange' script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Flamechange : MonoBehaviour
{
    ServerForTest T1;
    //ChangeText Fire;
    public GameObject goMyObject;
    Renderer myR;
    Vector3 iniScale;
    public AudioSource audiosource;
    // Start is called before the first frame update
    void Start()
    {
        T1 = GameObject.Find("InputManager").GetComponent<ServerForTest>();
        goMyObject = GameObject.Find("FlamesParticleEffect");
        iniScale = transform.localScale;
    }

    // Update is called once per frame
    void Update()
    {
        if (T1.testfromServer == 3 || T1.testfromServer == 4 || T1.testfromServer
            == 5)
        {
            transform.localScale = Vector3.zero;
        }


        if (T1.testfromServer == 6)
        {
;
```

```
            transform.localScale = iniScale;
            if(!audiosource.isPlaying)
            audiosource.Play();
        }
    }
}
```

B.3.3 Collision and Trigger Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CollisionAndTrigger : MonoBehaviour
{
    public Light light1;
    public Light light2;
    public Light light3;
    public Light light4;
    public Light light5;
//    public GameObject light6;

    // Start is called before the first frame update
    void Start()
    {
        light1.enabled = false;
        light2.enabled = false;
        light3.enabled = false;
        light4.enabled = false;
        light5.enabled = false;
    }

    // Update is called once per frame
    void Update()
    {

    }

    private void OnTriggerEnter(Collider other)
    {
        Debug.Log(other.name + " Has Entered!");
        light1.enabled = true;
        light2.enabled = true;
        light3.enabled = true;
        light4.enabled = true;
        light5.enabled = true;
    }
    private void OnTriggerExit(Collider other)
    {
        Debug.Log(other.name + " Has Exited!");
        light1.enabled = false;
        light2.enabled = false;
        light3.enabled = false;
        light4.enabled = false;
        light5.enabled = false;
    }
}
```

Bibliography

[1] O'Connor, C. IBM IoT Platform. Available online: https://www.ibm.com/blogs/internet-of-things/leaderiot- platforms/ (accessed on 28 January 2019).

[2] F. Tao and M. Zhang, "Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing," in IEEE Access, vol. 5, pp. 20418-20427, 2017.

[3] A. Barni, A. Fontana, S. Menato, M. Sorlini and L. Canetta, "Exploiting the Digital Twin in the Assessment and Optimization of Sustainability Performances," 2018 International Conference on Intelligent Systems (IS), Funchal - Madeira, Portugal, 2018, pp. 706-713.

[4] F. Tao, H. Zhang, A. Liu and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," in IEEE Transactions on Industrial Informatics, vol. 15, no. 4, pp. 2405-2415, April 2019.

[5] Madni, A.M.; Madni, C.C.; Lucero, S.D. Leveraging Digital Twin Technology in Model-Based Systems Engineering. Systems 2019, 7, 7.

[6] Lu, Qiuchen & Parlikad, Ajith Kumar & Woodall, Philip & Xie, Xiang & Liang, Zhenglin & Konstantinou, Eirini & Heaton, James & Schooling, Jennifer. (2019). Developing a dynamic digital twin at building and city levels: A case study of the West Cambridge campus. Journal of Management in Engineering. 36. 10.1061/(ASCE)ME.1943-5479.0000763.

[7] Mohammadi, N., Taylor, J. E. (2017). "Smart city digital twins." 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, pp. 1-5.

[8] S. Paladi, MIL, SIL, HIL for Testing Electronic Controls in Vehicle Engineering, Master Internship. 2019.

[9] "The Advantages of Using Digital Twins in HVAC Systems: A Primer", HPAC Engineering, 2020. [Online]. Available: https://www.hpac.com/iaq-ventilation/article/21133096/the-advantages-of-using-digital-twins-in-hvac-systems-a-primer. [Accessed: 08- Jun- 2020]

[10] "Lighting control system", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Lighting_control_system. [Accessed: 08- Jun- 2020]

[11] "Medium. 2020. How To Write A Simple 3D Character Controller In Unity." [online] Available at: ¡https://itnext.io/how-to-write-a-simple-3d-character-controller-in-unity-1a07b954a4ca¿ [Accessed 25 November 2020].

[12] Ibm.com. 2020. Using IBM Rhapsody C/C++ With The Raspberry Pi. [online] Available at: ¡https://www.ibm.com/support/pages/node/275367¿ [Accessed 25 November 2020].