Modeling of Autonomous City Drive System using MBSE concepts

Ananya Reginald Frederick [1], Ekaterina Bukharova [2], Joseph Tenesso fougou[3], Utkarsha Fegade[4], Vignesh Somasundaram[5]

1 Introduction

This paper presents how the concepts of Model-Based System Engineering are used in order to derive a solution for the development of Automotive systems. "Model-Based Engineering (MBE) is an approach to engineering that uses models as an integral part of the technical baseline that includes the requirements, analysis, design, implementation, and verification of a capability, system, and/or product throughout the acquisition life cycle." (Final Report, Model-Based Engineering Subcommittee, NDIA, Feb. 2011). Model-Based System Engineering eliminates the communication by relying on abstract models that makes it easier for engineering teams to communicate design intent, understand the impact of design changes, and analyze a system's design before building it. Furthermore, MBSE helps to define the system and, based on the results of modeling, to understand how the system will behave in certain conditions.

Model-Based system Engineering has been used in this work in order to develop an Autonomous City Drive system. Autonomous driving vehicle is capable of sensing its environment and moving safely with little or no human input.[4] In this work the Autonomous City Drive system will be implemented using Embedded Technology, and it needs to be a self-optimizing because an autonomous reaction to the dynamic city environment scenarios is expected. This high-level description of the system emphasizes on defining standard solutions in the conceptual design phase using different views on a system including modelling of use cases using 3D-Engineering tool, applying functional safety standards and security level approaches, derivation of requirements from use cases and Scenarios in the loop supported modeling.

The paper has been divided into 2 sections. The first is the Knowledge which is the theoretical concepts which were used to develop the user story and the second is the project which includes different models, diagrams created during application of those concepts.

[1] FH-Dortmund ananya.frederick005@stud.fh-dortmund.de
[2] FH-Dortmund ekaterina.bukharova001@stud.fh-dortmund.de
[3] FH-Dortmund joseph.tenessofougou001@stud.fh-dortmund.de
[4] FH-Dortmund utkarsha.fegade001@stud.fh-dortmund.de
[5] FH-Dortmund vignesh.somasundaram001@stud.fh-dortmund.de

2 Knowledge

2.1 Automotive software development

Software development process is the process of dividing software development work into distinct phases to improve design, implementation, and integration. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team. In the software development life cycle, various models are designed and defined.

2.1.1 V-model

The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing.

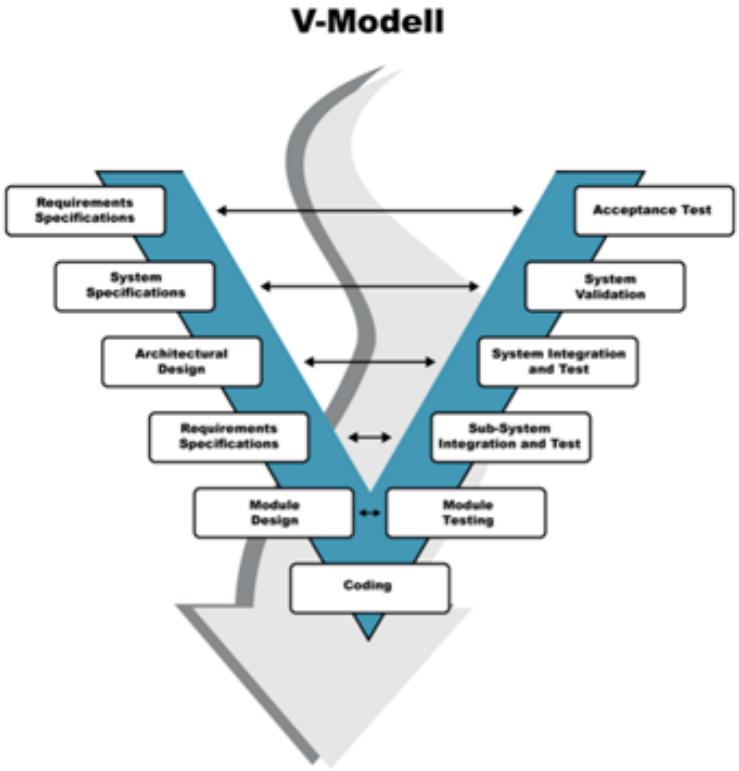

Figure. 1: V-Model[5]

For example, the development of electronic control units can be aligned with v-model technique. At the beginning the Development team receives the first set of requirements and tries to figure out how these requirements need to be detailed for system development. After this Requirements Specification for this control unit is done, the Development team needs to define System Specification, which is a starting point for the System development. Then the Architectural design phase begins with having requirements that are suited to design different modules and once the software modules designed, coding part can be started. On the right-hand side of the v-model there are a lot of testing and integration activities: on the Module design step, on Sub-system and on System levels. Development team needs to continuously provide information back to the initial phases of development to fix all the details in different modules and to fill some gaps and uncertainties up, that were unknown at the beginning. When the Validation phase begins, there are closed loops, where the Development team continuously sends information to the initial phases and creates more specific requirements and more concrete architecture.

2.1.2 Automotive SPICE process reference model - Overview

Automotive SPICE standard in the development process in general is used to see and to validate how the single companies and development departments are working according to the development standards. As there is a high relationship between different disciplines, one cannot talk about Software Engineering without System Engineering. That is why on the Automotive SPICE model there are blocks with System Engineering Processes and Software Engineering Processes. In the System Engineering Process Group there are development process steps like Requirements Elicitation and System Requirements Analysis that is also present on the Software Engineering processes. Usually the development process starts with requirements that are provided by the OEM and then these requirements are needed to be elicited and analyzed. Purpose of requirements elicitation is to gather, process and track evolving stakeholders needs and requirements throughout the lifecycle of the product to establish a requirements baseline that serves as the basis for defining the needed work products. Purpose of the System Requirements Analysis is to transform the defined stakeholder requirements into a set of system requirements that will guide the design of the system. It is an important step because if we find out inconsistencies, ambiguities, or errors in the requirements after we start implementation, it is more costly to fix this system. Once the first system requirements are derived, we need to break it down to Software requirements. The purpose of this Software Requirements step is to transform the software related parts of the system requirements into a set of SW requirements. For example, if we have a clear mechanical requirement (“the door must be able to close”), we need to analyze what functionality is provided by mechanical components and what can be implemented by Software. As we saw in the v-model representation in the Software Engineering processes group there are also integration, validation, and verification phases. At the System Qualification test phase, we need to ensure that the system’s objective and the expectation of customers are fulfilled.

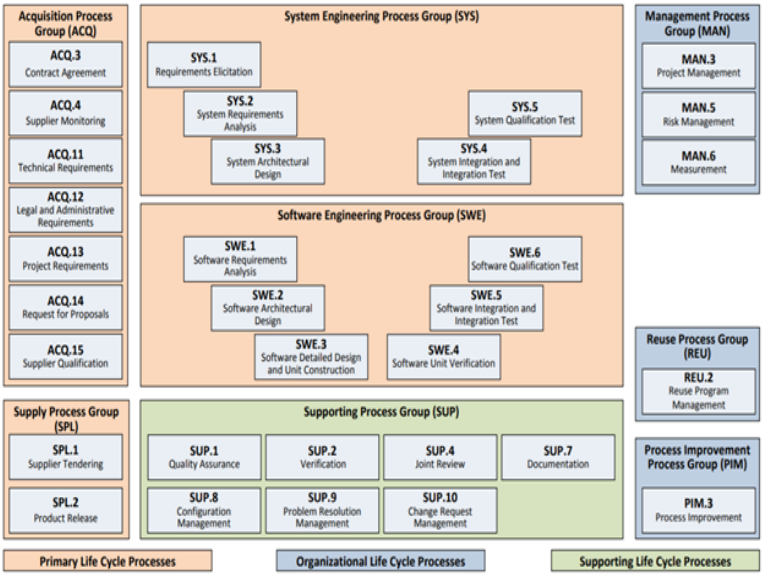

Figure. 2: Automotive SPICE process reference model[1]

2.2 AUTOSAR

AUTOSAR stands for Automotive Open System Architecture. It is a worldwide development partnership of vehicle manufacturers, suppliers, service providers and companies from the automotive electronics, semiconductor and software industry.

Vision- AUTOSAR aims to improve complexity management of integrated E/E architectures through increased reuse and exchangeability of SW modules between OEMs and suppliers. It provides various levels of exchangeability, namely, between suppliers and solutions, between vehicle platforms and between manufacturers' applications.[6]

AUTOSAR aims to standardize the software architecture of Electronic Control Units (ECUs). AUTOSAR paves the way for innovative electronic systems that further improve performance, safety and security.It also accelerates the development and maintenance and aim towards re-use of the software.AUTOSAR provides significant benefits to various types of industries like OEM's, suppliers and tool providers.

The Classic AUTOSAR Platform supports a layered architecture. Fig.3 shows the layered structure.The higher abstraction levels are Application Layer, Runtime Environment (RTE) and Basic Software (BSW) which runs on microcontroller.BSW is further divided into 6 layers- Services, ECU Abstraction, Microcontroller Abstraction and Complex Drivers.

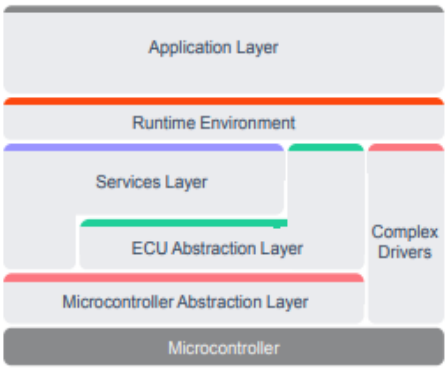

Figure. 3: AUTOSAR Layered Architecture[6]

Challenges with the Classic Platform-New types of in vehicle computers are required to fulfill the needs of performance, flexibility and connectivity. While doing so ensuring backwards compatibility with existing solutions and fulfillment of increasing requirements for safety and security are must as well. Hence, AUTOSAR Adaptive Platform is introduced. This is a combination of AUTOSAR Classic and Infotainment Platform. In comparison to the AUTOSAR Classic Platform the AUTOSAR Runtime Environment for the Adaptive Platform dynamically links services and clients during runtime.

2.3 Essence of Automotive Software Engineering

Realization of successful automotive system in terms of vehicle development, quality control, risk, cost management, safety, economic impact, energy consumption, comfort and high quality a coordinated and systematic development process focusing on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem.

Due to the increased complexity of products, globalization of marketplace, reduction of product development cycle, software as dominant force for change in almost all new products, worldwide deployment of new technology, increasing need for reuse of components, development in product generations, high safety criticality and large volumes, different countries with different regulations, different climatic conditions.

It is practiced through standards, norms and well defined processes and practices.

2.4 Functional safety (ISO26262)

Applicability for Road vehicles:

Derived from a more general functional safety standards like ISO 26262 builds on EC 61508, the general functional safety standard for electrical and electronic (E / E) systems and IEC 61508 which is designed for safety-related systems for series production passenger vehicles with a gross vehicle mass up to 3,500 kg and that are equipped with one or more E/E systems.

Definition: Absence of unreasonable risk due to the hazards caused by the malfunctioning behavior of electrical/electronic systems

Key components: Uses a system for steps to manage functional safety and regulate product development at the system, hardware, and software levels.

Automotive Safety Integrity Level (ASIL)

The ASIL is a key component for ISO 26262 compliance. ISO 26262 uses a system of steps to manage functional safety and regulate product development on a system, hardware, and software level. ASIL is determined at the beginning of development process. ASIL is used to analyze the risk associated with the functioning of the system. ASIL uses the following parameters - extent of harm to individuals that can occur in hazardous situation, probability of exposure regarding operational situations and ability to avoid a specified harm through timely reactions to determine the ASIL.

There are four ASILs identified by the standard: ASIL A, ASIL B, ASIL C, ASIL D. ASIL D dictates the highest integrity requirements on the product and ASIL A the lowest. Hazards that are identified as QM (Quality Managed) do not dictate any safety requirements.

2.5 SAHARA(Security-Aware Hazard and Risk Analysis Method)

SAHARA approach, like the ASIL is applied at initial development phase. This approach focuses on the sources of problems that could occur due to malfunction or foreseeable user misuse. This approach can be seen as a method by which the security of a system can be assessed equivalent to HARA attempt for safety.[7]

But the security issues are not addressed by HARA within the ISO 26262 standard. In parallel, the STRIDE threat model provides a way to methodically review system designs and highlight security design flaws, but does not support the categorization of security hazards or methodically formulate security requirements in such a way as to avoid identified risks.

SAHARA approach, combines security and safety analyses. Threats are quantified with reference to the ASIL analysis. Factors used to determine the SecL (Security Level) are resources (R) and know-how (K) that are required to pose the threat and the threats criticality (T). Like ASIL, SAHARA also identifies 4 levels: SecL 1, SecL 2, SecL 3 and SecL 4 with SecL 1 being the lowest. SecL 0 means that no threat is present.

2.6 Scenario Modeling

For Automated Driving, use cases specific to a general user story are created. A use case comprises of functional range, desired behaviour and functional system boundaries. These use cases are then refined to create scenes.A scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities. Only a scene representation in a simulated world can be allencompassing (objective scene, ground truth). In the real world it is incomplete, incorrect, un- certain, and from one or several observers' points of . And combining more than one scenes together forms a scenario. A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions and events as well as goals and values may be specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time[9].For scenario modeling TDSS and SCIL can be used which are discussed in detail in further subsections.

2.6.1 Test-Driven Scenario Specification

TDSS is based on the Scenario Modeling Language for Kotlin (SMLK), a Kotlin-based framework for modeling / programming the behavior of a reactive software component using loosely coupled threads of behavior, called scenarios. Scenarios can be executed as a scenario program. SMLK inherits the concepts of behavioral programming (BP), Live Sequence Charts (LSC) and the Scenario Modeling Language (SML). Scenarios can extend as well as constrain the behavior of other scenarios. This allows for a flexible modeling of behavior requirements close to how humans conceive and communicate them. SMLK also supports the modeling of test scenarios, which can be executed as JUnit tests, allowing us to model and execute specific scenario- /requirement interplays. In TDSS, we assume that we are developing a reactive software component and have a source of requirements (e.g., a document or human stakeholder) that specifies what events or data are the inputs and outputs of that components as well as what the desired relationship of these inputs and outputs is over time. Then, TDSS consists in a repeated process where we take a requirement, and first create a test scenario for it, which is expected to fail. We then add one or several scenarios to the specification in order to formalize the requirement and satisfy the test case. If the test fails, this might reveal a simple modeling problem, or an inconsistency in the requirements modeled thus far. If the test passes, the process is repeated, first with further tests for the same requirement, e.g., covering corner cases, and then the process is repeated for the next requirement. Finally, we obtain an executable requirements and test specification of the component to be developed. This process greatly reduces the risk of requirement ambiguities or inconsistencies to survive into later development stages. Moreover, the executable test and requirement specifications are useful in later development tasks, such as testing and failure interpretation. TDSS also allows us to consider fault tolerance concerns during the requirement specification and analysis, by carrying out fault injection tests, e.g.,

testing invalid input values. These tests are required by functional safety standards like ISO 26262[2].

TDSS combines formal scenario-based specification and analysis with TDD. TDD was originally designed for software unit testing, where tests are written and executed before writing the program code. In small iterations tests and code are extended and adapted following the red/green/refactor approach. The first step is writing a new test to validate a new functionality, which is expected to fail initially (red). In the second step the implementation must be extended or adapted until the tests pass (green). The third step (refactor) is cleaning up the code, without changing the functionality. The TDSS process is proposed as shown in Fig. 4.[2]

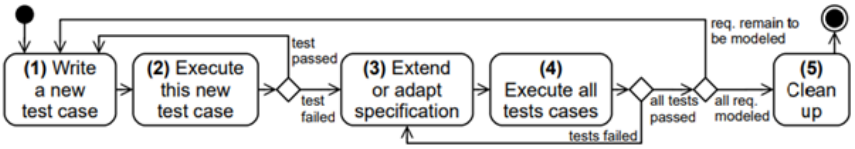

Figure. 4: Test-Driven Scenario Specification(TDSS) Process[2]

2.6.2 Scenario in Loop

A scenario is a specific possibility, that means a series of events that is projected to occur. “in-the-loop” concept seems to be the most efficient methodology to test the interaction of AVs and the control environment. [3]

In this the Scenario concerning the vehicles is also discussed, that mean Vehicle in the loop, which was directly developed to test ADAS (advanced driver assistance systems) functions. Vehicle in the Loop (ViL) represents the state when the vehicle is moving on the real surface or on a test bench, and the full environment is simulated, the actuators of the vehicle get their input signal directly from the simulated test environment. It can be considered as a Hardware-in-the-Loop approach. SciL is one step closer to a realistic environment simulation, because not only the vehicle and the surface are real, but some elements of traffic control can also be installed physically. This means that the vehicle has to use some of its own sensors, depending on the type of the test, to realize the traffic situation. SciL covers all the intermediate simulation states between ViL and real-traffic test approaches. For example, if the signal head or some other vehicles are also real, then these simulations are considered as SciL simulations. Apart from the test vehicle, the road surface, and the traffic simulator, the elements of SciL can either be real or simulated, depending on the current simulation configuration. The concept realizes a virtual twin or mixed reality technology, see Figure 5. [3]

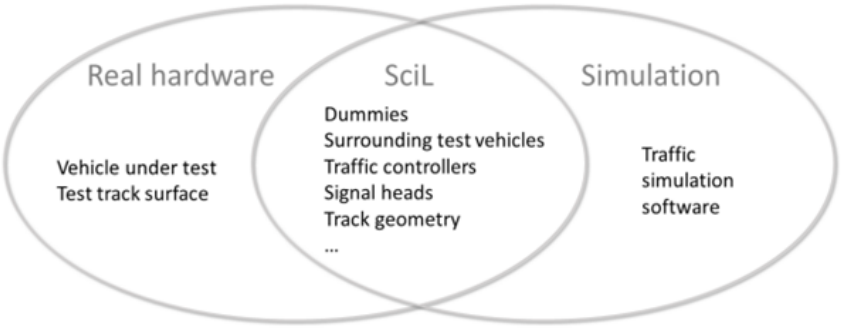

Figure. 5: The relation of real, simulated, and SciL elements. Possible SciL elements are in the middle.[3]

3 Project

In this project concepts of MBSE have been applied on the general user story “Autonomous City Drive”.

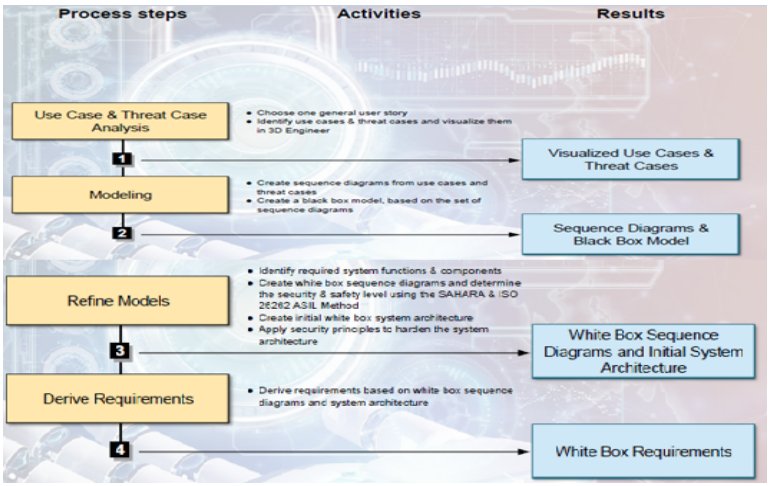

Figure. 6: Process steps involved for Model Based Requirement Derivation

3.1 Use case and threat case analysis

The first step is to identify use cases and threat cases and to visualize them in 3D Engineer Tool. Following use cases and threat cases were identified at the initial stage of the project:

1. Pedestrian crossing- This is the basic use case in which brake will be applied if camera detects a pedestrian crossing the road.
2. Hacker threat case- In this threat case, initially at a 4-way crossing traffic lights are working correctly. But then a hacker hacks the traffic light system such that both of the opposite lights turn green at the same time. To overcome this the cars should be able to detect both lights and if both are green, brakes should be applied.

Figure. 7: Hacker scenario visualization using 3D Engineer

Fig. 7,8 show the visualization of above cases in 3D Engineer.

3.2 Modeling

From the use cases and threat cases black box sequence diagrams are developed. Fig 8 illustrates the sequence diagram for pedestrian use case.

Figure. 8: Pedestrian scenario visualization and sequence diagram]

Fig 9 illustrates sequence diagram for hacker threat case

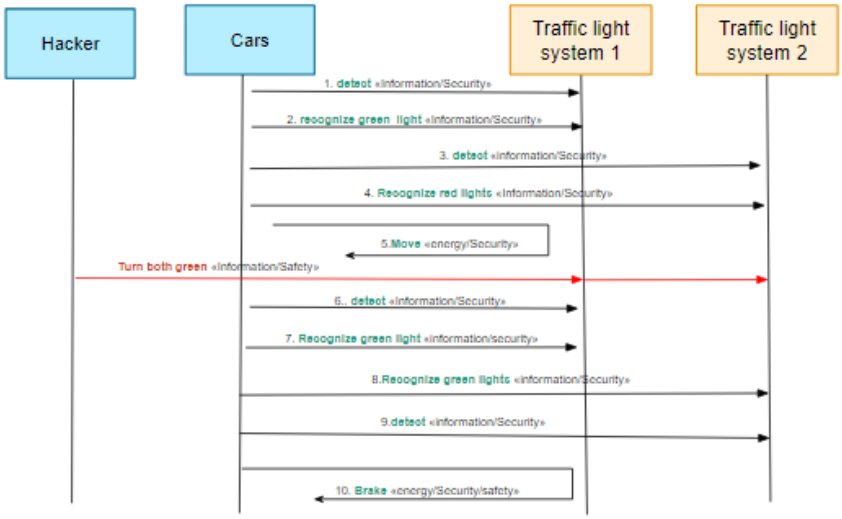

Figure. 9: Sequence diagram for hacker threat case

With the help of these use cases and threat cases an initial Application Environment model was developed.

The Environment model, in general, will tell us in what kind of situation, the system needs to be developed and how the interaction would take place with its surrounding environment. The environmental model for the Automotive system requires perceiving and understanding the Vehicle environment using sensors, providing self-localization, controlling the vehicle, and planning the routes.

Here, there are many aspects that will influence the Vehicle. Especially because it is a vehicle that interacts with living objects, all the aspects of the environment are carefully checked for the influencing variables. These influencing variables include weather conditions, terrain and road details, obstacles like pedestrians, other vehicles and infrastructure objects and they will act as major affecting criteria for the movement and destination path selection for the vehicle.

A pedestrian detector that locates walking front/crossing humans on a digital image is a key module for autonomous vehicles. The underlying computer vision algorithms are highly computationally demanding and require real-time response, giving information about speed and position of the humans that can appear with different poses, clothes, illuminations, and backgrounds.

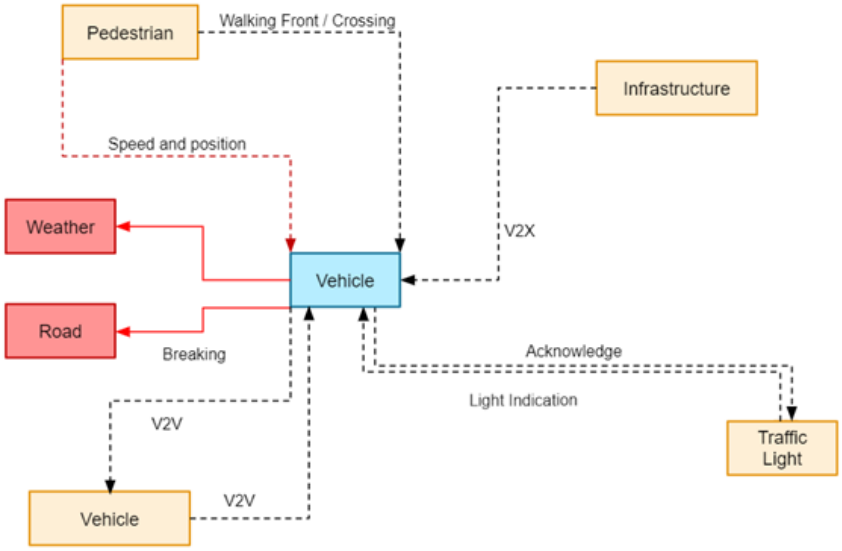

Figure. 10: Application Environment Model

Autonomous driving systems operate by using input images from a camera and one of the tasks these systems need to perform is the detection and understanding of traffic lights in a traffic environment, by localizing all relevant traffic lights in an image received from an on board camera mounted on the vehicle. These systems should also respond to non-living infrastructure, such as other vehicles, road signs, repair barriers, and other debris that could obstruct traffic. One of the tasks Automotive systems as well needs to perform is the detection and understanding of traffic lights in a traffic environment, by localizing all relevant traffic lights in an image received from an on board camera mounted on the vehicle.

3.3 Model Refining

After modeling the initial use cases, threat cases and application environment model, the next step is to refine the models to get a refined system. To extend the system further, the following threat cases were added:

1. Anti-Skid and Braking in case of Heavy Rain threat case- In case of heavy rain, camera would not be able to capture clear image, hence mid range and near range radar sensors will work together to detect obstacles. Based on obstacle distance either slow braking or immediate braking will take place and the ESP unit will be requested via the DASy, central gateway unit and vehicle control unit to prevent skidding.
2. Sticker on camera threat case- In case that someone puts a sticker on the camera, false detection issue could arise.To solve this issue along with camera,output of additional

sensors like vehicle motion and position sensors will be considered for decision making of autonomous drive.

After identification of these new threat cases, the next step is to identify system functions and components which would be required and create white box sequence diagrams for all the threat cases and use cases. After creating the white box sequence diagrams, security and safety levels are determined using the SAHARA and ISO 26262 ASIL determination explained in the previous sections.

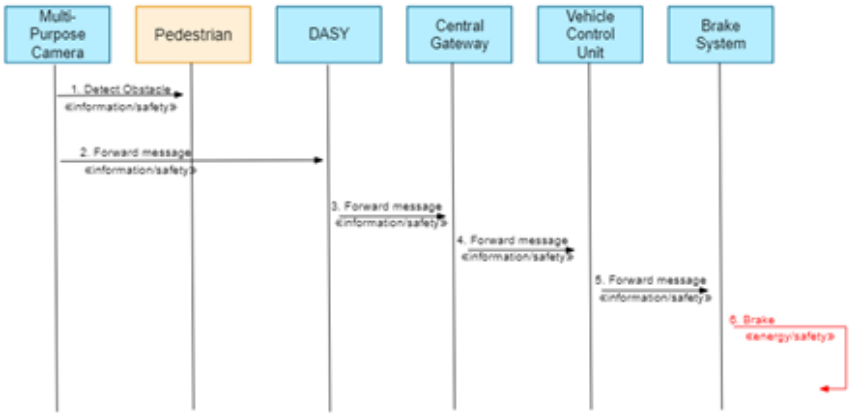

Figure. 11: White-box Sequence diagram for Pedestrian Crossingl

No.	Function	Stereotype	K	R	T/S	SecL		E	C	ASIL
1	Detect Obstacle	information/safety	0	0	0	0		4	1	2
2	Forward message	information/safety	0	0	0	0		-	-	-
3	Forward message	information/safety	0	0	0	0		-	-	-
4	Forward message	information/safety	0	0	0	0		-	-	-
5	Forward message	information/safety	0	0	0	0		-	-	-
6	Brake	energy/safety	0	0	0	0		-	-	2

Figure. 12: Pedestrian Crossing Function Table

Figure11 and 12 illustrate the white box sequence diagramand security and safety level determination for Pedestrian use case. After SAHARA and ASIL determination for pedestrian use case, only function.8 Brake is a threat function as ASIL for Brake is greater than 1.

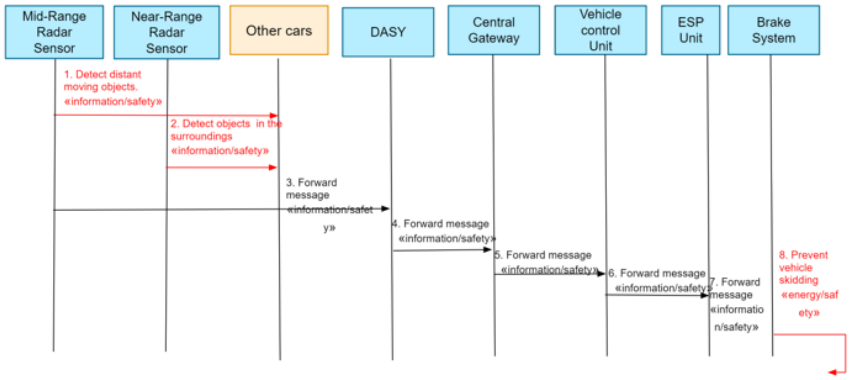

Figure. 13: Sequence diagram for Heavy rain threat case

No.	Function	Stereotype	K	R	T/S	SecL		E	C	ASIL
1	Detect distant moving objects	information/safety	1	1	3	3		2	3	2
2	Detect objects in the surroundings	information/safety	2	2	3	1		2	3	2
3	Forward message	information/safety	2	3	3	0		3	3	-
4	Forward message	information/safety	2	3	3	0		3	3	-
5	Forward message	information/safety	2	3	3	0		3	3	-
6	Forward message	information/safety	2	3	3	0		3	3	-
7	Forward message	information/safety	2	3	3	0		3	3	-
8	Prevent vehicle skidding	Energy	2	3	3	0		3	3	2

Figure. 14: Heavy Rain Function Table

Figure13 and 14 illustrate the white box sequence diagramand security and safety level determination forAnti-Skid and Braking in case of Heavy Rain threat case.. After SAHARA and ASIL determination for heavy rain threat case, function 1,2 and 8 are threat functions as either ASIL or SecL for them is greater than 1.

After analyzing the Anti-Skid and Braking in case of Heavy Rain threat case, there is a probability of error during detection of obstacle by mid-range radar sensors. Mid-range sensor works well in case there is no snow or rain affecting the radar input. But in case the weather has rain the Radar input/data may be compromised. This became the basis for the third threat case(RADAR threat case) which is a refinement of heavy rain threat case.A solution to avoid this error was found in the [8]. It discusses a new sensor called the Millimeter Radar Sensor. This readings from the Millimeter radar sensor coupled with the Mid-range Radar sensor is able to provide accurate reading even during rain or snow. In this case the millimeter RADAR sensor acts as a redundant sensor to the existing standard RADAR sensors, which is activated only during the case of rain/snow scenarios.

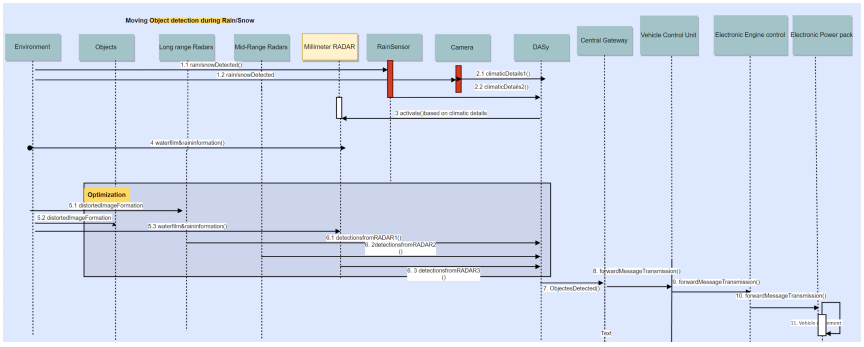

Figure. 15: Sequence diagram for RADAR threat case

No	Function	Stereo-type	K	R	T/S	SecL		E	C	ASIL
1	Climatic and Environmental detections by Rainsensor and camera	Material and information	0	0	1	1		4	3	2
2	Climatic and Environmental information detection with rainsensors and camera and forw ard the information to DASy	Information	0	0	3	4		3	2	2
3	Activation signal from DASy to the millimeter RADAR	Information	0	0	1	3		3	2	2
4	Water film size and other noise related information	Information and energy	0	0	1	1		1	1	0
5	Detections from all the environment sources by RADARs with environmental influence	Information	0	0	3	4		3	3	3
6	Information pass of RADAR detections to DASy for further processing and error reductions	Information	2	2	3	0		-	-	-
7	Forw arding detection information from DASy to central gatew ay	Information	2	3	3	0		-	-	-
8	Forw ard message	Information	2	3	3	0		-	-	-
9	Forw ard message	Information	2	3	3	0		-	-	-
10	Forw ard manipulatedmessage	Information	2	3	3	0		-	-	-
11	Autonomous Vehicle movement based on new etections	Energy	2	3	3	0		-	-	3

Figure. 16: RADAR Function Table

Figure15 and 16 illustrate the white box sequence diagramand security and safety level determination for RADAR threat case. After SAHARA and ASIL determination for RADAR threat case, function 1,2,3,5 and11 are threat functions as either ASIL or SecL for them is greater than 1.

Figure17 and 18 illustrate the white box sequence diagramand security and safety level determination for Camera threat case. After SAHARA and ASIL determination for sticker on camera threat case, function 1 and 9 are threat functions as either ASIL or SecL for them is greater than 1.

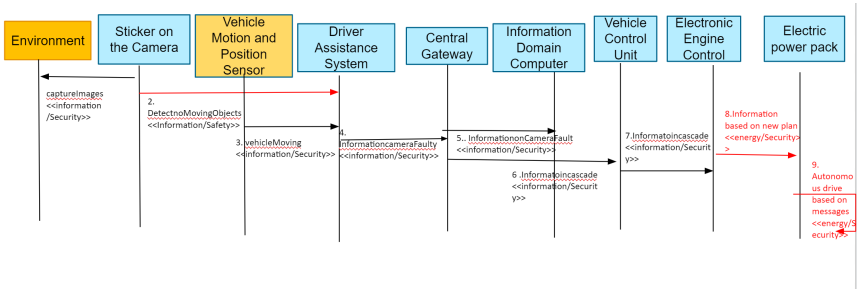

Figure. 17: Sticker on Camera threat case

No.	Function	Stereotype	K	R	T/S	SecL	E	C	ASIL
1	Scenario sticker stuck on the camera to depict faulty environment	Malicious external activity	1	1	3	3	4	1	2
2	Image captured dynamic behaviour shows no moving objects	information/safety	2	3	3	1	1	3	0
3	Vehicle status from motion and position sensor	information/safety	2	3	3	1	1	3	0
4	DASy decision on camera behaviour and status	information/safety	2	3	0	1	-	-	-
5	Camerastatus to the Information domain	information/safety	2	3	0	0	-	-	-
6	Information on object detection	information/safety	2	3	0	0	-	-	-
7	Forward information / No objects nearby	information/safety	0	3	0	0	-	-	-
8	Information based on new messages	information/safety	2	3	3	0	-	-	-
9	Autonomous driving based on new messages	energy/safety	2	3	3	0	-	-	2

Figure. 18: Sticker on Camera Function Table

The next step is to create a system architecture. A system architecture is the conceptual model that defines the structure, behavior, and more views of a system.The purpose of the System Architectural Design Process is to establish a system architectural design and identify which system requirements are to be allocated to which elements of the system, and to evaluate the system architectural design against defined criteria.[1]

As a result of successful implementation of this process:

1. A system architectural design is defined that identifies the elements of the system;
2. The system requirements are allocated to the elements of the system;
3. The interfaces of each system element are defined;
4. The dynamic behavior objectives of the system elements are defined;
5. Consistency and bidirectional traceability are established between system requirements and system architectural design;
6. The system architectural design is agreed and communicated to all affected parties. [1]

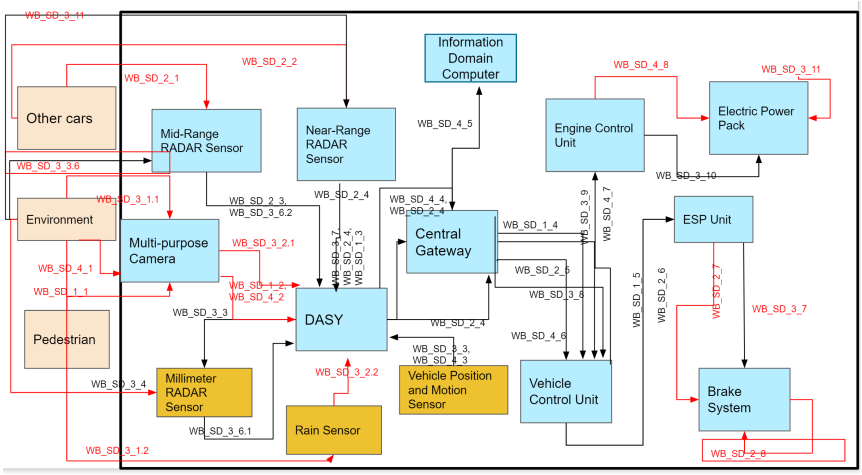

Figure. 19: System Architecture

3.4 Model-based Requirement Analysis

ID	Description	SAHARA	ISO 26262	Derived from			
		Security Level (0-4)	Safety Level (0-4)	Requirements	System Architecture	Use Cases + Sequence Diagram	Threat Cases + Sequence Diagram
R01	DASy sends notification after mid-range sensor detects an object	3	-	-	WBM01	-	TC01
R02	central gateway forwards notification for mid range sensor	2	1	-	WBM01	-	TC01
R03	Vehicle slows down after mid-range camera detects an object	1	2	-	WBM01	-	TC01
R04	DASy sends notification after near-range sensor detects an object	-	-	-	WBM01	-	TC01
R05	central gateway forwards notification for near range sensor	-	-	-	WBM01	-	TC01
R06	Vehicle applies instant brake after near-range camera detects an ol	1	2	-	WBM01	-	TC01
R07	If camera detects heavy rain then it informs the ESP unit via DASy unit ,Central Gateway unit and vehicle control unit	1	1	-	WBM01	-	TC01
R08	When Esp unit receives the signal from Camera Unit it manipulates the speed to prevent skidding	2	2	-	WBM01	-	TC01
R09	If camera does not detect moving objects but motion sensor detect movement of vehicle then it informs the Inforamtion Domain Computer that there is a fault in the camera via DASy, Central Gateway unit and Vehicle control unit	3	3	-	WBM01	-	TC03
R10	If camera detects moving objects and motion sensor detects movement of vehicle then it informs the Inforamtion Domain Computer that Camera is working fine via DASy, Central Gateway unit and Vehicle control unit	3	3	-	WBM01	-	TC03
R11	If camera detects rain or snow, it informs DASy which then requests the data from both Mid_Range and Millimeter Radar Sensors and sends this data to the vehicle control unit via Central gateway	2	2	-	WBM01	-	TC02
R12	If the Camera Detects that the weather is clear, it informs DASy which then requests for Mid_range sensor value and sends this value to Vehicle control unit via Central gateway	2	2	-	WBM01	-	TC02
R13	The vehicle must be able to detect an object	3	-	-	WBM01	-	TC01
R14	The vehicle must be able to detect wet terrain	3	2	-	WBM01	-	TC01
R15	The vehicle must be able to break in case of accident	4	4	-	WBM01	-	TC01
R16	The vehicle must have the ability to calculate distances	4	4	-	WBM01	-	TC01
R17	The decision message must be forwarded to the Central Gateway.	0	2	R04	WBM01	-	TC01
R18	The vehicle must be able to Detect distant moving objects	3	2	-	WBM01	-	TC02
R19	The camera sensor should be able to differentiate moving objects in the frames	3	-	-	WBM01	-	TC02
R20	The camera must be able to report anomalies to the connected modules in case of stagnant or uncategorised images	1	2	R07	WBM01	-	TC02
R21	The information should be cascaded to the DASy	0	2	R08	WBM01	-	TC02
R22	The informatin should be authenticated by the DASy and validate the image	0	2	R09	WBM01	-	TC02
R23	The information from the DASy should be cascaded to the Gateway module and then to the Engine control Unit	0	2	R10	WBM01	-	TC02
R24	The vehicle must be able to Detect distant and near range moving objects with the help of mid-range and near-range RADAR sensors.	3	2	-	WBM01	-	TC02
R25	The system must be able to detect tampering with the camera.	3	2	-	WBM01	-	TC03

Figure. 20: Requirement for Autonomous City Drive

After creating the sequence diagrams for use and threat cases and the system architecture, requirements can be derived based on them. The tablein figure.20 shows the requirements

derived for the Autonomous City Drive user story. The table consists of the description of the requirement, its SAHARA security level and ASIL safety level and from what the requirement is derived from.

3.5 Test Driven Scenario Specification

Once the white box requirements for the system are derived, the TDSS process discussed in section. was followed to create scenario specification. Figure.21 shows the test cases which were executed for the Autonomous City Drivebased on the requirements.

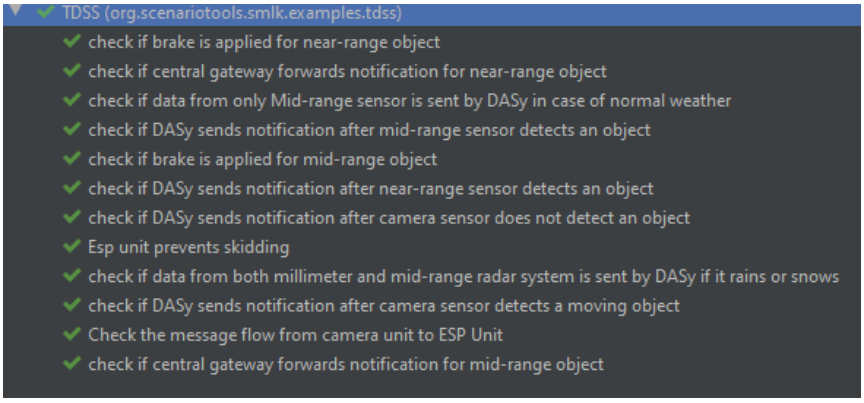

Figure. 21: Test cases for Autonomous City Drive

After writing the test cases, scenarios were added to Scenario Specification, so that the requirements are fulfilled and hence in the end all the test cases pass. In the end following scenarios were added to the Scenario Specification.

1. Braking- When mid-range sensor detects an object, it sends this notification to the DASy. Upon receiving this data, DASy forwards this message to Vehicle Control Unit via the Central Gateway, hence requesting the Brake System to apply a slow brake.

Figure. 22: Braking scenario code

When near-range sensor detects an object, it sends this notification to the DASy. Upon receiving this data, DASy forwards this message to Vehicle Control Unit via the

Central Gateway, hence requesting the Brake System to apply a firm brake. Figure22. Shows the code snippet for this scenario.

2. Anti-skid Braking- If camera Detects Heavy rain then it requests the ESP unit to prevent skidding via the DASy, central gateway unit and vehicle control unit. Figure.23 Shows the code snippet for this scenario.

```
// Scenario for Anti-Skid Braking System Threat case
// If camera Detects Heavy rain then it informs the Esp unit via the DASy, central gateway unit and vehicle control unit
scenario(cameraUnit.detectHeavyRain( request: "Heavy Rain")){ this: Scenario
    request(dasyUnit.sendNotification( request: "Heavy Rain"))
    request(centralGatewayUnit.forwardNotification( request: "Heavy Rain"))
},
scenario(vehiclecontrolunit.forwardgatewaymessage( request: "Heavy Rain")){ this: Scenario
    request(espUnit.ManipulateSpeed( request: "Prevent Skidding"))
},
```

Figure. 23: Code for Anti-skidding scenario

3. RADAR Switching- The first scenario is when the Multi-purpose camera detects rain or snow, it sends this notification to DASy unit. Upon receiving this data, the DASy unit should request for the Millimeter sensor to be activated. Once activated the DASy unit requests for the data from both the Millimeter sensor and Mid-range sensor. It then computes the data and send the data from both the sensors to the Vehicle control unit via the Central Gateway unit.
In the case of the Second scenario the weather is clear and there is no rain or snow. Once the DASy unit receives the information from the Multipurpose-camera unit, it doesn't activate the Millimeter Radar sensor and just requests data from the Midrange radar sensor and forwards it to the Vehicle control unit via the Central gateway unit. Figure.24 Shows the code snippet for these scenarios.

```
//Scenarios for Radar Threat case
//In Case Rain or snow is detected by the Camera then the DASy system requestes for the data from both the Radar Sensor and
scenario (cameraUnit.detectRainSnow_Radar( request: "Rain or snow detected")) {...},
scenario(dasyUnit.recieveRadarNotification( request: "Rain or snow detected")) {...},
scenario(dasyUnit.recieveRadarNotification( request: "Rain or snow detected")){ this: Scenario
    request(dasyUnit.sendRadarData( request: "Radar reading"))
    request(dasyUnit.sendmillimeterRadarData( request: "Millimeter Radar Reading"))
    request(centralGatewayUnit.forwardRadarData( request: "Radar reading"))
    request(centralGatewayUnit.forwardmilimeterRadarData( request: "Millimeter Radar Reading"))
},
//In case the camera detects clear weather then only the mid range radar data is used and forwarded to vehicle control unit
scenario(cameraUnit.detectRainSnow_Radar( request: "No rain or snow. Clear weather")) {...},
scenario(dasyUnit.recieveRadarNotification( request: "No rain or snow. Clear weather")) {...},
scenario(dasyUnit.recieveRadarNotification( request: "No rain or snow. Clear weather")) { this: Scenario
    request(dasyUnit.sendRadarData( request: "Only mid range radar reading"))
    request(centralGatewayUnit.forwardRadarData( request: "Only mid range radar reading"))
}
```

Figure. 24: Code for RADAR switching scenario

4. Camera Fault Detection- If camera does not detect moving objects but motion sensor detect movement of vehicle then it informs the Inforamtion Domain Computer that there is a fault in the camera via DASy, Central Gateway unit and Vehicle control unit.
If camera detects moving objects and motion sensor detects movement of vehicle then it informs the Inforamtion Domain Computer that Camera is working fine via DASy, Central Gateway unit and Vehicle control unit. Figure.25 Shows the code snippet for this scenario.

```
// Scenario for Camera Fault Detection Threat case
// detecting non moving objects by camera
scenario(cameraUnit.cameraSensing( request: "Detecting non moving Objects")) { this: Scenario
    request(dasyUnit.forwardCameraNotification( request: "Detecting non moving Objects"))
},
// detecting moving objects by camera
scenario(cameraUnit.cameraSensing( request: "Detecting moving Objects")) {...},

// detecting vehicle movement by the vehicle motion sensor
scenario(vehicleMotionandPositionSensor.vehicleMotionSensing( request: "Vehicle is moving")) {...},
// detecting vehicle not moving by the vehicle motion sensor
scenario(vehicleMotionandPositionSensor.vehicleMotionSensing( request: "Vehicle is not moving")) {...},
scenario(dasyUnit.forwardCameraNotification( request: "Detecting non moving Objects")){...},
scenario(dasyUnit.forwardCameraNotification( request: "Detecting moving Objects")){...},
scenario(centralGatewayUnit.forwardCameraNotification( request: "faulty camera")){ this: Scenario
    request(informationDomainComputer.computerMessage( request: "Please check camera"))
},
scenario(centralGatewayUnit.forwardCameraNotification( request: "camera is good")){ this: Scenario
    request(informationDomainComputer.computerMessage( request: "camera health is good"))
},
```

Figure. 25: Code for Camera Fault Detection Scenario

3.6 Scenario In Loop

For Scenario in loop testing, feature files were created based on the requirements. The different scenarios were then tested using JUnit Testing in loop. Figure.26 shows the different scenarios test in loop for the 5 features.

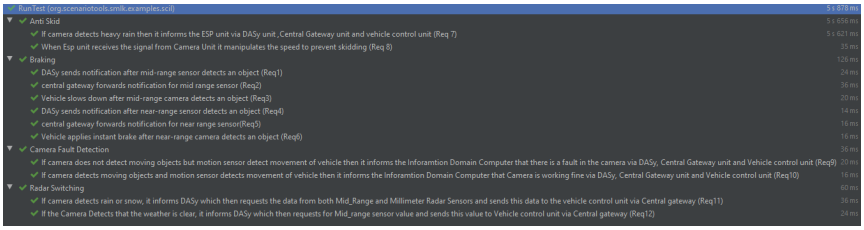

Figure. 26: Scenario in Loop Test Results

4 Conclusion

Following are the learnings from this project

- Different processes involved in automotive software development
- Set specifications for the development of a project using TDSS
- Modeling of use cases using 3D-Engineering tool
- Applying functional safety standards like ISO26262-ASIL and security level approach SAHARA.

- Derivation of requirements from use cases
- SCIL supported modeling

5 References

1. Automotive SPICE Process Assessment / Reference Model Author(s): VDA QMC Working Group 13 / Automotive SIG Version: 3.0 Date: 2015-07-16
2. 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)
3. 2019, The Scenario-In-Loop(SciL) automotive simulation concept and its realisation principles for traffic control
4. wiki - https://en.wikipedia.org/wiki/Self-driving-car
5. www.telemotive.de
6. www.autosar.org
7. https://past.date-conference.com/proceedings-archive/2015/pdf/0622.pdf
8. https://tuprints.ulb.tu-darmstadt.de/epda/000765/PhdThesis.pdf
9. Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving