Department of Mathematics and Computer Science
2IMP30 (2020-GS4)
Postbus 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Author
Luo Xuanshu (1618962)
Wang Jingjing (1494880)
Zhao Xin (1329553)

Responsible Lecturer
dr.ir. I. Barosan

Date
November 16, 2021

The Report on the Final Project: Digital Twin of a Smart Room

2IMP30 (2020-GS4) System Design Engineering

SECTION I - PROJECT GR 6

Luo Xuanshu (1618962)
x.luo1@student.tue.nl

Wang Jingjing (1494880)
j.wang8@student.tue.nl

Zhao Xin (1329553)
x.zhao1@student.tue.nl

Technische Universiteit
Eindhoven
University of Technology

Table of contents

Title
The Report on the Final Project: Digital Twin of a Smart Room

1 Introduction **1**
1.1 Components . . . 1
1.2 Stakeholders . . . 1

2 Methodology **2**
2.1 TRIZ 9 boxes . . . 2
2.2 Requirements . . . 3
2.3 TRIZ contradictions and solutions . . . 6
2.4 System context . . . 9
2.5 System architecture . . . 10
2.6 Use cases . . . 10
2.7 System Behaviors . . . 13
2.8 Graphical user interface . . . 14
2.9 Execution and model testing . . . 15

3 Conclusion **15**

Appendix A Other Requirement Diagrams of *PhysicalEntityReq* **17**
A.1 Requirement diagram of 1.1 *IntelligentLampsReq* . . . 17
A.2 Requirement diagram of 1.2 *HVACReq* . . . 17
A.3 Requirement diagram of 1.3 *FireSensorReq* . . . 18
A.4 Requirement diagram of 1.4 *MovementDetectionReq* . . . 18
A.5 Requirement diagram of 1.5 *CO2SensorReq* . . . 18
A.6 Requirement diagram of 1.6 *OccupancySensorReq* . . . 19
A.7 Requirement diagram of 1.7 *CommunicationSysReq* . . . 19
A.8 Requirement diagram of 1.8 *AVSysReq* . . . 19
A.9 Requirement diagram of 1.9 *SASysReq* . . . 20
A.10 Requirement diagram of 1.10 *InteractionReq* . . . 20

1 Introduction

The rapid and significant progress in information technology domains enables the digital evolution of almost everything around us. One of the most influential is the Internet of Things. For instance, intelligent system powered Internet-connected devices in a smart room automatically process various works, allowing people to focus more on work or life. To make full use of the current resources and compress the cost, main-stream plans are upgrading the existing rooms into intelligent ones, inducing emergence of huge demands on the development of digital system, *a.k.a.*Digital Twin of a Smart Room (DTSR). However, due to the complexity, lack of understanding and communication, such development often fails to achieve desired requirements.

Fortunately, this kind of tasks can be formalized as a model. Concretely, models are used to perform abstract analysis towards the domain of interest regardless of the details. With the help of such system engineering approach based on Model-Driven methodology (SYSMOD), we manage to obtain a feasible and reliable way to implement such system in a graceful style [1]. In this report, we will follow the methodology and create a SysML model of DTSR using *IBM Rhapsody* and show the usage of the proposed model by *Unity game engine* [2, 3].

1.1 Components

The desired DTSR system consists of five kinds of components: Physical Entity (PE), Virtual Entity (VE), Digital Data (DT), Services (Ss) and Connections. Figure 1.1 describes the relationships between components using the Agile SCRUM project management framework [4]. With respect to the PE, it includes intelligent lamps, HVAC, fire sensors, movement detection sensors, CO_2 sensors, occupancy sensors, a communication system, an audio-video system, a security and access system, and at least two touch panels for interaction with the users. With the help of these components, the DTSR can automatically control the behaviors of the smart room and provide desired services and security for users.

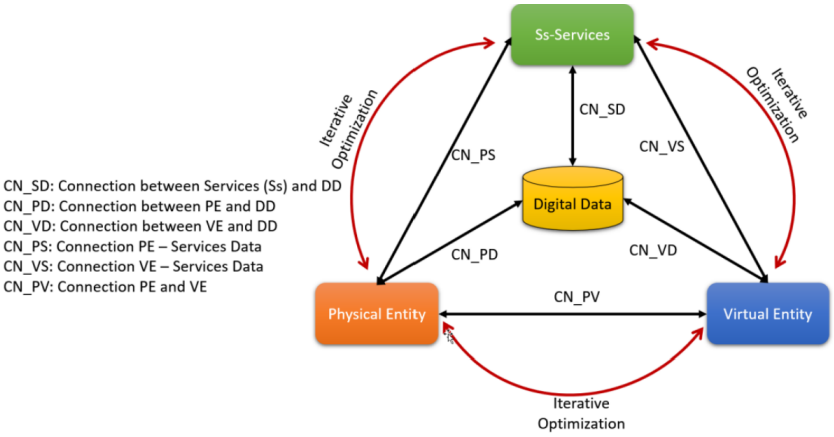

Figure 1.1: The five components of DTSR represented by Agile SCRUM project management framework

1.2 Stakeholders

The stakeholder analysis illustrates different kind of people involved in the system and their corresponding behaviors or interactions. We summarize the stakeholders and their obligation / description in the Table 1.1 below.

Stakeholder	Obligation / Description
Designers	In charge of the design
Developers	System implementation, testing and deployment
Users	Maybe employers and employees, or teachers and students, etc.
Maintenance personnel	System maintenance, repair and replacement
Safety guards	In charge of the security and monitoring potential threats
Administrator	The one who owns the building or the superintendent

Table 1.1: The stakeholders and corresponding obligation / description

2 Methodology

In this chapter, we will follow the methodology given to analyze and implement the model for DTSR step by step.

2.1 TRIZ 9 boxes

Generally, it is hard to fully understand the system requirements, contexts and behavior in the early stage. To deal with this problem, TRIZ 9 boxes tool can help us to identify and understand the project domain problems by specifying sub-system, (current) system, and super-system in pat, present and future stage accordingly [5]. In the end, for this system, we have a 3×3 matrix, namely TRIZ 9 boxes depicted in Figure 2.1:

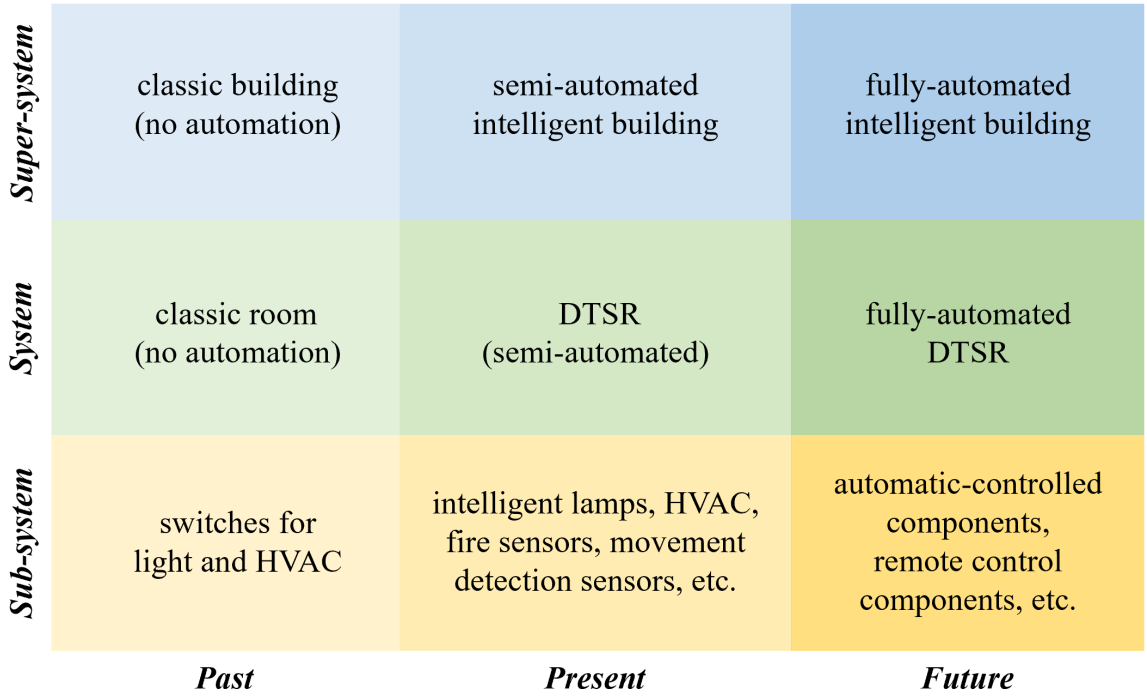

Figure 2.1: TRIZ 9 boxes matrix of the DTSR

With the help of the TRIZ 9 boxes, we can manage to understand the DTSR system clearly, and have powerful guidance for the later analysis and implementations.

2.2 Requirements

TRIZ 9 boxes help us to identify domain problems of the DTSR, facilitating the specification of the requirements. In this section, we summarize all the grouped requirements of the DTSR in plain texts.

1. *PhysicalEntityReq* (its requirement diagrams are shown in Figure 2.2 and Appendix A)

   1.1 *IntelligentLampsReq*

   1.1.1 The system shall install intelligent lamps.
   1.1.2 The system shall modify the lights in the room based on the human presence.
   1.1.3 The system shall modify the lights in the room based on external lighting.
   1.1.4 The system shall be capable to monitor external lighting.
   1.1.5 The system shall turn off the lights when people are not detected in the room.
   1.1.6 The system shall turn off the lights when external lighting is sufficient.
   1.1.7 The system shall turn on the lights when people are detected in the room.
   1.1.8 The system shall turn on the lights when external lighting is insufficient.

   1.2 *HVACReq*

   1.2.1 The system shall install an HVAC.
   1.2.2 The system shall be capable to monitor the temperature in the room.
   1.2.3 The system shall turn off the HVAC if nobody is inside.
   1.2.4 The system shall turn on the cooler if the temperature in the room is higher than a certain threshold (`tempt_th`) and there is someone inside.
   1.2.5 The system shall turn on the heater if the temperature in the room is lower than a certain threshold (`tempt_th`) and there is someone inside.
   1.2.6 The system shall turn off the heater or the cooler if the temperature in the room is at a certain threshold (`tempt_th`).
   1.2.7 The system shall switch on ventilation in the HVAC when CO_2 concentration is higher than a certain threshold (`CO2_th`) and there is someone inside.
   1.2.8 The system shall switch off ventilation in the HVAC when CO_2 concentration is no more than a certain threshold (`CO2_th`) and there is someone inside.

   1.3 *FireSensorReq*

   1.3.1 The system shall install fire sensors.
   1.3.2 The system shall trigger an alarm whenever a fire is detected in the room.
   1.3.3 The system shall switch off the HVAC whenever a fire is detected in the room.

   1.4 *MovementDetectionReq*

   1.4.1 The system shall install movement detection sensors.
   1.4.2 The system shall modify the lights in the room based on the human presence.

   1.5 *CO2SensorReq*

   1.5.1 The system shall install CO_2 sensors.
   1.5.2 The system shall be capable to obtain CO_2 concentration.

   1.6 *OccupancySensorReq*

   1.6.1 The system shall install occupancy sensors.
   1.6.2 The system shall be capable to detect the number of people in the DTSR.

   1.7 *CommunicationSysReq*

   1.7.1 The system shall install a communication system.

1.7.2 The system shall support audio-video system.

1.8 *AVSysReq*

1.8.1 The system shall have an audio-video system.

1.9 *SASysReq*

1.9.1 The system shall have security access system.

1.9.2 The system shall be capable to be accessed safely.

1.9.3 The system shall support user authentication.

1.9.4 The system shall keep the door locked if the authentication fails.

1.9.5 The system shall unlock the door if the authentication passes.

1.10 *InteractionReq*

1.10.1 The system shall provide at least two touch panels for interaction with the users.

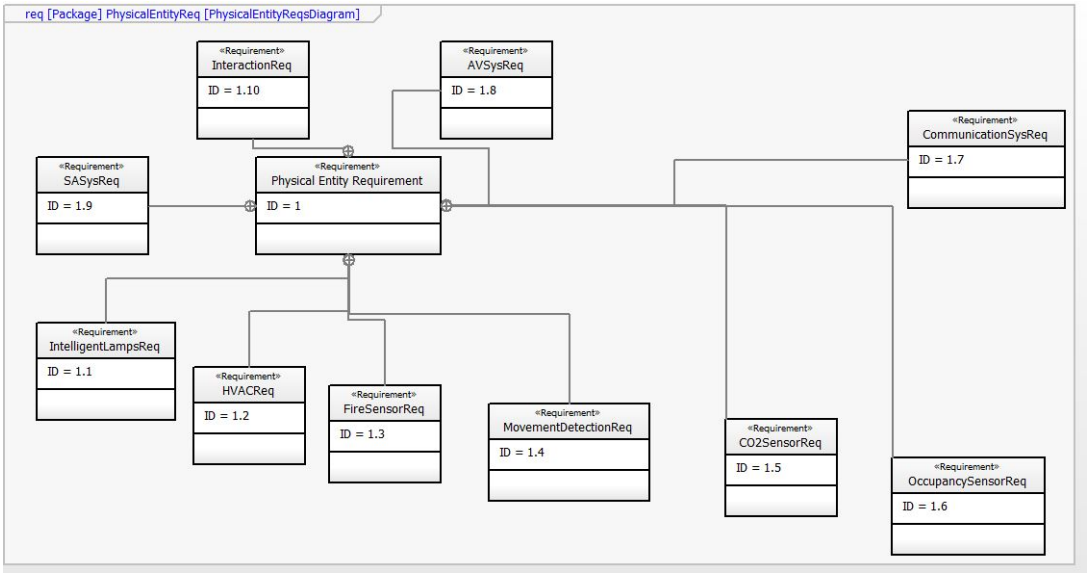

Figure 2.2: The requirement diagram of *PhysicalEntityReq*

2. *VirtualEntityReq* (its requirement diagrams are shown in Figure 2.3)

2.1 The system shall provide 3D CAD-CAM models of the PE.

2.2 The system shall provide models of the software controllers.

3. *DigitalDataReq* (its requirement diagrams are shown in Figure 2.4)

3.1 The system shall be capable to gather data from PE.

3.2 The system shall be capable to gather data from VE.

4. *ServiceReq* (its requirement diagrams are shown in Figure 2.5)

4.1 The system shall provide services to the user.

4.2 The system shall provide services to the PE.

4.3 The system shall provide services to the VE.

4.4 The system shall have the ability to monitor the air quality.

4.5 The system shall have the ability to control the air quality.

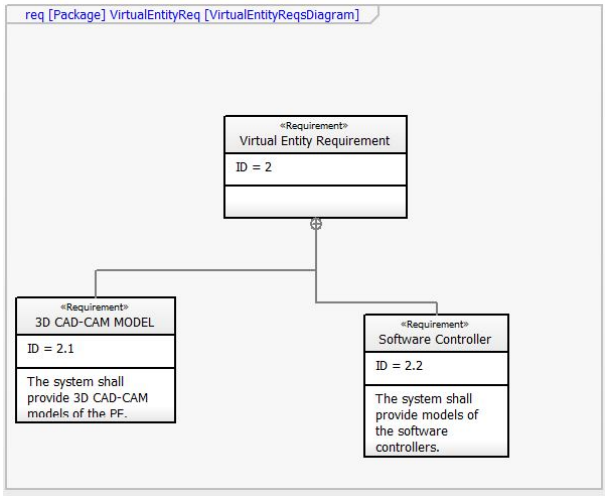

Figure 2.3: The requirement diagram of *VirtualEntityReq*

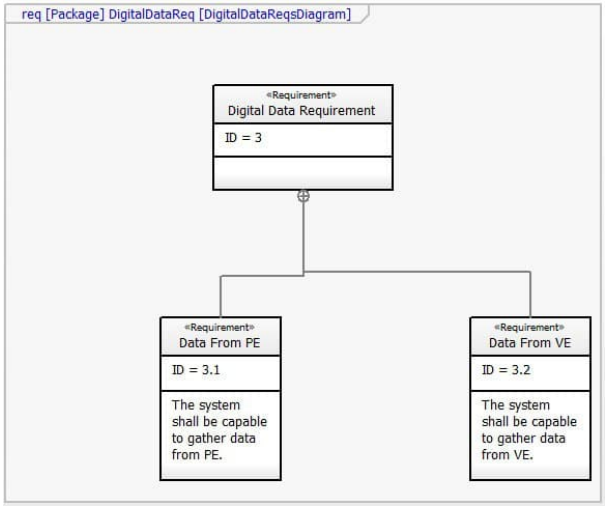

Figure 2.4: The requirement diagram of *DigitalDataReq*

4.6 The system shall acquire weather prediction.

4.7 The system shall control the temperature in the room according to the weather outside.

4.8 The system shall support remote control to manipulate all the services in the DTSR.

5. *ConnectionReq* (its requirement diagrams are shown in Figure 2.6)

   5.1 The system shall utilize convincing cyber-security technology.

   5.2 The system shall be of enough upstream and downstream bandwidth.

   5.3 The system shall provide high-level QoS *w.r.t* network.

   5.4 The system shall support TCP/IP communication.

Note that *PhysicalEntityReq*, *VirtualEntityReq*, *DigitalDataReq*, *ServiceReq* and *ConnectionReq* are parts of top-level requirements, namely *DTSRReq*. The rest of the corresponding requirement diagrams are shown in Appendix A.

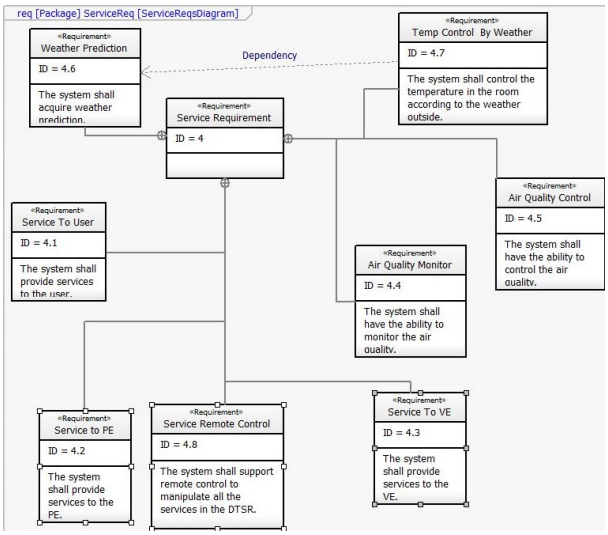

Figure 2.5: The requirement diagram of *ServiceReq*

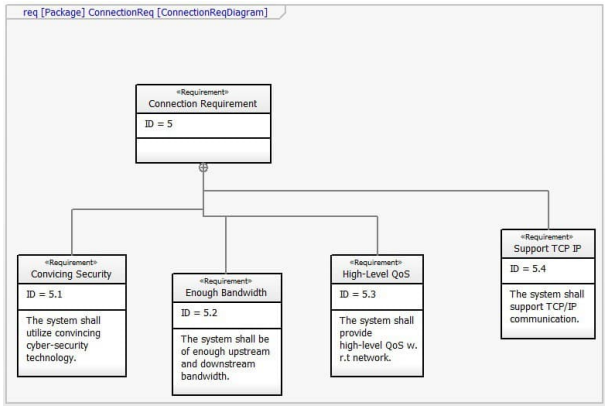

Figure 2.6: The requirement diagram of *ConnectionReq*

2.3 TRIZ contradictions and solutions

Ideally, we intend to achieve all the good properties of the system. But in practice, trade-offs between properties or parameters exist commonly. Therefore, it is crucial for us to analyze the contradictions and try to solve them using TRIZ contradiction matrix. In this section, we list the contradictions within this system and provide solutions accordingly. Note that for the feature part, we choose one or more from *The 39 Features of Altshuller's Contradiction Matrix* [6]. And solutions are selected from *TRIZ 40 inventive principles* [7].

1. The intelligent lamps are expected to provide optimal lighting, but resulting in high energy consumption.
    - Improving feature(s): 18 (illumination intensity)
    - Contradict feature(s): 22 (loss of energy)
    - Solution(s):
        - 1 (segmentation): split the room into multiple parts and manage lighting separately
        - 16 (partial or excessive action): only provide acceptable lighting in most areas
2. The intelligent lamps are expected to save energy, but resulting in low lighting quality.

- Improving feature(s): 21 (power)
- Contradict feature(s): 18 (illumination intensity)
- Solution(s):
  - 1 (segmentation): split the room into multiple parts and save energy separately
  - 16 (partial or excessive action): only provide acceptable energy consumption policy in most cases

3. We want to utilize a modified room shape to make full use of daylight, but resulting in high manufacturing complexity and cost, harder still, difficulties in changing the structure of the room.
   - Improving feature(s): 12 (shape)
   - Contradict feature(s): 32 (ease of manufacture)
   - Solution(s):
     - 1 (segmentation): split the room into multiple parts and improve them separately
     - 32 (changing color or optical properties): use glass with better light transmission
4. We want to save energy usage, but resulting in low cooling / heating speed.
   - Improving feature(s): 21 (power)
   - Contradict feature(s): 9 (speed)
   - Solution(s):
     - 35 (parameter change): use acceptable cooling / heating speed.
5. We want to make the system problem-free and completely predictable, but resulting in increase of the complexity.
   - Improving feature(s): 27 (reliability)
   - Contradict feature(s): 36 (device complexity)
   - Solution(s):
     - 1 (segmentation): separate the room into multiple parts and analyze them separately.
6. We want all the sensors can provide real-time accurate monitoring results, but resulting in high power consumption and high complexity to deal with multiple data stream.
   - Improving feature(s): 28 (measurement accuracy)
   - Contradict feature(s): 21 (power), 36 (device complexity)
   - Solution(s):
     - 3 (local quality): categorize the sensors based on their functions, set proper refresh rate accordingly.
7. We want the DTSR can be easily to transformed from normal rooms, but resulting in low completion of the desired functions.
   - Improving feature(s): 32 (ease of manufacture)
   - Contradict feature(s): 35 (adaptability and versatility)
   - Solution(s):
     - 13 (the other way round): try to make full use of the existing hardware and software case by case.
8. The DTSRs are expected to have optimal user experiences, but that could lead to a longer development and investigation time.

- Improving feature(s): 33 (ease of operation)
- Contradict feature(s): 32 (ease of manufacture)
- Solution(s):
  - 5 (merging): The developer should arrange the work carefully to achieve parallel implementation.

9. We want the system is easy to maintain / repair, but resulting in difficulties to design such systems.
   - Improving feature(s): 34 (ease of repair)
   - Contradict feature(s): 32 (ease of manufacture)
   - Solution(s):
     - 1 (segmentation): only focusing on vulnerable parts is sufficient for most cases.
     - 10 (prior action): prepare enough alternative replacement parts in advance.
     - 35 (parameter changes): we can use acceptable parameters to reduce fault rate. For example, set lower speed for fans to extend service life.
10. We want our design can adapt all kinds of traditional rooms, but resulting in complexity in design and development.
    - Improving feature(s): 35 (adaptability or versatility)
    - Contradict feature(s): 32 (ease of manufacture)
    - Solution(s):
      - 1 (segmentation): We can separate the whole design into multiple parts and try to enhance the adaptability separately.
11. We want our system is easy to use, namely automated control as much as we can. But it will cause complexity in design and development.
    - Improving feature(s): 38 (extent of automation)
    - Contradict feature(s): 32 (ease of manufacture)
    - Solution(s):
      - 1 (segmentation): We can separate the whole design into multiple parts and try to enhance the extent of automation separately.
      - 26 (copying): We can investigate mature products that are compatible with our system, and substitute parts that are easy to control or develop.
12. We want our system is easy to use, namely automated control as much as we can. But it will cause complexity in maintenance and repair.
    - Improving feature(s): 38 (extent of automation)
    - Contradict feature(s): 34 (ease of repair)
    - Solution(s):
      - 1 (segmentation): We can separate the whole design into multiple parts and try to enhance the repairability accordingly.
      - 35 (parameter changes): we can use acceptable parameters to reduce fault rate. For example, set lower speed for fans to extend service life.
13. We want our system is easy to use, namely automated control as much as we can. But users may not adapt to such systems due to strangeness.

- Improving feature(s): 38 (extent of automation)
- Contradict feature(s): 33 (convenience of use)
- Solution(s):
  - 1 (segmentation): We can separate the whole design into multiple parts and try to enhance the convenience accordingly.
  - 10 (prior action): We should pay enough time and effort on the user interface and experiences in advance to minimize the potential gaps.

14. We want our system is easy to use, namely automated control as much as we can. But the subsystems are tend to be complex and hard to integrated as a whole.

    - Improving feature(s): 38 (extent of automation)
    - Contradict feature(s): 36 (device complexity)
    - Solution(s):
      - 10 (prior action): Extensive and in-depth investigation on the choice of sub-system to make sure that they can cooperate gracefully.

2.4 System context

The system context diagram shown in Figure 2.7 clearly illustrates the interactions among users, environment, and DTSR components organized structurally. Concretely, the DTSR is located in the middle of the diagram as a block, and other components are all around the DTSR block. The environment consists of temperature, weather and brightness of external lighting.

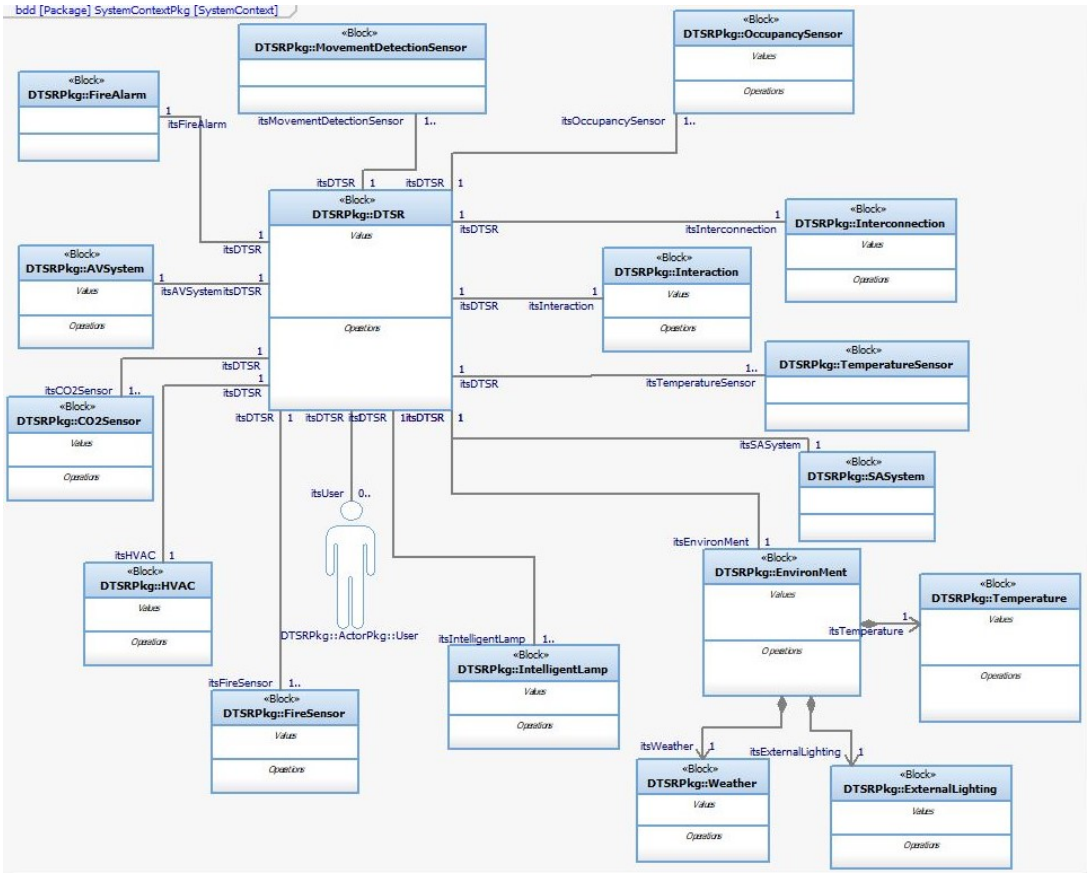

Figure 2.7: The system context diagram of the DTSR

2.5 System architecture

The system architecture of the DTSR system is shown in Figure 2.8. We can observe that apart from the key components of the DTSR system, we also add use cases that will be explained in detail in the next section as blocks.

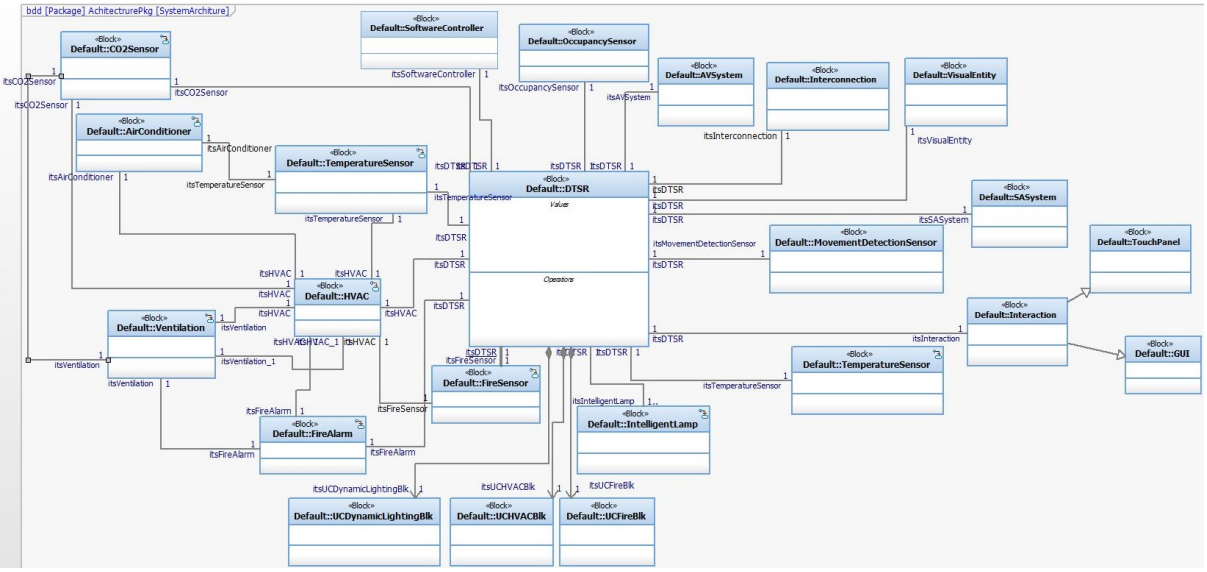

Figure 2.8: The system architecture of the DTSR

2.6 Use cases

To specify the exact functions of the DTSR system, in this chapter, we list the use cases and describe the corresponding actors and behaviors in detail. The use case diagrams will be shown accordingly.

1. Name: *UC_DynamicLighting* (the corresponding use case diagram is shown in Figure 2.9)

Goal	The light will be turned on or off based on the location of the user in the DTSR.
Main actor(s)	MovementDetectionSensors, OccupancySensor
Secondary actor(s)	IntelligentLamps
Pre-condition(s)	The movement detection sensors are installed and work properly.
Trigger	The state of the movement detection sensor changes.
Main flow	1. The movement detection sensor finds someone is in a certain area. 2. The intelligent light at that area will be turned on.
Secondary flow	1. The movement detection sensor finds nobody is in a certain area. 2. The intelligent light at that area will be turned off.

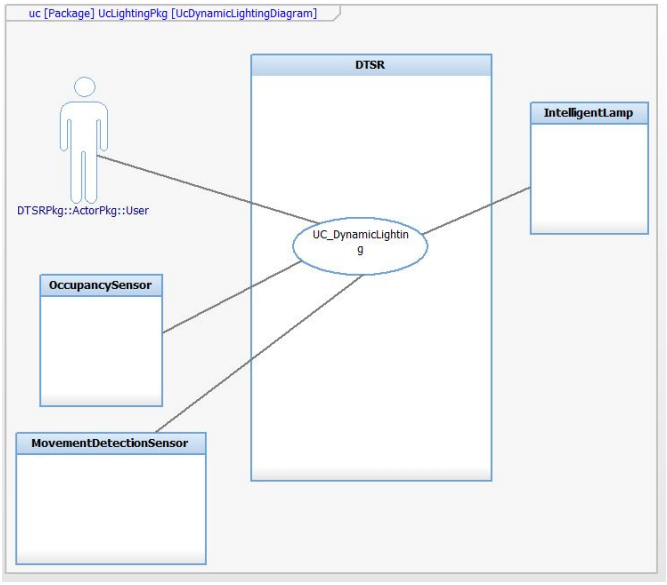

Figure 2.9: The use case diagram of *UC_DynamicLighting*

2. Name: *UC_HVAC* (the corresponding use case diagram is shown in Figure 2.10)

Goal	1. Keep the CO_2 concentration lower than `CO2_th` in the DTSR. 2.Keep the temperature in the DTSR at `temp_th`.
Main actor(s)	TemperatureSensor, CO2Sensor, OccupancySensor
Secondary actor(s)	HVAC
Pre-condition(s)	1. The HVAC is installed and works properly. 2. The movement detection and occupancy sensors are installed and work properly.
Trigger	1. The CO_2 concentration is detected to be higher than `CO2_th`. Or 2. The temperature is detected to be different from `temp_th`.
Main flow	If CO_2 concentration is detected to be higher than `CO2_th`: - The control logic of *UC_ControlCO_2* will be triggered If The temperature is detected to be different from `temp_th`: - The control logic of *UC_AirConditioner* will be triggered.
Secondary flow	If nobody is in the DTSR, no actions will happen.

3. Name: *UC_ControlCO_2* (the corresponding use case diagram is shown in Figure 2.10)

Goal	Keep the CO_2 concentration lower than `CO2_th` in the DTSR.
Main actor(s)	CO2Sensor, OccupancySensor
Secondary actor(s)	HVAC
Pre-condition(s)	1. CO_2 sensors are installed and functioning normally. 2. The ventilation in HVAC works properly.
Trigger	The CO_2 concentration is detected to be higher than `CO2_th`.
Main flow	1. The ventilation in HVAC will be turned on. 2. After `time_intv_HVAC`, re-check the CO_2 concentration. 3. If the CO_2 concentration is still higher than `CO2_th`, go to 1 4. If the CO_2 concentration is no more than `CO2_th`, turn off the ventilation
Secondary flow	If nobody is in the DTSR, no actions will happen.

4. Name: *UC_AirConditioner* (the corresponding use case diagram is shown in Figure 2.10)

Goal	Keep the temperature in the DTSR at `temp_th`.
Main actor(s)	TemperatureSensor, OccupancySensor
Secondary actor(s)	HVAC
Pre-condition(s)	1. Temperature sensors are installed and functioning normally. 2. The HVAC works properly.
Trigger	The temperature is detected to be different from `temp_th`.
Main flow	If somebody is in the DTSR and the temperature is higher than `temp_th`: 1. The cooler in HVAC will be turned on. 2. After `time_intv_HVAC`, re-check the temperature. 3. If the temperature is still higher than `temp_th`, go to 1 4. If the temperature is not higher than `temp_th`, turn off the cooler If somebody is in the DTSR and the temperature is lower than `temp_th`: 1. The heater in HVAC will be turned on. 2. After `time_intv_HVAC`, re-check the temperature. 3. If the temperature is still lower than `temp_th`, go to 1 4. If the temperature is not lower than `temp_th`, turn off the heater
Secondary flow	If nobody is in the DTSR, no actions will happen.

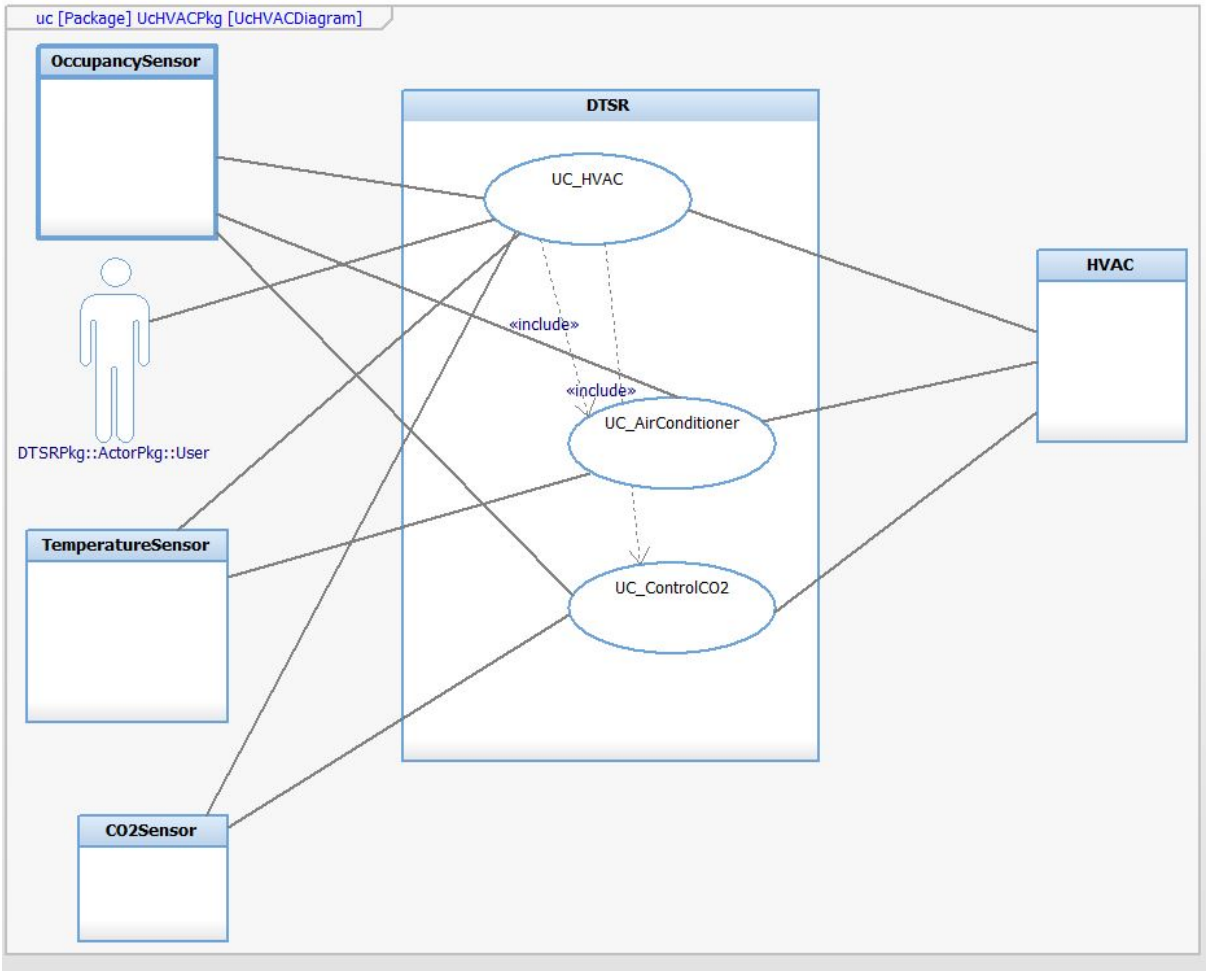

Figure 2.10: The use case diagram of *UC_HVAC*, *UC_ControlCO*$_2$ and *UC_AirConditioner*

5. Name: *UC_FireAlarm* (the corresponding use case diagram is shown in Figure 2.11)

Goal	When a fire breaks out, the alarm will be triggered.
Main actor(s)	FireSensor
Secondary actor(s)	FireAlarm
Pre-condition(s)	Both the fire alarm and fire sensors are installed and working properly.
Trigger	A fire is monitored.
Main flow	If the fire alarm is not triggered: 1. The fire sensor constantly detects fire for `time_intv_fire`. 2. The fire alarm will be triggered.
Secondary flow	If the fire alarm is triggered: 1. The fire sensor constantly detects no fire for `time_intv_fire`. 2. The fire alarm will be turned off.

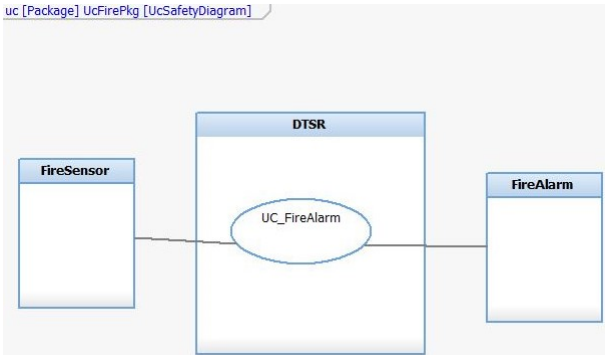

Figure 2.11: The use case diagram of *UC_FireAlarm*

2.7 System Behaviors

To specify the behavior of the DTSR system, we first define all the use cases based on requirements and TRIZ methodology, then model the behavior of all use cases using state machine, sequence, or activity diagrams. In this section, we will provide the behaviors of the use cases we described in the previous section as follows:

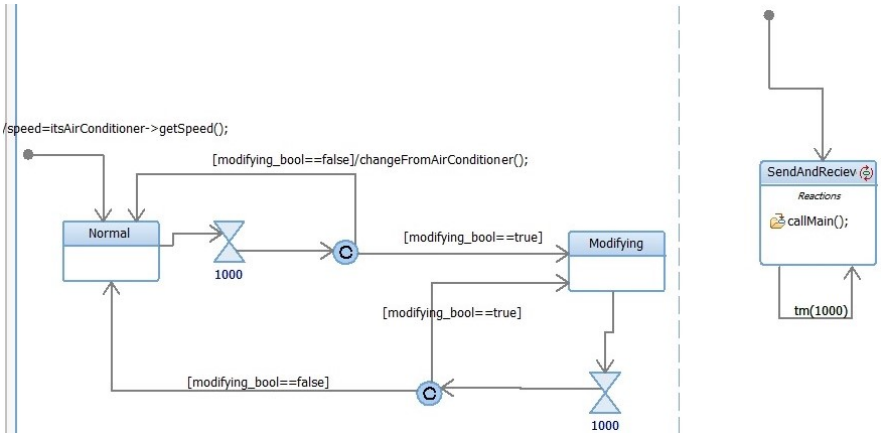

Figure 2.12: The system behavior of *UC_AirConditioner*.

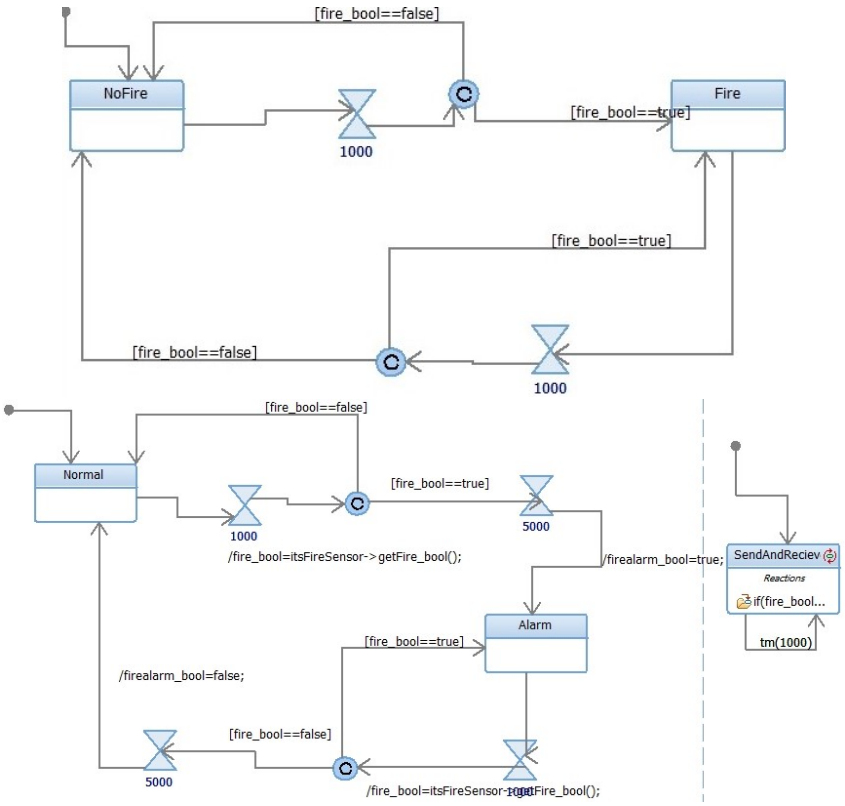

Figure 2.13: The system behavior of *UC_FireAlarm*.

2.8 Graphical user interface

The GUI of the DTSR system is illustrated in Figure 2.15. On the top of the GUI, the occupancy status of the room is provided (in our design, 1 means no people inside, and 0 for existence of users). The *light* slider below can control the brightness of the intelligent lamps. The left side of the GUI is designed for ventilation control to show the real and target CO_2 concentration. The knob is used to model the CO_2 concentration monitored by CO_2 sensors. Similarly, on the right side, the real and target temperature in the room are given, and the knob is used to control the target temperature. The switch in the middle is used to model the fire monitored by fire sensors. When switched on, a fire happens. With the help of the GUI panel, we can easily model the behavior of the DTSR system.

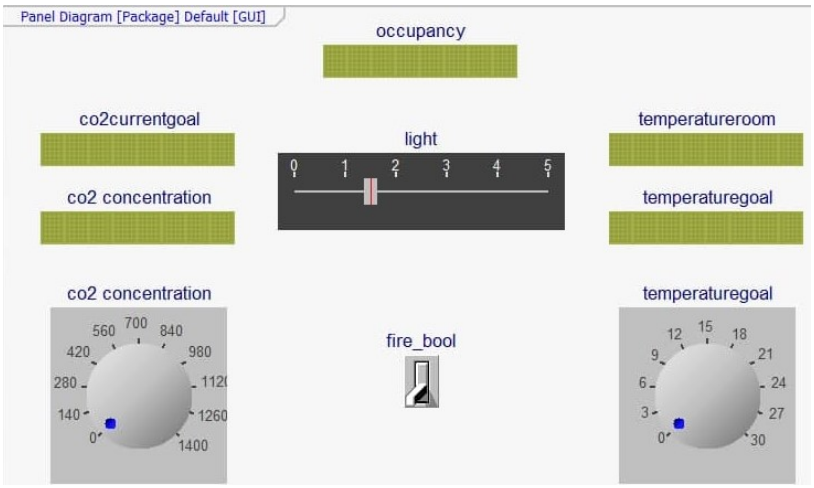

Figure 2.15: The graphical user interface of the DTSR.

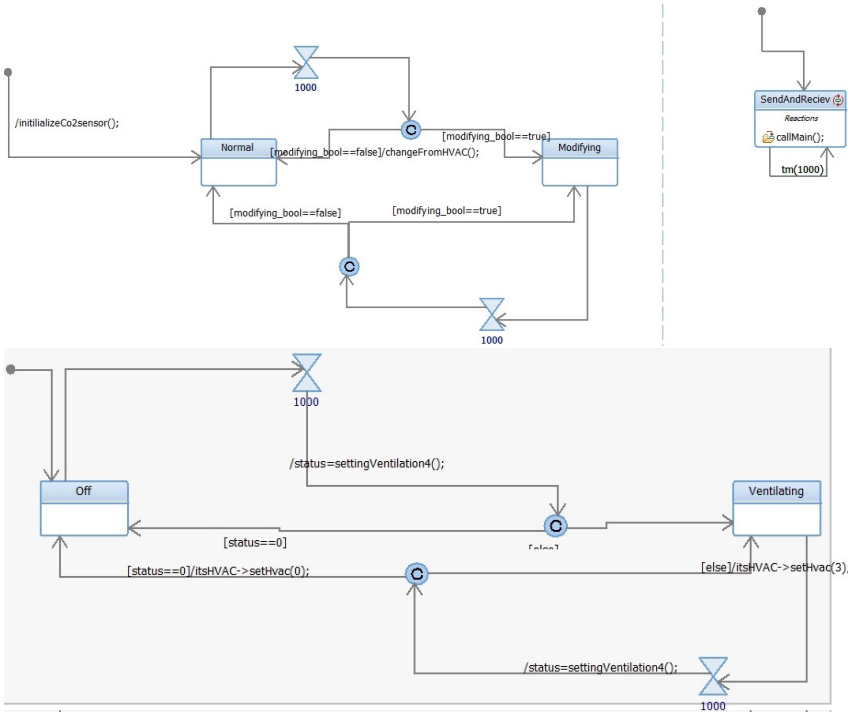

Figure 2.14: The system behavior of *UC_ControlCO*$_2$.

2.9 Execution and model testing

After building all the components described above, we can execute our model and check the correctness of our model. We have tested our DTSR model in the following environment.

Operation system	Windows 10 Home 10.0.18363
IBM Rhapsody	9.0.0.R00_202002261110
Visual Studio	Community 2017, Version 15.9.36
Unity	2019.1.0f2 Personal

Our testing results can be found in the attached video.

3 Conclusion

In this report, we perform analysis and implementation steps of building a DTSR system using *SysML* methodology by *IBM Rhapsody*. First, TRIZ 9 boxes analysis helps us to identify and understand the demands and functions of the DTSR, facilitating us to list the requirements. Then we utilize TRIZ 40 inventive principles to analyze and solve the contradictions between properties and parameters of components. The system context diagram provides clear illustration of the interactions between the components, users and environment. And the system architecture shows the relationships between the components within the DTSR. To specify the DTSR behaviors, we first define the use cases of the DTSR, namely *UC_DynamicLighting*, *UC_ControlCO*$_2$, *UC_AirConditioner*, *UC_HVAC*, and *UC_FireAlarm*, then give corresponding modeling of the behaviors using activity diagrams, sequence diagrams, or state machine diagrams. Finally, we show the GUI of the DTSR. Apart from this report, we also handed on the project files, a slides to show the results and a video to show and verify the functions. They demonstrate the correctness and feasibility of the DTSR system we designed.

Bibliography

[1] Tim Weilkiens. *SYSMOD-The systems modeling toolbox-pragmatic MBSE with SysML*. Lulu. com, 2016.

[2] Hans-Peter Hoffmann. Deploying model-based systems engineering with ibm® rational® solutions for systems and software engineering. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 1–8. IEEE, 2012.

[3] Will Goldstone. *Unity game development essentials*. Packt Publishing Ltd, 2009.

[4] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[5] Imoh M Ilevbare, David Probert, and Robert Phaal. A review of triz, and its benefits and challenges in practice. *Technovation*, 33(2-3):30–37, 2013.

[6] Ellen Domb, Joe Miller, Ellen MacGran, and M Slocum. The 39 features of altshuller's contradiction matrix. *The TRIZ Journal*, 11(11):10–12, 1998.

[7] Nadhmi Gazem and Azizah Abdul Rahman. Improving triz 40 inventive principles grouping in redesign service approaches. *Asian Social Science*, 10(17):127–138, 2014.

A Other Requirement Diagrams of *PhysicalEntityReq*

Here we provide other requirement diagrams of *PhysicalEntityReq*. They are basically diagrams of secondary requirements.

A.1 Requirement diagram of 1.1 *IntelligentLampsReq*

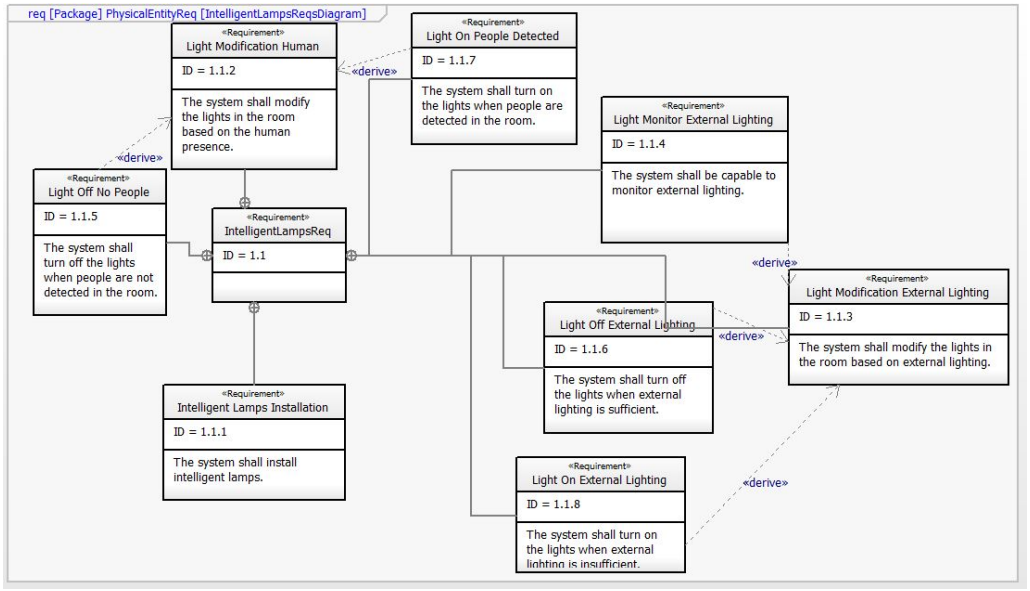

Figure A.1: The requirement diagram of 1.1 *IntelligentLampsReq*

A.2 Requirement diagram of 1.2 *HVACReq*

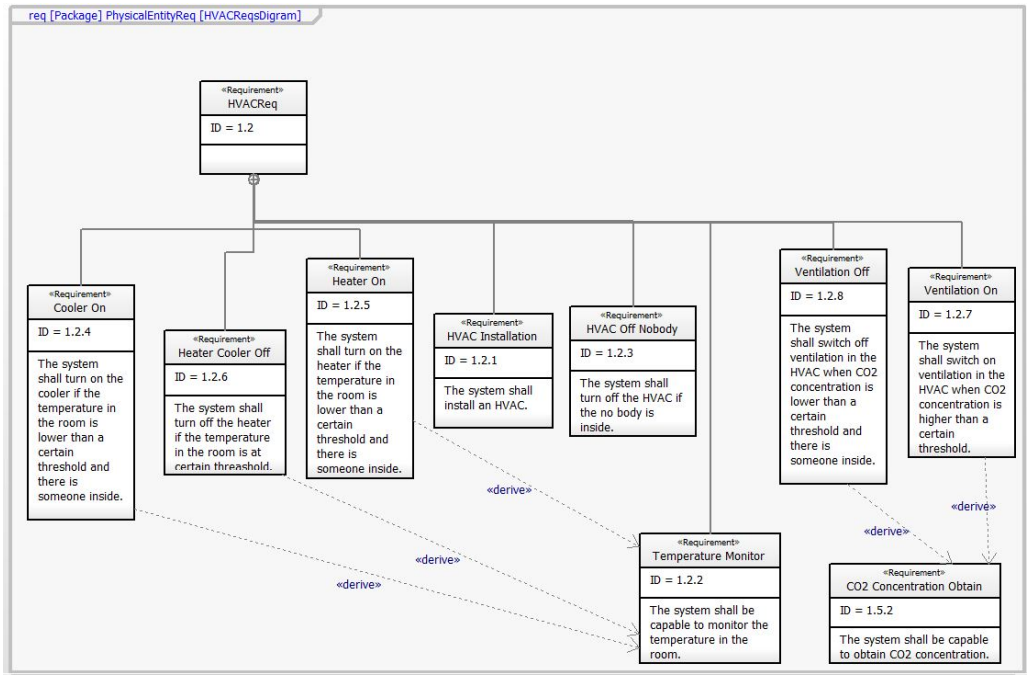

Figure A.2: The requirement diagram of 1.2 *HVACReq*

A.3 Requirement diagram of 1.3 *FireSensorReq*

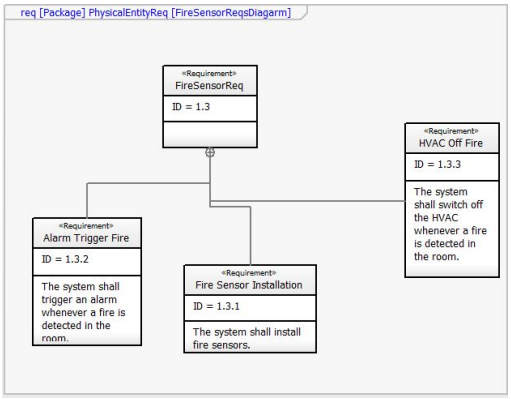

Figure A.3: The requirement diagram of 1.3 *FireSensorReq*

A.4 Requirement diagram of 1.4 *MovementDetectionReq*

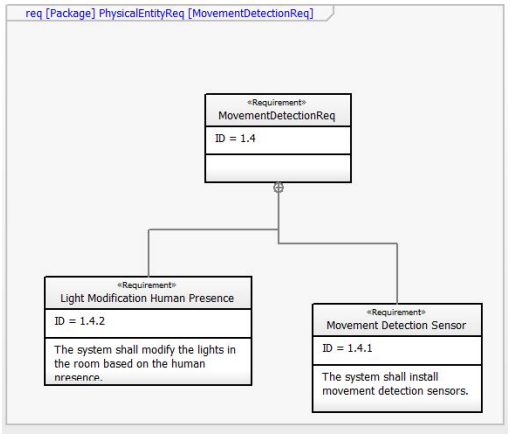

Figure A.4: The requirement diagram of 1.4 *MovementDetectionReq*

A.5 Requirement diagram of 1.5 *CO2SensorReq*

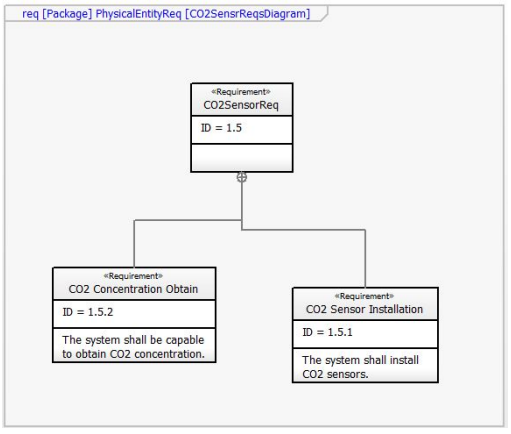

Figure A.5: The requirement diagram of 1.5 *CO2SensorReq*

A.6 Requirement diagram of 1.6 *OccupancySensorReq*

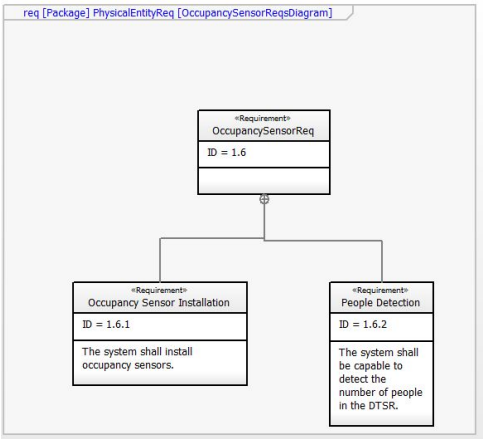

Figure A.6: The requirement diagram of 1.6 *OccupancySensorReq*

A.7 Requirement diagram of 1.7 *CommunicationSysReq*

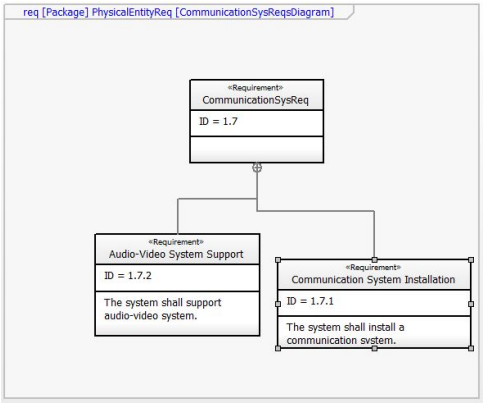

Figure A.7: The requirement diagram of 1.7 *CommunicationSysReq*

A.8 Requirement diagram of 1.8 *AVSysReq*

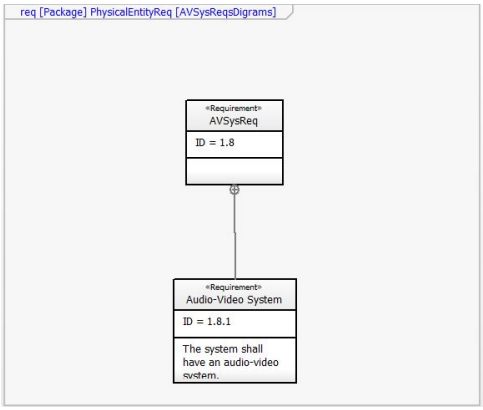

Figure A.8: The requirement diagram of 1.8 *AVSysReq*

A.9 Requirement diagram of 1.9 *SASysReq*

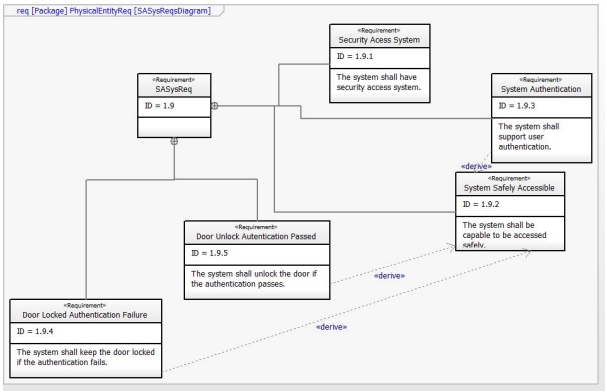

Figure A.9: The requirement diagram of 1.9 *SASysReq*

A.10 Requirement diagram of 1.10 *InteractionReq*

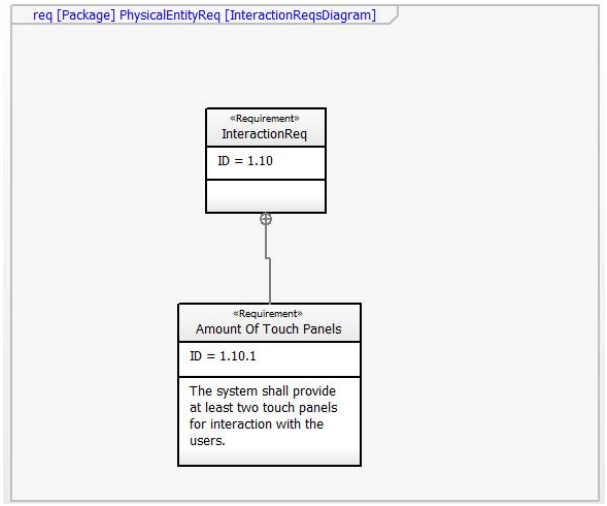

Figure A.10: The requirement diagram of 1.10 *InteractionReq*