Autonomous Vehicle Scenario Modeling System Requirements Specification

Version 0.6

10/29/2024

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-22-24	All team members	Initialization of the document and addition of rudimentary information.
0.2	Oct-24-24	All team members	Continuation of documentation
0.3	Oct-28-24	Davian, Isabella, and William	Continuing to work on document, divide sections and assign work between team
0.4	Oct-28-24	Hannah	Update TOC, updating grammar in section 1, updating references, comments, updating bibliography.
0.5	Oct-29-24	Isabella	Work on remaining parts of Section 2.
0.6	Oct-29-24	All team members	Worked on Sections 4-7.

Table of Contents

1. Introduction

1.1. Purpose and Scope

PolyVerif provides itself as a noteworthy option in autonomous vehicle testing. However, seeing as preliminary documentation and software dates back as early as 2021, it is still a relatively new tool on the market. A lack of sufficient testing scenarios within PolyVerif's database leaves room for improvement, which is within the scope of this project's goals. To expand the testing within PolyVerif, Embry–Riddle Aeronautical University will be utilized as a testing ground. The campus provides a set of unique scenarios with varying traffic laws, object interactions, and traffic patterns. This should provide the necessary dynamic and customizable plane to bolster PolyVerif's capabilities.

1.2. Intended Audience and Reading Suggestions

This SRS includes all vital information required to properly understand the Autonomous Vehicle Project. For those unfamiliar with PolyVerif or simulation software, it is suggested to read through the entire document, focusing on background information first and then once a basic understanding is reached, continue reading through the document and focus on the more specific and technical sections. Below are some reading suggestions to fully understand the concepts gone over in this document:

Reading Suggestions:

For Those Unfamiliar with PolyVerif and Simulation Software:

https://www.dropbox.com/scl/fi/dmdztn1lk4r2dg8rvk9zd/PolvVerif_Format_v1.0.pptx?rlkey=8muost983lvnrraxnod5e5957&e=2&st=218c5k5l&bmus=1&dl=0

https://www.avvc.net/

For those Familiar with PolyVerif and Simulation Software:

https://drive.google.com/file/d/1vX68Kx4N8oJJodt5KfSYOHsDzV--3T9r/view?usp=sharing

1.3. Document Conventions

The fonts throughout this document are consistent for each section and there is no highlighting of sections, each section is separated by its header in bold.

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: <Definitions>

Term	Definition
PolyVerif	An open-source validation and verification 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.
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 product has high-level requirements
Scenic	A language for scenario specification and data generation

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

This does not apply to our project right now; we do not currently have any abbreviations.

Table 3: <Caption>

Term	Definition
e.g.	

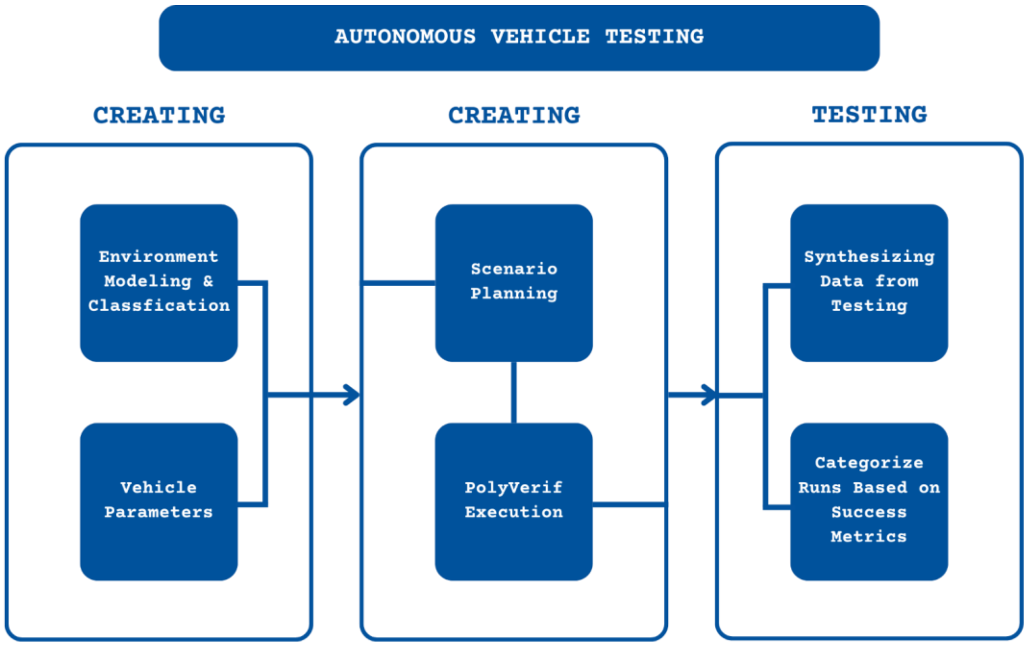
2. General Description

2.1. Product Perspective

This product is intended to expand upon pre-existing test simulations found in the PolyVerif GitHub. Currently, 4 distinct maps exist, with several different traffic scenarios involving each map. The product adds another map to these test simulations, the ERAU campus, and a new test scenario (the number of scenarios is subject to change depending on time). The structure and function of pre-existing code or products will not be altered with this product's introduction, rather, this project will utilize the software currently available, and add plugins, APIs, and supporting code, if necessary, but only to support the functionality of the ERAU map and scenario(s). The existing code allows us to build on the verification and validation of our initial scenario that we will be testing, as well as any other scenarios that we may create in the future.

These additional scenarios will be valuable to the overall PolyVerif system because the amount of raw data to pull from is lacking. The overall goal of PolyVerif is to assist in the creation of autonomous vehicles, and to have a successful autonomous vehicle, it needs to access data created from simulations to address as many different scenarios as possible. PolyVerif does have a library of scenarios to choose from, however, more scenarios to draw from means an objectively safer autonomous vehicle. The addition of the map and scenarios our product provides will contribute to the overall performance of PolyVerif when applied to real-world autonomous vehicles.

2.2. Product Features


AUTONOMOUS VEHICLE TESTING
CREATING
CREATING
TESTING
Environment Modeling & Classfication
Vehicle Parameters
Scenario Planning
PolyVerif Execution
Synthesizing Data from Testing
Categorize Runs Based on Success Metrics

This chart illustrates the functionality of the PolyVerif framework and functionality.

Both Environment Modeling and Classification and Vehicle Parameters are responsible for creating the landscape and environment for the digital twin vehicle to operate. Vehicle Parameters gives the vehicle instructions on how to operate (e.g, how to move forward, turn, stop, etc.)

Scenario planning is responsible for creating the problem needing to be tested by the simulation user. If the user wants to test an autonomous vehicle stopping at a red light, the user will select that scenario. Additionally, this scenario planning gives instruction to the vehicle parameters on how to operate, to move in accordance with the selected scenario.

PolyVerif execution is responsible for running the simulation according to the selected scenario.

Synthesizing Data from Testing and Categorize Runs Based on Success Metrics are both generated when the simulation successfully runs, and a report is generated that contains simulation data in an organized and efficient manner.

2.3. User Classes and Characteristics

<< This section should identify each type of user of the system (by function, location, type of device), the number in each group, and the nature of their use of the system. State any user characteristics that may be influential or significant in the structure of the product. Certain requirements may pertain only to certain user classes. Include possibly important characteristics such as educational level, experience level, privilege levels, and technical expertise as necessary.

This diagram is coming soon (use case diagram of the system).

2.3.1. Actors

This section presents the actors in the system.

- Simulation User
  - A user that runs the simulation and collects the data when the simulation ends
  - Can also use this data for external products, such as implementing the gathered data into an autonomous vehicle.
- GitHub Contributor
  - A person that contributes updates to the PolyVerif product as a whole.
  - A simulation user can make changes to PolyVerif locally, as PolyVerif is open source, however, these changes will not be reflected in the PolyVerif GitHub unless a GitHub Contributor approves these changes

2.3.2. Use Cases

This section presents the Use Cases, developed for the system.

- Running a simulation
  - A simulation user uses the system to collect data on various traffic scenarios. If the user needs to collect data on an autonomous vehicle response when at a red light and a car passing through th intersection, the user will run that corresponding scenario and collect the data from that simulation.
- Generating a report
  - When the simulation successfully runs, the user can generate a report of the simulation's results and corresponding data

2.3.3. Scenarios

<u>Scenario 1: Running Simulation</u>

Description: The user runs a simulation using the PolyVerif framework

Actors: Simulation User

Preconditions:

- PolyVerif has successfully been installed onto a machine with Linux Ubuntu OS

Trigger Condition: The user chooses to create an account.

Steps:

1. PolyVerif successfully loads, and the user is prompted to select a map for the simulation to run in. (ALT 1)
2. The user is prompted to select a type of scenario to run.
3. The user is prompted to run the simulation.
4. The simulation runs to completion. (ALT 2)

(ALT 1)

1.1. PolyVerif fails to load correctly.

(ALT 2)

4.1. The simulation fails to fully run to completion.

<u>**Scenario 2: Generating a report**</u>

Description: The user generates a report from a simulation

Actors: Simulation User

Preconditions:

- PolyVerif has successfully been installed onto a machine with Linux Ubuntu OS
- A simulation has successfully run to completion

Trigger Condition: The simulation is successfully run to completion

Steps:

1. The user is prompted to generate a report of the simulation results. (ALT 1)
2. The report loads, and the user has access to the report generated from the simulation.

(ALT 1)

1.1. The user is not prompted to generate a report
Note: This prompt should occur immediately after the simulation runs, if not, simulation must be re

(ALT 2)

2.1. Report fails to generate successfully due to errors in simulation.
2.2. Report generates successfully, but has faulty data present.

2.4. General Constraints

The machine must have Ubuntu V.18.04 or V.20.04 installed because it cannot run otherwise.

The simulation must be run in the terminal, it will not run otherwise.

Currently, the only computer usable is in the Micaplex MP 224(WiDe Lab), we are working on getting PolyVerif installed on Serena's computer.

2.5. Operating Environment

Micaplex MP 224 WiDe Lab Computer

OS: Linux

Ubuntu: Version 18.04 or later

2.6. User Documentation

Research Papers:

PolyVerif: An Open-Source Environment for Autonomous Vehicle Validation and Verification Research Acceleration

PolyFlows: Modular Test Framework Design for Autonomous Vehicles

An Integrated Scenario-Based Testing and Explanation Framework for Autonomous Vehicles

Testing and Validation Framework for Autonomous Aerial Vehicles

Scenic: a language for scenario specification and data generation

Websites:

SUMO Download - https://eclipse.dev/sumo/

Git Hub Repositories:

PolyVerif GitHub - https://github.com/PolyVerifFramework/PolyVerif

2.7. Assumptions and Dependencies

Assumptions:

Must be connected to Wi-Fi

PolyVerif must be successfully installed

User has list of test cases that are pre – determined

Dependencies:

Our simulation is built in PolyVerif

3. External Interface Requirements

3.1. User Interfaces

3.1.1. Mobile User Interface

This is not applicable to our project.

3.1.1.1. Login Screen

This is not applicable to our project.

3.1.1.1 Invalid Login Popup

This is not applicable to our project.

3.1.1.2. Register Screen

This is not applicable to our project.

3.2. Hardware Interfaces

This is not applicable to our project.

3.3. Software Interfaces

- PolyVerif
- Blender 3.5
- MapsModelImporter Plugin 0.6.2

- RenderDoc 1.26
- LGSVL 2020.06

3.4. Communications Interfaces

This is not applicable to our project.

4. Behavioral Requirements

4.1. Same Class of User

This is not applicable to our project, as all users will have the same permissions throughout the system.

4.2. Related Real-world Objects

[REQ-BEHAV.1] The ERAU Campus will be modeled using the Open Street Map (OSM) program.

[REQ-BEHAV.2] This model will include traffic intersections, vehicles, and environmental factors that are relevant to each scenario for our simulation(s).

[REQ-BEHAV.3] All intersections throughout the ERAU Campus will be modeled, but we will be focusing on the intersection between Aerospace Blvd and Clyde Morris for our simulation(s).

[REQ-BEHAV.5] This model will then be implemented into PolyVerif, allowing us to alter and customize things such as speed and position throughout our simulation.

[REQ-BEHAV.6] Within PolyVerif, the simulation will begin with the vehicle at a pre-determined starting point and the user will be able to begin the simulation, pause it, and stop it.

4.3. Stimulus

The PolyVerif simulation will not have any dynamic stimuli, with two main scenarios being simulated: (0) the autonomous vehicle will stop on time when approaching a red light, or (1) the autonomous vehicle will not stop on time and run a red light.

As such there are no stimuli requirements.

4.4. Functional

<< Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These include:

- *a) validity checks on the inputs,*
- *b) exact sequence of operations, and*
- *c) responses to abnormal situations, including:*
    - *1) overflow,*
    - *2) communication facilities, and*
    - *3) error handling and recovery.*

In addition, data flow diagrams and data dictionaries can be used to show the relationships between and among the functions and data. Refer to data flow diagram in this section.

Example:

[REQ-89] The default settings for the display windows shall be as specified in Table F-1 in Appendix F. >>

[REQ-FUNCT.1]

To be expanded on once system has been utilized more.

5. Non-behavioral Requirements

5.1. Performance Requirements

[REQ-PER.1] The system shall not accept more simulations than can be accurately simulated.

[REQ-PER.2] The system shall not terminate a simulation to begin a new simulation.

5.2. Safety Requirements

[REQ-SAFE.1] The system shall not accept corrupted files.

[REQ-SAFE.2] The system shall not corrupt files.

[REQ-SAFE.3] The system shall not store sensitive data.

5.3. Qualitative Requirements

<< There are several attributes of systems that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified. >>

5.3.1. Availability

<< This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. >>

[REQ-AVAIL.1]

To be expanded on once system has been utilized more.

5.3.2. Security

<< This should specify the factors that will protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:

a) utilize certain cryptographical techniques,
b) physical lockouts such as requiring a key to press a button
c) keep specific log or history data sets,
d) assign certain functions to different modules,
e) restrict communications between some areas of the program, and/or
f) check data integrity for critical variables. >>

[REQ-SEC.1]

To be expanded on once system has been utilized more.

5.3.3. Maintainability

<< This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement, for example, for certain modularity, interfaces, or complexity. Requirements should not be placed here just because they are thought to be good design practices. >>

[REQ-MAINT.1]

To be expanded on once system has been utilized more.

5.3.4. Portability

<< This should specify attributes of the system that relate to the ease of porting the system's software to other host machines and/or operating systems. This may include the following:

a) percentage of components with host-dependent code,
b) percentage of code that is host dependent,
c) use of a proven portable language,
d) use of a particular compiler or language subset, and
e) use of a particular operating system

This should also specify attributes of the hardware that relate to the ease of utilization, possibly also requirements on which terrain it can traverse, how easy the system is to carry, etc. >>

[REQ-PORT.1]

To be expanded on once system has been utilized more.

5.4. Design and Implementation Constraints

[REQ-CONST.1] The system will operate using PolyVerif.

[REQ-CONST.2] The system will run on Ubuntu V.18.04 or V.20.04.

6. Other Requirements

<< Miscellaneous sections for requirements that still are necessary but do not fall into the previous categories (which can also be altered) >>

6.1. Database Requirements

This is not applicable to this project.

6.2. Operations

[REQ-OPS.1] The system shall save all map render information locally.

[REQ-OPS.2] The system shall save all scenario information locally.

7. Analysis Models

7.1. Data Flow Model

7.1.1. Data Sources

This diagram will be coming soon.

7.1.2. Data Sinks

This diagram will be coming soon.

7.1.3. Data Dictionary

Name	***Description***	***Structure***	***Range***
	To be filled in.		

This diagram will be coming soon.

7.1.4. Context Diagram (Level 0 Data Flow Diagram)

This diagram will be coming soon.

7.1.5. Level 1 Data Flow Diagram

This diagram will be coming soon.

7.1.6. Level 2 Data Flow Diagram

This diagram will be coming soon.

7.2. Class Model

This diagram will be coming soon.

7.3. State Model

This diagram will be coming soon.

8. To Be Determined List

- PolyVerif may change to a different, unknown simulator.