# Autonomous Vehicle Scenario Modeling System Design Document

**Version 0.9**

**11/21/2024**

```
             ____------------  _____
\~~~~~~~~~~~/~_--~~~-----~~~~~       \
 `---`\  _-~       /                    \
    _-~ <_          /                      \[]
  / ___     ~~--[""] |      ________-------'_
> /~` \     |-.   `\~~.~~~~~                _ ~ - _
 ~|   ||\%   |       |    ~  ._                  ~ _   ~ ._
    `_//|_%  \      |          ~   .              ~-_   /\
           `--__     |    _-____  /\               ~-_ \/.
                ~--_ /  ,/ -~-_ \ \/           _______---~/
                    ~~-/._<   \ \`~~~~~~~~~~~~~~     ##--~/
                          \    ) |`------##---~~~~-~   ) )
                           ~-_/_/                   ~~ ~~
```

# Document Control

## Distribution List

The following list of people will receive a copy of this document every time a new version of this document becomes available:

Teaching assistants:

Clay Pate

Customer(s):

M. I. Akbas

AVVC

Project team members:

Isabella Acosta

Serena Conticello

Hannah Ramsden

William Reimer

Davian Rosario-Ortiz

## Change Summary

The following table details changes made between versions of this document:

| Version | Date | Modifier | Description |
|---|---|---|---|
| 0.1 | Oct-31-24 | All team members | Creation of the document, initialization of the document and addition of rudimentary information. |
| 0.2 | 11-3-24 | All team members | Initial information was added to the document. |
| 0.3 | 11-4-24 | All team members | Initial information was added to the document. |
| 0.4 | 11-5-24 | All team members | Split work by section and assigned sections to each team member. |
| 0.5 | 11-14-24 | Isabella | Worked on Section 1 of the document. |
| 0.6 | 11-18-24 | William | Worked on Section 3 of the document. |
| 0.7 | 11-19-24 | All team members | Continued working on sections 4-6 of the document. |
| 0.8 | 11-20-24 | William and Isabella | Worked on Sections 1, 2, and 3 of the document. |
| 0.9 | 11-21-24 | All team members | Making final edits to document. |

# Table of Contents

# 1. Introduction

## 1.1. Purpose and Scope

The purpose of the SDD is to provide the reader with a base outline of the functionality, technical details, and overall framework of a product. It communicates how the system is going to perform and the goals and objectives the product should complete. Our SSD goes over our simulation platform and our project will be testing using PolyVerif. The purpose of the PolyVerif system is to provide a simulation platform that allows for the testing of autonomous vehicles by way of verification and validation. The system can be used to create various types of simulations, whether they be complex or simple, that can be tested in multiple environments tailored to the results that should be produced. It allows for the simulation parameters, environments, and traffic conditions to be customized, providing accuracy and efficiency between the different environments tested. The objectives and goals of this system are to provide validation and verification to these simulations which can help identify potential issues within the simulation before they are presented in real–world scenarios.

## 1.2. Project Executive Summary

The main goal of the project is to expand the application space and examples within PolyVerif and integrate PolyVerif with machine learning validation tools. The tool needs to be learned and find potential issues and fix them and how it can be combined with tools in the lab to provide more complex research cases. Our future uses will involve machine learning, like neuroevolution, once the tool is stably integrated.

### 1.2.1. System Overview

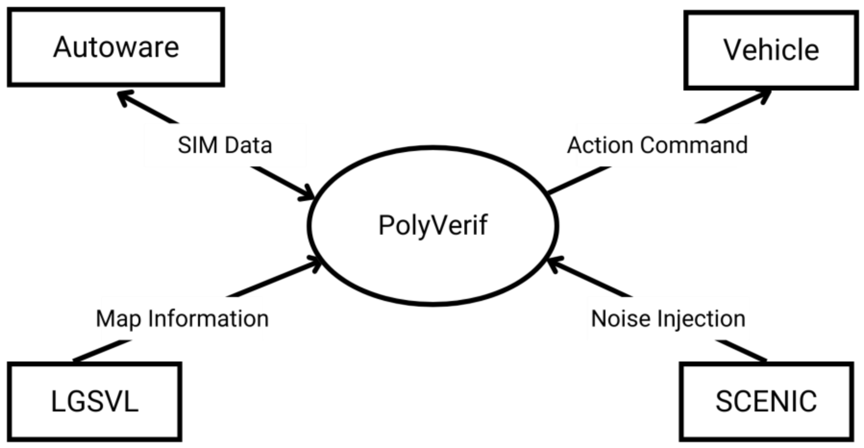


Figure 1. System context diagram

### 1.2.2. Design Constraints

The CPU that is being used to run PolyVerif must have an 8-minimum core. The operating system is running Ubuntu 20.04 64-bit and Python must be running on version 3.8. The GPU must be running NVIDIA GTX 1080 (8 GB or higher). The computer that is being used to run the simulation is in the Micaplex in the MP 224 WiDe Lab. It is required to have ID badge access to enter the Micaplex as well as ID badge access to enter the MP 224 WiDe Lab. Login credentials are also needed to login into the computer that the simulation is being run on. The computer must be connected to the Wi-Fi network and have all the required programs and their versions (listed above) on it to run a successful simulation.

### 1.2.3. Future Contingencies

The only future contingency that we are aware of is that the PolyVerif simulation platform may change to another platform next semester. If this is to happen, we will work with our product owner on how to navigate this change and how it will affect our project.

## 1.3. Document Organization

This document covers the Introduction of the Project, the System Architecture in this project, the Human Interface Machine, the Detailed Design of the project, the External Interfaces involved within the project, and the System Integrity Controls.

## 1.4. Project References

[1] R. Razdan, Mustafa İlhan Akbaş, R. Sell, M. Bellone, Mahesh Menase, and Mohsen Malayjerdi, “PolyVerif: An Open-Source Environment for Autonomous Vehicle Validation and Verification Research Acceleration,” *IEEE Access*, vol. 11, pp. 28343–28354, Jan. 2023, doi: https://doi.org/10.1109/access.2023.3258681.

[2] M. I. Akbaş, M. Menase, S. Verma, and R. Razdan, “PolyFlows: Modular Test Framework Design for Autonomous Vehicles,” *2024 IEEE International Conference on Mobility, Operations, Services and Technologies (MOST)*, pp. 50–59, May 2024, doi: https://doi.org/10.1109/most60774.2024.00014.

[3] M. I. Akbas, “Testing and Validation Framework for Autonomous Aerial Vehicles,” *Journal of Aviation/Aerospace Education & Research*, Jan. 2021, doi: https://doi.org/10.15394/jaaer.2021.1849.

[4] Q. Goss, W. C. Pate, and Mustafa İlhan Akbaş, “An Integrated Scenario-Based Testing and Explanation Framework for Autonomous Vehicles,” May 2024, doi: https://doi.org/10.1109/most60774.2024.00015.

[5] D. J. Fremont *et al.*, “Scenic: a language for scenario specification and data generation,” *Machine Learning*, vol. 112, Feb. 2022, doi: https://doi.org/10.1007/s10994-021-06120-5.

[6] “Eclipse SUMO - Simulation of Urban MObility,” *Eclipse SUMO - Simulation of Urban MObility*. https://eclipse.dev/sumo/

[7] Geeks, G. for. (2024, June 19). *Verification and validation in software engineering*. GeeksforGeeks. https://www.geeksforgeeks.org/software-engineering-verification-and-validation/

[8] TWI. (2024). *What is an autonomous vehicle?* https://www.twi-global.com/technical-knowledge/faqs/what-is-an-autonomous-vehicle

[9] *Autonomous Vehicle Verification Consortium*. (n.d.). https://www.avvc.net/

## 1.5. Definitions, Acronyms, and Abbreviations

### 1.5.1. Definitions

This section lists terms used in this document and their associated definitions.

**Table 1: System Definitions**

| Term | Definition |
|---|---|

| | |
|---|---|
| PolyVerif | An open-source validation and verification simulation framework for autonomous vehicles. |
| Autonomous Vehicle Testing | An autonomous vehicle utilizes a fully automated driving system to allow the vehicle to respond to external conditions that a human driver would manage. We are currently using PolyVerif to test these autonomous vehicle scenarios. |
| Verification | The process of checking that software achieves its goal without any bugs. |
| Validation | The process of checking whether the software product is up to the mark or in other words the product has high-level requirements. |
| Scenic | A language for scenario specification and data generation, focusing on modeling complex environments for the use of systems such as robotic and autonomous. |

### 1.5.2. Acronyms

This section lists the acronyms used in this document and their associated definitions.

**Table 2: Acronyms**

| Term | Definition |
|---|---|
| AVVC | Autonomous Vehicle Verification Consortium |
| SUMO | Simulation of Urban Mobility |

### 1.5.3. Abbreviations

This is not applicable to our project.

**Table 3: Abbreviations**

| Term | Definition |
|---|---|
| n/a | n/a |

## 2. System Architecture

*<< In this section, describe the system and/or subsystem(s) architecture for the project. This is a high-level description of the system itself, and therefore references to external entities should be minimal, as they will be described in detail in Section 5, External Interfaces. >>*

### 2.1. System Hardware Architecture

This is not applicable to our project.

### 2.2. System Software Architecture

*<< In this section, describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), computer languages, and programming computer-aided software engineering tools (with a brief description of the function of each item). Use structured organization diagrams/object-oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If appropriate, use subsections to address each module. >>*

*Note: The 3D environments and test scenario(s) provided by this product are both dependent on the currently existing PolyVerif architecture and expand upon the functionality of PolyVerif as a whole. Because of this, existing PolyVerif software and organization will be included in this section and treated as a part of the product.

To analyze the PolyVerif software architecture, an insightful first step would be to look PolyVerif's directory structure, which when successfully installed, should look like the following:

- adehome
    - AutowareAuto
    - lgsvl_msgs
    - ros2-lgsvl-bridge
    - OSSDC-SIM-v1_1-Linux
    - PythonAPI
    - Scenic
    - Avp_demo
    - Node
        - Node_Perception_Validation
        - Node_Control_Validation
        - Node_Localization_Validation
        - Node_Path_Planner_Validation

- Test_Cases
- PolyReports
- Poly_Suite

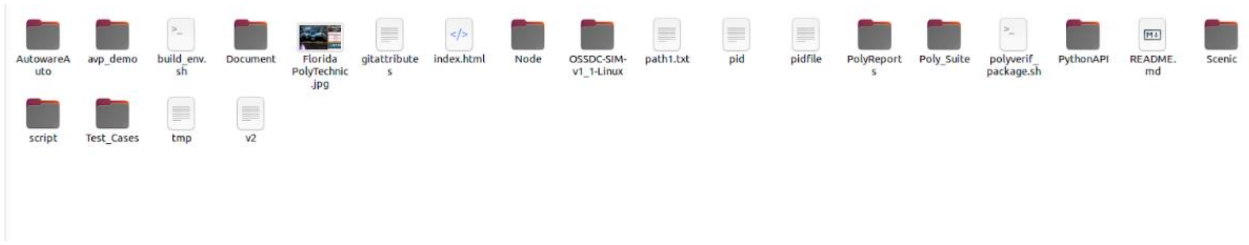


Figure 2. File Explorer View of System

PolyVerif is reliant on Python, both for functionality, and for additional API's that are discussed in Section 5. C++ is also used, along with custom languages found in Scenic, which will also be later discussed. Both Python and JavaScript contribute to making the various GUI's for displaying reports and variable information, as well as contributing to displaying the Scenic environment.

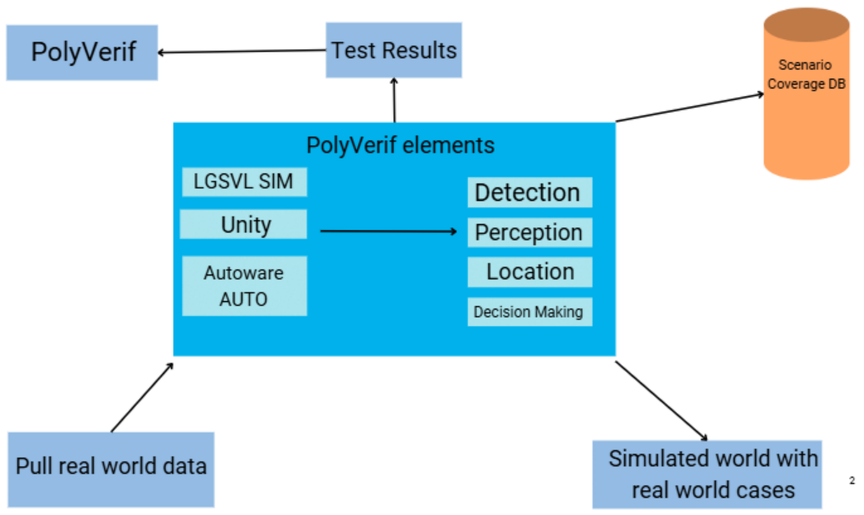


Figure 3. Software Architecture

### 2.2.1. Scenic

The Scenic directory is responsible for the creation, management, and control of simulated objects, models, obstacles, etc. It is important to note that Scenic plays a large role in the creation of the simulated environment and objects in both the product, and other previously made products stored in the Test_Cases folder. The selected environment is then sent to a simulator, which creates a 3D virtual environment reflecting the test case.

Scenic uses a programming language like Python in structure, and this language is how the user can change the object variables in both our product and others.

Additionally, the directories: lgsvl_msgs, and ros2-lgsvl-bridge load and run the simulation. These directories are responsible for the functionality of the simulator itself, whereas Scenic creates the simulations.

### 2.2.2. PythonAPI

Used for modifications of the simulated environment and objects based on the scenario(s) selected. Additional Python code can be used to change the behaviors present in the simulation to match certain conditions to fulfill criteria in the test scenario; behaviors that Scenic will be unable to handle alone.

The specifics of these additional Python files will be expanded upon in future updates, once the scenario(s) are better defined

More information on the PythonAPI will be covered in Section 5, as its functionality reaches beyond the scope of the product and is more relevant to the functionality of PolyVerif as a whole.

### 2.2.3. AutowareAuto

This directory is responsible for the mechanics behind the simulated vehicles being autonomous. Some examples of functionality within the scope of this product include the autonomous vehicles:

- Detecting other objects near the vehicle (including cars, buildings, etc.)
- Setting the path for the autonomous vehicle to follow
- Determining distance between objects and the vehicle
- Stopping the vehicle if an object gets too close

### 2.2.4. Poly_Suite

*Note: There are several other directories that will be included in this section

- Node
  - Node_Perception_Validation
  - Node_Control_Validation
  - Node_Localization_Validation
  - Node_Path_Planner_Validation
- Test_Cases
- PolyReports

These directories all contribute to the organization and user experience of this priduct. The Node and corresponding sub directories all allow the user to access the different validation functions, the Test_Cases directory ensures the user that PolyVerif is functioning properly, and PolyReports allows the simulation reports to be generated in a readable format. Poly_Suite also contributes to GUI functionality, specifically by packaging PolyVerif functions in a way that the user can simply click buttons and boxes to access the corresponding functions. This is demonstrated in Section 3.

## 2.3. Internal Communications Architecture

*<< In this section, describe the overall communications within the system, for example: LANs, buses, etc. Include the communications architecture(s) being implemented, such as X.25, Token Ring, etc. Provide a diagram depicting the communications path(s) between the system and subsystem modules. If appropriate, use subsections to address each architecture being employed. >>*

### 2.3.1. System Wide Communication

The PolyVerif framework relies on a modular structure, integrating tools like Scenic for scenario generation, AutowareAuto for AV control software, and a simulator for environment modeling, as mentioned in the previous sections. These tools communicate using Python APIs, ensuring interoperability between components.

Simulated sensor outputs (e.g., LiDAR, cameras) are streamed to AutowareAuto for processing. AutowareAuto processes simulated data (e.g., sensor feeds) from the simulator. Its role is to replicate the control, perception, and planning functionalities of an autonomous system, enabling PolyVerif to assess system decisions in real-time.

### 2.3.2. Scenario Testing

PolyVerif can use APIs to introduce real-world complexities such as noise in sensor data, GPS signals, or perception systems. These inputs test AV robustness and help researchers validate performance under imperfect conditions. The use of these phenomenon is dependent on the needs of the user and can be enabled or disabled at will.

Additionally, outputs from digital sensors present on the simulated vehicle are stored for later analysis. Reports are generated to evaluate metrics like detection precision, misclassification rates, and false negatives, but again, vary depending on the needs of the user.

## 3. Human-Machine Interface

*<< This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Any additional information may be added to this section and may be organized according to whatever structure best presents the operator input and output design. Depending on the nature of the project, it may be appropriate to repeat these sections at both the subsystem and design module levels. Additional information may be added to the subsections if the suggested lists are inadequate to describe the project inputs and outputs. >>*

### 3.1. Inputs

#### 3.1.1. Validation Suite

(Sources:

Upon launch of PolyVerif, the user is directed to a blank pane (except for the options on the left third side of the pane) titled “Validation Suite”. In this pane, the user can select 5 different options by clicking on a box to the left of each option. These 5 options are the following:

- Detection Validation
- Control Validation
- Localization Validation
- Mission Planning Validation

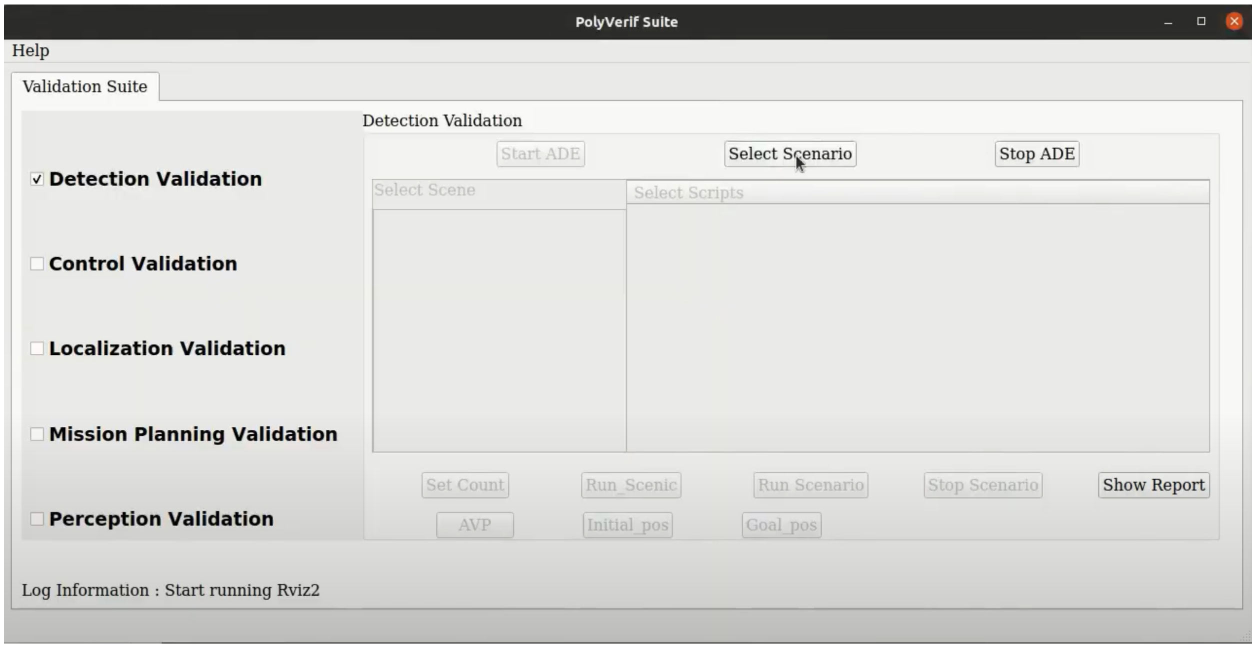
Once the user selects an option, the rest of the pane is filled with options and selections related to the option selected. For example, if the user selects the *Detection Validation* option, the middle and right-hand side of the pane is filled with options relating to the Detection Validation functions as well as selecting and running a simulation.

After selecting an option, the user must press the S*tart ADE* button. This allows the user to access scene selection and enables most of the functionality of this software. From here, the use

When the user selects any of these options, the user is presented with the same list of scenarios in the *Select Scenario* pane; *BorregasAvenue, CubeTown, SanFrancisco, and JTA_R2*. These options are responsible for setting the simulation environment (how the surroundings and detail looks). The options in the *Select Scripts* pane vary from each other slightly and will be explained in further detail in the next sections.

#### 3.1.2. Detection Validation

Located in the validation suite, this input is applicable when the user selects the *Detection Validation* option. When selected, multiple options are generated (and can be selected via buttons) in the middle and right-hand side of the pane. However, only three buttons can be selected: *Select Scenario*, *Stop ADE*, and *Show Report*. This is because most of the other options only become applicable when a simulation is selected, running, or has finished running. The only useful option that the user can select at this time is Select Scenario. The user could select *Show Report;* however, it would show a report with no variables, displaying N/A for calculated results. Additionally, the user could also click *Stop ADE*, but nothing will happen at this time, as a simulation is currently not running.



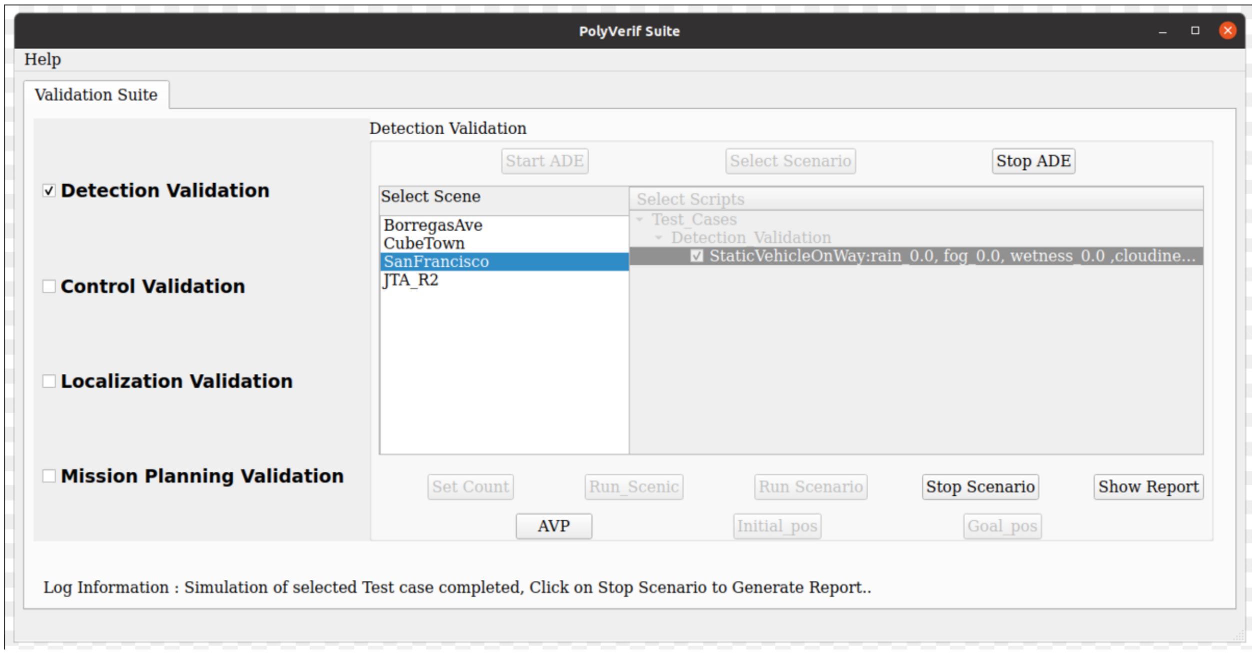
Log Information : Start running Rviz2

Figure 4.PolyVerif Validation Suite – Detection Validation

When the user does click the *Select Scenario* button, several different options in both the Select Scene, and Select Scripts panes appear. For Select Scene, the default options are as previously mentioned in Section 3.1.1. For Select Scripts, the default option is *Test Cases,* with several suboptions available to determine the scenario being run (setting the conditions and variables to be tested).

When a scene and script are selected, this enables the *Run Scenario* button. When clicked, the simulation begins and once loaded, immediately runs. Data collection begins automatically.

After the simulation ends, the user can click on the Show Report button to generate a report highlighting the values of the tested variables (details on report specifics will be explained in the Output section). It is possible for the user to click on the *Show Report* button; however, this will lead to incorrect variable results, N/A as a result, or a combination of the two. It is the responsibility of the user to click the *Show Report* button only after the simulation ends to ensure correct results (note, no technical issues will arise due to this form of user error, only incorrect end values).



Log Information : Simulation of selected Test case completed, Click on Stop Scenario to Generate Report..

Figure 5. PolyVerif Validation Suite – Detection Validation, Scenario Selection

### 3.1.3. Control Validation

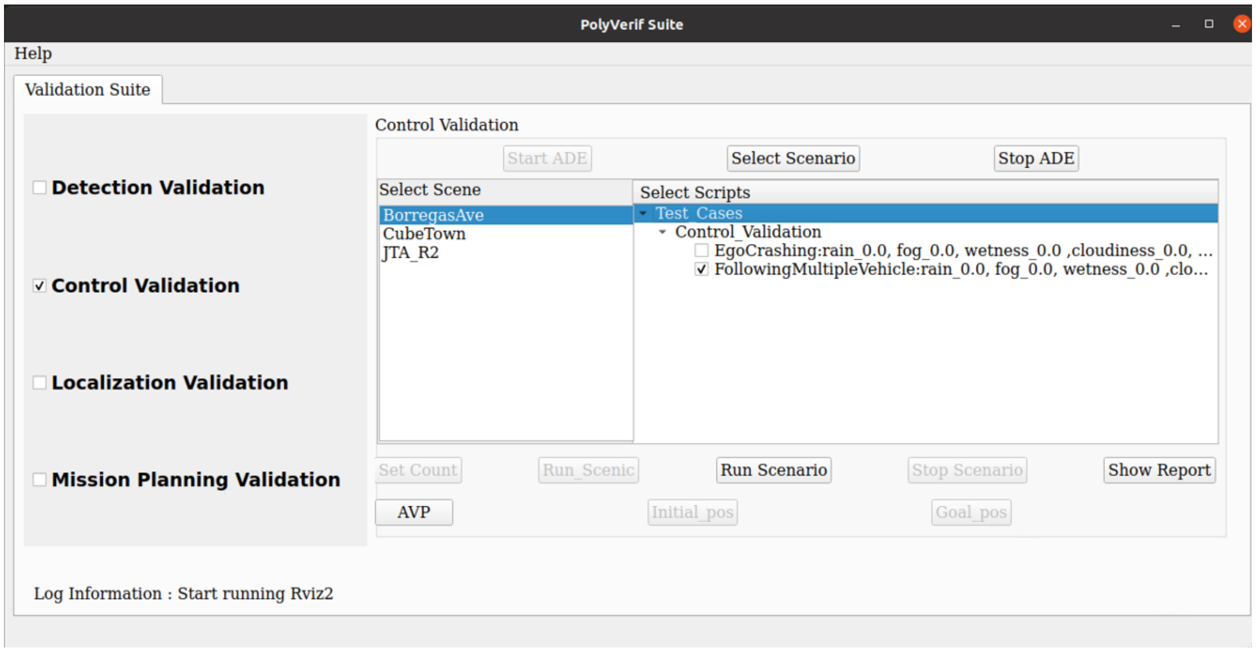
Located in the validation suite, this input is applicable when the user selects the *Control Validation* option. When selected, multiple options are generated (and can be selected via buttons) in the middle and right-hand side of the pane. However, only three buttons can be selected: *Select Scenario*, *Stop ADE*, and *Show Report*. This is because most of the other options only become applicable when a simulation is selected, running, or has finished running. The only useful option that the user can select at this time is Select Scenario. The user could select *Show Report;* however, it would show a report with no variables, displaying N/A for calculated results. Additionally, the user could also click *Stop ADE*, but nothing will happen at this time, as a simulation is currently not running.

When the user does click the *Select Scenario* button, several different options in both the Select Scene, and Select Scripts panes appear. For Select Scene, the default options are as previously mentioned in Section 3.1.1. For Select Scripts, the default option is *Test Cases,* with several suboptions available to determine the scenario being run (setting the conditions and variables to be tested).

The sub-options in the *Test Cases* differ from Detection Validation and will vary depending on the needs of the simulation being run, and the scenarios being tested.

When a scene and script are selected, this enables the *Run Scenario* button. When clicked, the simulation begins and once loaded, immediately runs. Data collection begins automatically.

After the simulation ends, the user can click on the Show Report button to generate a report highlighting the values of the tested variables (details on report specifics will be explained in the Output section). It is possible for the user to click on the *Show Report* button; however, this will lead to incorrect variable results, N/A as a result, or a combination of the two. It is the responsibility of the user to click the *Show Report* button only after the simulation ends to ensure correct results (note, no technical issues will arise due to this form of user error, only incorrect end values).



Log Information : Start running Rviz2

Figure 6. PolyVerif Validation Suite – Control Validation, Test Cases

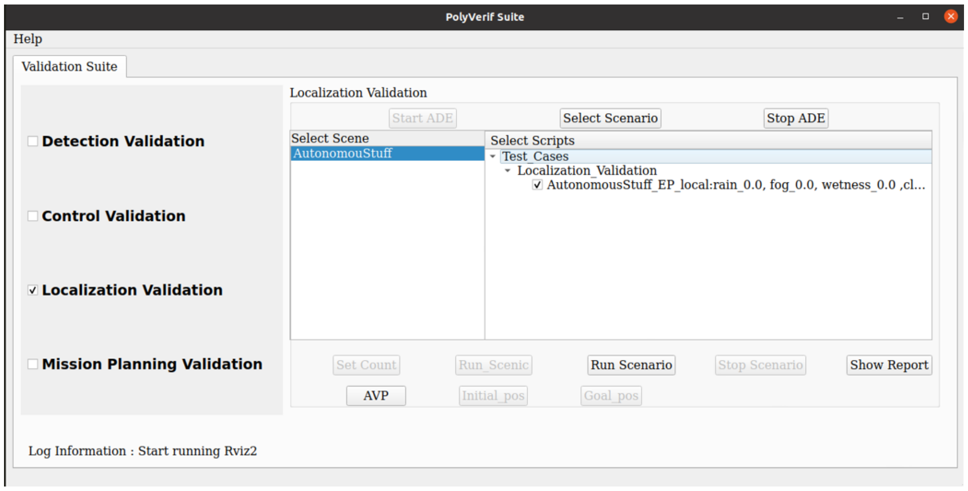
### 3.1.4. Localization Validation

Located in the validation suite, this input is applicable when the user selects the *Localization Validation* option. When selected, multiple options are generated (and can be selected via buttons) in the middle and right-hand side of the pane. However, only three buttons can be selected: *Select Scenario*, *Stop ADE*, and *Show Report*. This is because most of the other options only become applicable when a simulation is selected, running, or has finished running. The only useful option that the user can select at this time is Select Scenario. The user could select *Show Report;* however, it would show a report with no variables, displaying N/A for calculated results. Additionally, the user could also click *Stop ADE*, but nothing will happen at this time, as a simulation is currently not running.

When the user does click the *Select Scenario* button, several different options in both the Select Scene, and Select Scripts panes appear. For Select Scene, the options are as previously mentioned in Section 3.1.1. For Select Scripts, the option is *Test Cases,* with several suboptions available to determine the scenario being run (setting the conditions and variables to be tested).

When a scene and script are selected, this enables the *Run Scenario* button. When clicked, the simulation begins and once loaded, immediately runs. Data collection begins automatically.

After the simulation ends, the user can click on the Show Report button to generate a report highlighting the values of the tested variables (details on report specifics will be explained in the Output section). It is possible for the user to click on the *Show Report* button; however, this will lead to incorrect variable results, N/A as a result, or a combination of the two. It is the responsibility of the user to click the *Show Report* button only after the simulation ends to ensure correct results (note, no technical issues will arise due to this form of user error, only incorrect end values).



Log Information : Start running Rviz2

Figure 7. PolyVerif Validation Suite – Localization Validation

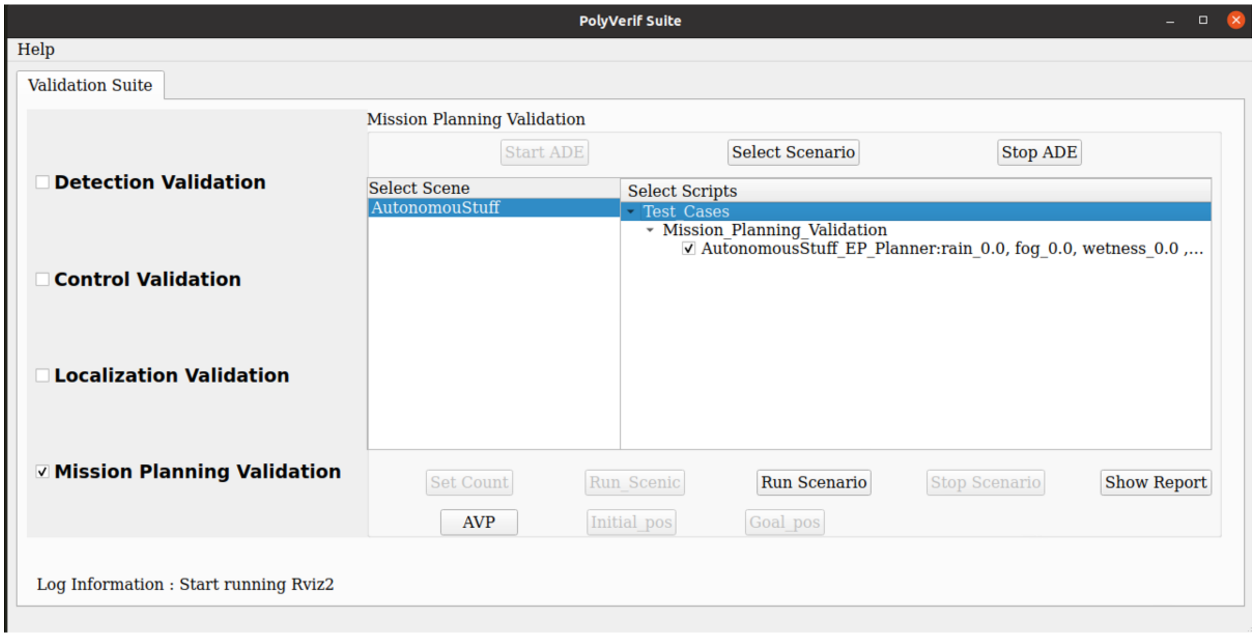
### 3.1.4. Mission Planning Validation

Located in the validation suite, this input is applicable when the user selects the *Mission Planning Validation* option. When selected, multiple options are generated (and can be selected via buttons) in the middle and right-hand side of the pane. However, only three buttons can be selected: *Select Scenario*, *Stop ADE*, and *Show Report*. This is because most of the other options only become applicable when a simulation is selected, running, or has finished running. The only useful option that the user can select at this time is Select Scenario. The user could select *Show Report;* however, it would show a report with no variables, displaying N/A for calculated results. Additionally, the user could also click *Stop ADE*, but nothing will happen at this time, as a simulation is currently not running.

When the user does click the *Select Scenario* button, several different options in both the Select Scene, and Select Scripts panes appear. For Select Scene, the options are as previously mentioned in Section 3.1.1. For Select Scripts, the option is *Test Cases,* with several suboptions available to determine the scenario being run (setting the conditions and variables to be tested).

When a scene and script are selected, this enables the *Run Scenario* button. When clicked, the simulation begins and once loaded, immediately runs. Data collection begins automatically.

After the simulation ends, the user can click on the Show Report button to generate a report highlighting the values of the tested variables (details on report specifics will be explained in the Output section). It is possible for the user to click on the *Show Report* button; however, this will lead to incorrect variable results, N/A as a result, or a combination of the two. It is the responsibility of the user to click the *Show Report* button only after the simulation ends to ensure correct results (note, no technical issues will arise due to this form of user error, only incorrect end values)



Log Information : Start running Rviz2

Figure 8. PolyVerif Validation Suite – Mission Planning Validation

## 3.2. Outputs

### 3.2.1. Show Report Selection

For the *Show Report* output to be applicable several conditions must first be met; a simulation must have run to completion, and once completed, the stop scenario button must be clicked. Only when these two conditions are met will the Show *Report* output function as intended. From here, the Show Report button can be pressed, and this will display a Detection Validation Report. From this pane, several options are presented to the user and will vary depending on how many simulations were previously run. These options are individual reports, with variables taken from the time the report was generated.

### 3.2.2. Selecting a Report: Detection Validation

When the conditions and tasks from Section 3.2.1 have been completed, the user is able to select a report to view from the Select Simulation Report Pane on the

Left-hand side of the Simulation Report Pane. When a report is selected, this triggers a dropdown option that has only one option. The name can vary and, in the figure, below, begins with FollowingMultiple... Clicking on this dropdown option will enable the *Generate Report* button, allowing it to be clicked. Clicking it will display the Detection Validation Report. The following is displayed to the user in the center of the Simulation Reports pane:

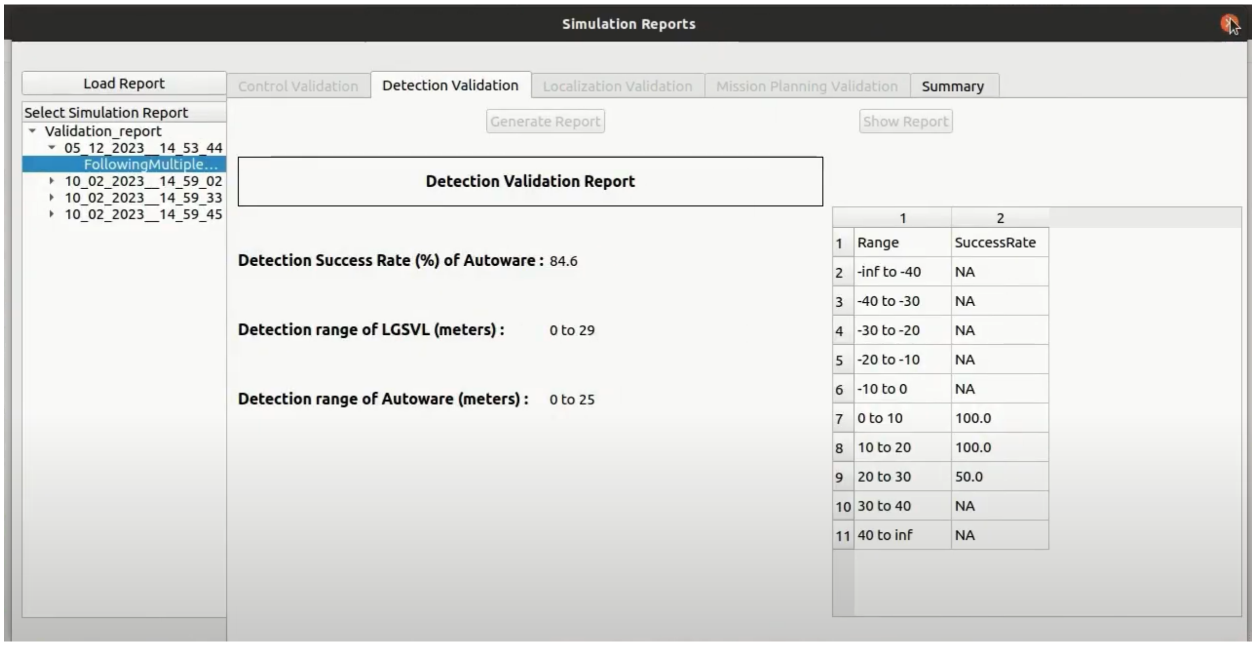
*Note: (*insert value here*) will be used for values generated by the simulation, as values can differ between runs. An example of a complete Detection Validation Report is provided below.

Detection Success Rate (%) of Autoware: (*insert value here*)

Detection range of LGSVL (meters): (*insert value here*)

Detection range of Autoware (meters): (*insert value here*)

Additionally, a table also appears on the right-hand side of the pane, displaying both Range and Success Rate. The values of the table will vary per simulation.



**Detection Validation Report**

**Detection Success Rate (%) of Autoware :** 84.6

**Detection range of LGSVL (meters) :** 0 to 29

**Detection range of Autoware (meters) :** 0 to 25

| | 1 | 2 |
|---|---|---|
| 1 | Range | SuccessRate |
| 2 | -inf to -40 | NA |
| 3 | -40 to -30 | NA |
| 4 | -30 to -20 | NA |
| 5 | -20 to -10 | NA |
| 6 | -10 to 0 | NA |
| 7 | 0 to 10 | 100.0 |
| 8 | 10 to 20 | 100.0 |
| 9 | 20 to 30 | 50.0 |
| 10 | 30 to 40 | NA |
| 11 | 40 to inf | NA |

Figure 9: PolyVerif – Simulation Report

### 3.2.3. Selecting a Report: Control Validation

When the conditions and tasks from Section 3.2.1 have been completed, the user is able to select a report to view from the Select Simulation Report Pane on the Left-hand side of the Simulation Report Pane. In the figure, when a report is selected, this triggers a dropdown option that has only one option. The name can vary, but in the in the figure below, begins with EgoCrashing... Clicking on this dropdown option will enable the *Generate Report* button, allowing it to be clicked. Clicking it will display the

Control Validation Report. The following is displayed to the user in the center of the Simulation Reports pane:

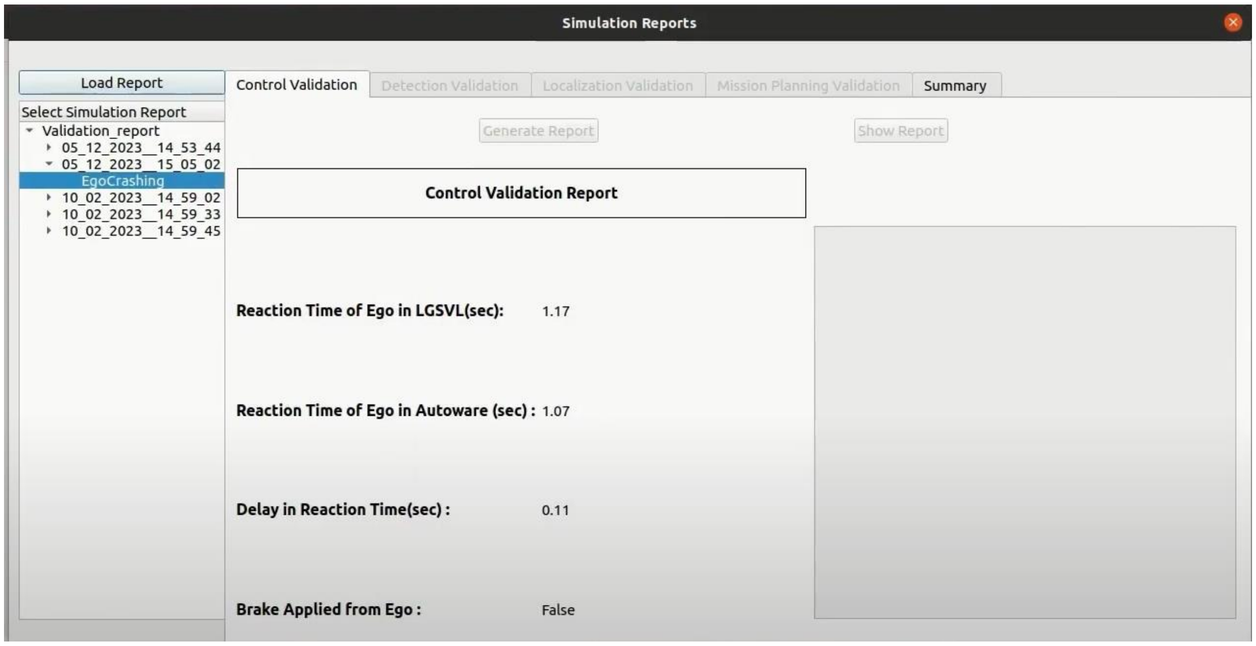
*Note: (*insert value here*) will be used for values generated by the simulation, as values can differ between runs. An example of a complete Control Validation Report is provided below.

Reaction Time of Ego in LGSVL (sec): (*insert value here*)

Reaction Time of Ego in Autoware(sec): (*insert value here*)

Delay in Reaction Time(sec): (*insert value here*)

Brake Applied from Ego: (*insert value here*)



Control Validation Report

Reaction Time of Ego in LGSVL(sec): 1.17

Reaction Time of Ego in Autoware (sec) : 1.07

Delay in Reaction Time(sec) : 0.11

Brake Applied from Ego : False

Figure 10: PolyVerif – Simulation Report, Control Validation

### 3.2.4. Selecting a Report: Localization Validation

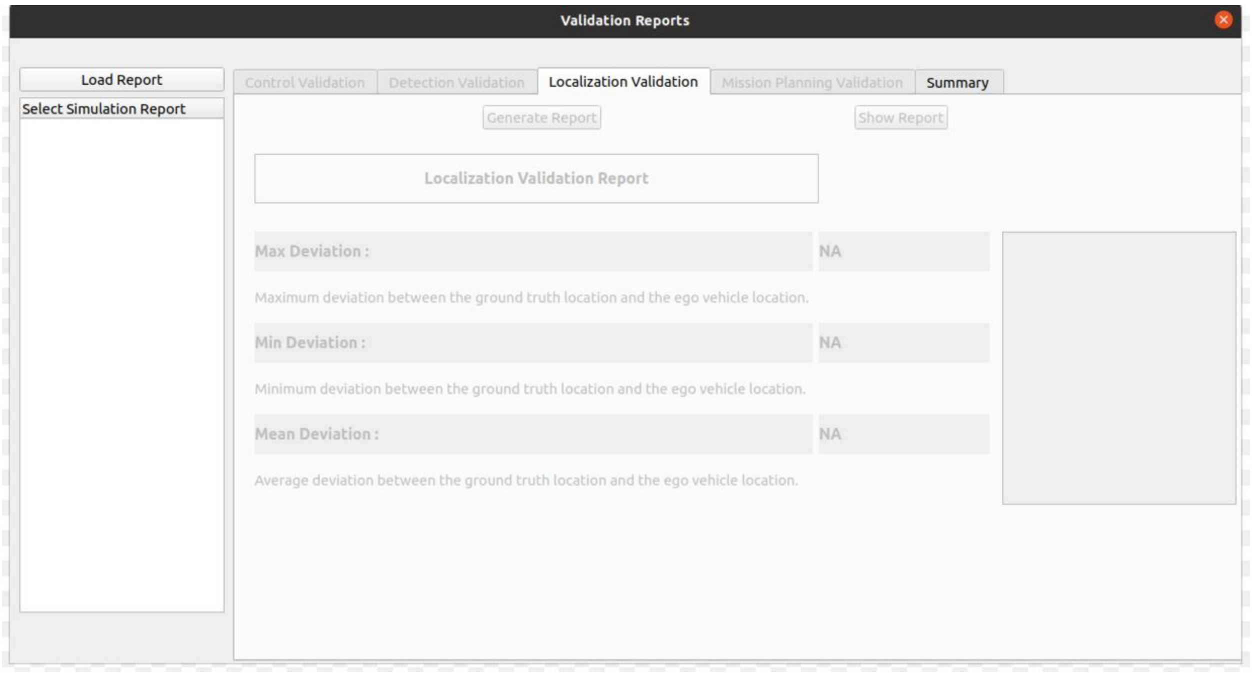
When the conditions and tasks from Section 3.2.1 have been completed, the user is able to select a report to view from the Select Simulation Report Pane on the Left-hand side of the Simulation Report Pane. In the figure, when a report is selected, this triggers a dropdown option that has only one option. The name can vary depending on the simulation and the user's PolyVerif setup. Clicking on this dropdown option will enable the *Generate Report* button, allowing it to be clicked. Clicking it will display the Localization Validation Report. The following is displayed to the user in the center of the Simulation Reports pane:

*Note: (*insert value here*) will be used for values generated by the simulation, as values can differ between runs. An example of a blank Localization Validation Report is provided below.

Max Deviation: (*insert value here*)

Min Deviation: (*insert value here*)

Mean Deviation: (*insert value here*)



Max Deviation : NA

Maximum deviation between the ground truth location and the ego vehicle location.

Min Deviation : NA

Minimum deviation between the ground truth location and the ego vehicle location.

Mean Deviation : NA

Average deviation between the ground truth location and the ego vehicle location.

Figure 11: PolyVerif – Simulation Report, Localization Validation

### 3.2.5. Selecting a Report: Mission Planning Validation

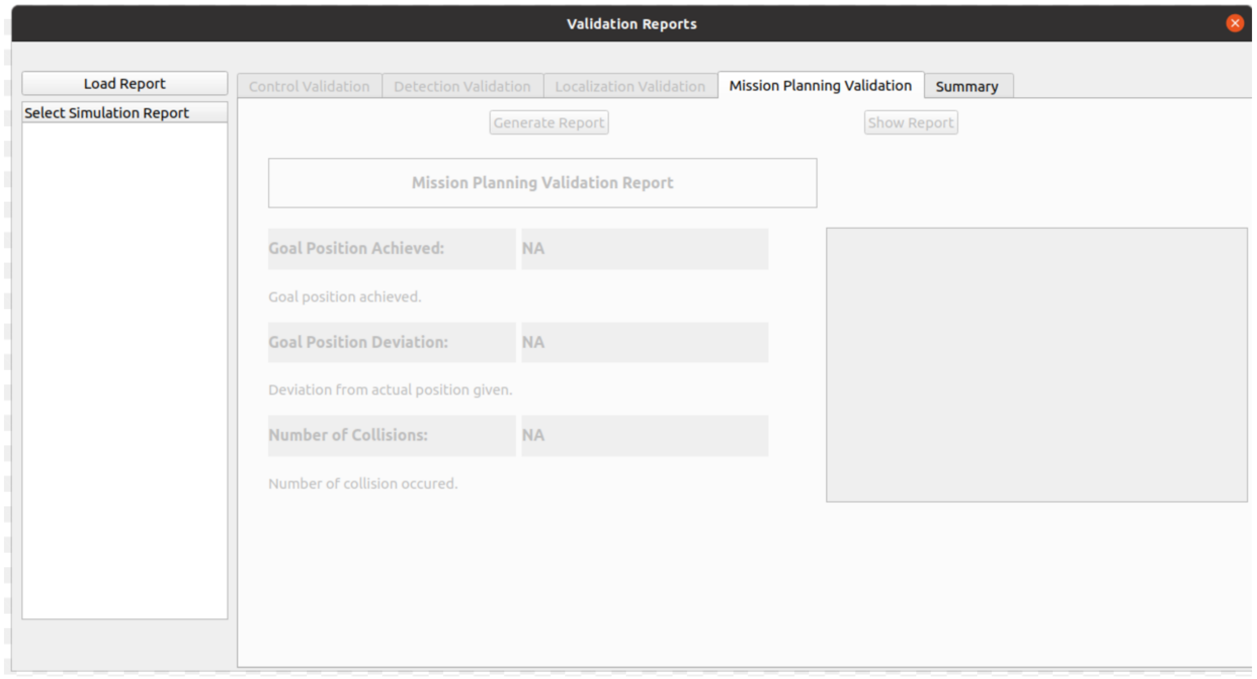
When the conditions and tasks from Section 3.2.1 have been completed, the user is able to select a report to view from the Select Simulation Report Pane on the Left-hand side of the Simulation Report Pane. In the figure, when a report is selected, this triggers a dropdown option that has only one option. The name can vary depending on the simulation and the user's PolyVerif setup. Clicking on this dropdown option will enable the *Generate Report* button, allowing it to be clicked. Clicking it will display the Mission Planning Validation Report. The following is displayed to the user in the center of the Simulation Reports pane:

*Note: (*insert value here*) will be used for values generated by the simulation, as values can differ between runs. An example of a complete Mission Planning Validation Report is provided below.

Goal Position Achieved: (*insert value here*)

Goal Position Deviation: (*insert value here*)

Number of Collisions: (*insert value here*)



Mission Planning Validation Report

Goal Position Achieved: NA

Goal position achieved.

Goal Position Deviation: NA

Deviation from actual position given.

Number of Collisions: NA

Number of collision occured.

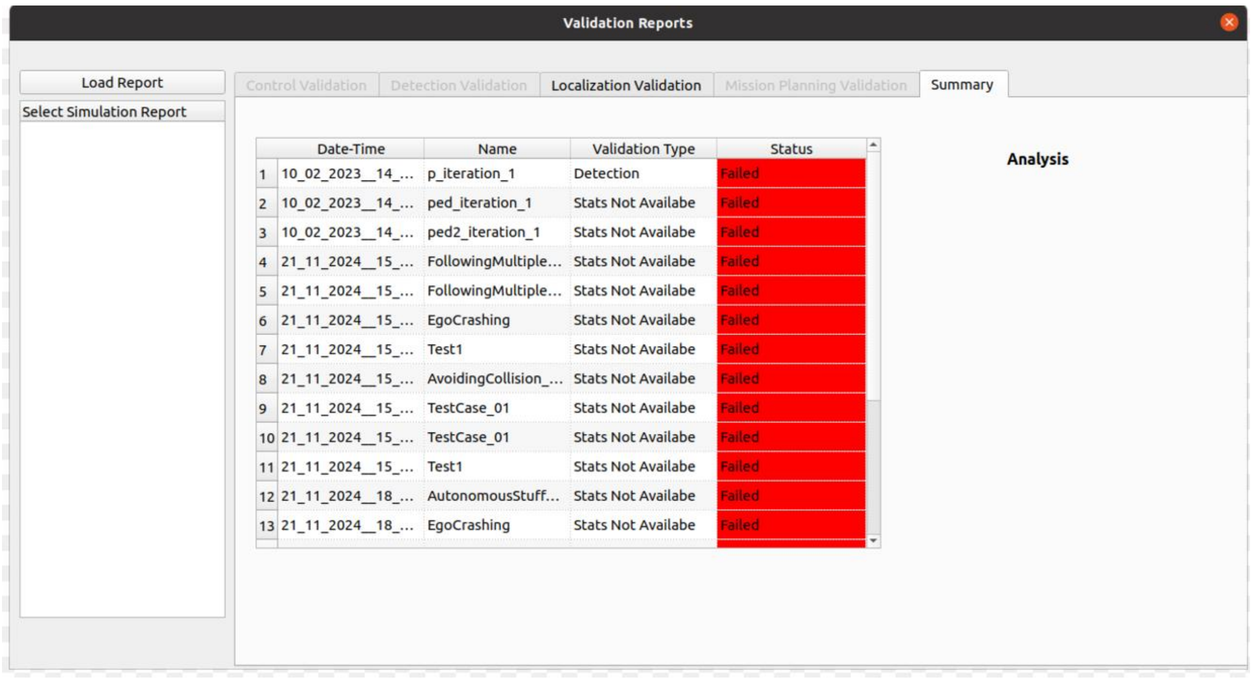
Figure 12: PolyVerif – Simulation Report, Mission Planning Validation

### 3.2.6. Selecting a Report: Summary

In addition to a report detailing the specifics of the simulation, there is a general summary that highlights if certain tests succeeded or failed. This can include:

- Did the vehicle collide with other objects?
- Did the vehicle reach its destination?
- Did pedestrians behave correctly as set by the user?
- Did other vehicles behave correctly as set by the user?

Below is a sample of a failed run.



| | Date-Time | Name | Validation Type | Status |
|---|---|---|---|---|
| 1 | 10_02_2023__14_... | p_iteration_1 | Detection | Failed |
| 2 | 10_02_2023__14_... | ped_iteration_1 | Stats Not Availabe | Failed |
| 3 | 10_02_2023__14_... | ped2_iteration_1 | Stats Not Availabe | Failed |
| 4 | 21_11_2024__15_... | FollowingMultiple... | Stats Not Availabe | Failed |
| 5 | 21_11_2024__15_... | FollowingMultiple... | Stats Not Availabe | Failed |
| 6 | 21_11_2024__15_... | EgoCrashing | Stats Not Availabe | Failed |
| 7 | 21_11_2024__15_... | Test1 | Stats Not Availabe | Failed |
| 8 | 21_11_2024__15_... | AvoidingCollision_... | Stats Not Availabe | Failed |
| 9 | 21_11_2024__15_... | TestCase_01 | Stats Not Availabe | Failed |
| 10 | 21_11_2024__15_... | TestCase_01 | Stats Not Availabe | Failed |
| 11 | 21_11_2024__15_... | Test1 | Stats Not Availabe | Failed |
| 12 | 21_11_2024__18_... | AutonomousStuff... | Stats Not Availabe | Failed |
| 13 | 21_11_2024__18_... | EgoCrashing | Stats Not Availabe | Failed |

Figure 13: PolyVerif – Validation Report, Summary

*<< This section describes the system output design relative to the user/operator. In this section, show and describe a mapping to the high-level data flows described in Section 1.2.1. System Overview. For example, outputs include reports, data display dialogs and GUIs, query results, etc. The following should be provided, if appropriate:*

- *Identification of field names for reports and data display GUI elements*
- *Description of report and screen contents (provide a graphical (image) representation of each layout and define all data elements associated with the layout or reference the data dictionary)*
- *Description of the purpose of the output, including identification of the primary users*
- *Description of any access restrictions or security considerations >>*

# 4. Detailed Design

*<< This section provides the information needed for a system development team to completely build and integrate the hardware, program and integrate the software, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate commercial off-the-shelf (COTS) (parts purchased from somewhere else and not made by the team) packages into a single system. Every subsystem itself and every detailed requirement should map back to the SRS. >>*

## 4.1. Hardware Detailed Design

This is not applicable to our project.

## 4.2. Software Detailed Design

There is no additional software or source code which we have implemented into the PolyVerif program at this time.

### 4.2.1. Database Management Subsystem

This is not applicable to our project in this current iteration.

### 4.2.2. Internal Communications Detailed Design

This is not applicable to our project in this current iteration.

*<< If the system includes more than one component there may be a requirement for internal communication to exchange information, provide commands, or support input/output functions. This section should provide enough detailed information about the communication design to correctly build and/or procure the communications components for the system. Include the following information in the detailed designs (as appropriate):*

- *The number of servers and clients to be included on each area network*
- *Specifications for bus timing requirements and bus control*
- *Format(s) for data being exchanged between components*
- *Graphical representation of the connectivity between components, showing the direction of data flow (if applicable), and approximate distances between components; information should provide enough detail to support the procurement of hardware to complete the installation at a given location*
- *LAN topology >>*

This is not applicable to our project in this current iteration.

# 5. External Interfaces

## 5.1. Hardware Interface Architecture

This is not applicable to our project, as we do not have a hardware component for our project.

## 5.2. Hardware Interface Detailed Design

This is not applicable to our project, as we do not have a hardware component for our project.

## 5.3. Software Interface Architecture

PolyVerif communicates with external systems and tools to manage simulations, validate scenarios, and generate result reports. The primary software interfaces include Scenic, PythonAPI, SUMO, the different validation modules, and the external scenario data.

Scenic defines and customizes objects, models, and environments in the simulation and uses Python-based scripting to communicate with the rest of the system. Scenic's interface is integrated with the PolyVerif directory for simulation management via the scenario files.

PythonAPI enables the modification of behaviors and environment parameters during simulations and provides a dynamic interaction between the user's code and simulation scenarios.

SUMO provides the capability to simulate complex urban environments, including traffic flow, road networks, and interactions between vehicles and infrastructure. PolyVerif integrates SUMO through PythonAPI, enabling PolyVerif to dynamically configure urban scenarios based on user inputs.

The validation modules validate critical functions like detection, control, localization, mission planning, and perception while processing simulation inputs and generating detailed result reports using local scripts. The validation modules interact with interal test cases and scenarios stored in PolyVerif.

The external scenario data can incorporate datasets or models from external sources such as OpenStreetMap (OSM) or pre-defined scenarios, such as Embry-Riddle Aeronautical University's Campus. The external scenario data is imported through file transfer protocols or direct downloads into the PolyVerif environment.

## 5.4. Software Interface Detailed Design

### 5.4.1 Scenic

Scenic uses .scenic files that are used to define object properties and environmental layouts. These files are compatible with PolyVerif's existing directory structure. Scenic validates object properties and environmental constraints before launching a simulation. Any errors detected are flagged in plain-text format and displayed in the PolyVerif console. These errors are then logged during scenario execution, and any invalid .scenic files produce warnings.

#### 5.4.2 PythonAPI

PythonAPI integrates .py scripts that define object behaviors or modify simulation parameters dynamically. PythonAPI executes scripts sequentially, raising exceptions for syntax or runtime errors, with outputs being displayed in real time. Any errors are logged in the system and are highlighted on the validation suite user interface.

#### 5.4.3 SUMO

PolyVerif integrates with SUMO using XML files to define road networks and configure traffic parameters, which can be either pre-configured in SUMO or dynamically generated by PolyVerif scripts. Communication operates in a loop-back mode where PolyVerif sends configurations, initiates SUMO simulations, and captures live updates for logging and visualization. Errors, such as missing files or invalid setups, are logged by SUMO and relayed to PolyVerif for user alerts or reports.

#### 5.4.4 Validation Modules

The validation modules require the input of scenario files, scripts, or configuration files, while generated result reports are output in .html or .csv files. The validation modules request specific scenarios and scripts during simulation execution. If any dependencies are missing, PolyVerif will halt its current process and alert the user. Simulation-specific errors such as invalid parameters or system crashes are logged, and the validation modules report any missing or invalid data with a clear visual indication.

#### 5.4.5 External Scenario Data

The external scenario data input into PolyVerif can be of many file formats, including .osm,. scenic, and .py file formats. Files are validated upon importation, and any incompatible files are prompted with error messages. Any issues or errors are displayed as pop-up alerts or detailed in error logs.

# 6. System Integrity Controls

Since PolyVerif utilizes a controlled environment, the main integrity control is the log-in system for the PC workstation located in the John Mica Engineering and Aerospace Innovation Complex (MicaPlex). The system prevents unauthorized users from having access to the simulation platform, ensuring that simulation operation and data is secure.