Versatile Autonomous Navigation Testing and Guidance Environment System Requirements Specification

Version 1.1

02/04/2025

Document Control

Distribution List

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

Teaching assistants:

Alejandro Gonzalez Nunez

Customer(s):

Dr. M. Ilhan Akbas
Alejandro Gonzalez Nunez

Project team members:

Jim Pamplona
Jona Ortiz
Mai Evans
Jack Lee

Change Summary

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

Version	Date	Modifier	Description
0.1	10/25/2024	Jack Lee	Added title, version, distribution list
0.2	10/25/2024	Jim Pamplona	Fixed formatting
0.3	10/26/2024	Jim Pamplona, Jona Ortiz, Jack Lee	Edited 1.1, 1.2, 1.3, 1.4, 1.5, 1.5.1, 1.5.2, 1.5.3, 2.1, 2.2, 2.3, 2.3.1, 2.3.2, 2.3.3, 2.5, 2.6, 2.7, 3.2, 3.3, 3.4
0.4	10/28/2024	Jim Pamplona, Jona Ortiz	Edited 4.1,4.2,4.3,4.4,4.5,
0.5	10/29/2024	Mai Evans, Jack Lee	Edited/proofread: TOC, 1.2, 1.3, 1.4, 1.5.2, 2.3.3, 2.4 3.3, 4.1, 4.2, 4.3, 4.4, 4.5, 5.1, 5.2, 5.3, 5.3.1, 5.3.2, 5.3.3, 5.3.4, 5.4
0.6	10/31/2024	Jim Pamplona	Updated Use cases and DFDs
0.7	11/25/2024	Jack Lee	Edited: 1.1, 1.5.2, 2.1, 2.7, 4.3 Removed: Hardware Interfaces, Safety Requirements, Security
0.8	11/26/2024	Jim Pamplona	Updated all sections based on feedback from Alejandro.
0.9	2/1/2025	Jim Pamplona	Edited Purpose and Scope, Use Cases, and Requirements.
1.0	2/2/2025	Jack Lee, Jona Ortiz, Mai Evans	Created and filled in: 3.1, 3.2, 3.3, 3.5, 3.6, 3.8 Updated: Table of Contents, 1.5.2

1.1	2/4/2025	Jim Pamplona, Jack Lee	Added content to 3.7 Added content to 4.4

Table of Contents

1. Contents

2. Introduction

2.1. Purpose and Scope

As uncrewed aerial systems (UAS) become increasingly prevalent in both commercial and military applications, ensuring effective collision avoidance is critical. Traditional testing methods can be costly, time-consuming, and lack the scalability needed to evaluate a wide range of collision scenarios. The **Versatile Autonomous Navigation Testing and Guidance Environment (VANTAGE)** system addresses these challenges by providing a more efficient and reliable platform for testing and validating UAS collision avoidance systems. It enables researchers and analysts to analyze telemetry data and refine their avoidance strategies.

This System Requirements Specification (SRS**)** document outlines the functional requirements, interactions, and purpose of the VANTAGE system. The current iteration employs a two-tiered simulation approach, combining low- and high-fidelity simulations to ensure rigorous testing.

The scope of this system is to offer UAS researchers a comprehensive solution for assessing the effectiveness of collision avoidance systems. VANTAGE will utilize flight telemetry data to identify potential collisions and violations, providing actionable insights for system improvements. The system features a controller for managing both simulation tiers. AI functionalities and hardware integration are out of scope for this phase due to time constraints but may be explored in future iterations.

2.2. Intended Audience and Reading Suggestions

This document is intended for a variety of stakeholders involved in various aspects of the system's development, deployment, and use. Each type of stakeholder will find specific sections of this document that are more relevant to their roles regarding the system.

SRS-1. **Developers and Testers:** Sections 2.2, 2.3, 2.3.3, 2.5, 2.6, 3.3, 4, and 6
- Focus on creating, testing, and refining the system.
- Specified sections cover technical requirements, system interactions, testing procedures, and future enhancements, all essential for building, integrating, and validating the system.

SRS-2. **Project Managers**: Sections 1.1, 1.5, 2.1, 2.2, 2.3, 6.1.4, and 6.1.5
- Oversee the project's progress and ensure that objectives satisfy the scope, are met on time, and are within budget.
- Specified sections provide an overview of the system's purpose, requirements, milestones, and risk management, helping them track progress and ensure alignment with goals.

SRS-3. **Documentation Writers**: Sections 1.1, 1.2, 1.4, 1.5, and 2.3
-
- Sections covered explain the system's purpose, scope, and functionality, offering the necessary details to create clear and accurate user and technical documentation.

2.3. Document Conventions

There are no typographical conventions that denote special significance in this document.

2.4. Project References

- Product proposal document
  - *https://github.com/MLM-Simulation-and-Testing-for-UAS/mlmst-uas/blob/main/Docs/Project_Proposal_CS490.pdf*
- UAV Testing Sim Repository
  - https://github.com/AkbasLab/UAV-TestingSim
- *Gazebo documentation*
  - *https://gazebosim.org/docs/harmonic/install/*
- *ROS documentation*
  - *https://docs.ros.org/en/humble/index.html*
- *ArduPilot documentation*
  - *https://ardupilot.org/dev/docs/sitl-with-gazebo.html*
- *Julia documentation*
  - *https://docs.julialang.org/en/v1/*
- Python documentation
  - https://docs.python.org/3/
- PyQt5 documentation
  - https://www.riverbankcomputing.com/static/Docs/PyQt5/

2.5. Definitions, Acronyms, and Abbreviations

Various acronyms, abbreviations, and terms are utilized throughout this document that may be unknown to some (or most) stakeholders of this system. Please refer below to a glossary that specifies the definitions of said terms.

2.5.1. Definitions

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

Table 1: Definitions

Term	**Definition**
UAS	Unmanned Aerial systems; Aircraft that can be operated without a human pilot, crew, or passengers on board. UAS do not have to be autonomous, but the UAS systems simulated in this project are.
High-fidelity	A physics-based simulation of UAS within a 3D environment that comes close to the real-life scenarios with detailed sensor data
Low-Fidelity	A physics-based simulation of UAS modelled as points with no sensor data
ArduPilot	An Open source autopilot system.
Gazebo	An open-source 3D simulator that allows users to design robots, test algorithms, and perform regression testing in realistic environments

Python	A programming language that is used to create a variety of software and applications
Linux-based environments	A computing system that uses the Linux operating system.

2.5.2. Acronyms

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

Table 2: Acronyms

Term	Definition
SDD	System Design Document
VANTAGE	Versatile Autonomous Navigation Testing and Guidance Environment
UAS	Uncrewed Aerial Systems
ML	Machine Learning
ROS	Robot Operating System
CSV	Comma Separated Values
SITL	Software in the Loop
JSON	JavaScript Object Notation
COTS	Commercial-Off-The-Shelf Software
LiDAR	Light Detection and Ranging
CRUD	Create, Read, Update, Delete
LAN	Local Access Network
SAR	Search-And-Rescue

2.5.3. Abbreviations

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

Table 3: Abbreviations

Term	Definition
e.g.	Example given
etc.	Et cetera

3. General Description

3.1. Product Perspective

This project aims to develop an open-source framework that employs a two-tiered simulation model which shall start with a low-fidelity simulator for rapid testing and then process the data created into a high-fidelity 3d simulator for more precise collision avoidance.

This simulation-driven development would provide a safer and more cost-effective solution to testing complex behaviors. With the application development it would be ideal for experimenting with multi-agent scenarios, adjusting parameters, and observing the potential outcome in controlled scenarios. It would allow for users to focus more on development with the benefit of testing without physical limitations. This in turn would eliminate the need for regulatory compliance in the simulation phase and would help researchers and developers explore new UAS dynamics considering the constraints of air traffic compliance in later development stages.

The model is not self-contained, as it depends on employing a few existing technologies for its development and implementation, namely ArduPilot, Gazebo, ROS, and the programming languages Julia and Python. ArduPilot is an open source UAS development framework with built-in support for a wide range of drone types, sensors, and environments. Gazebo is a visual simulation software that can interface with ArduPilot to render drone simulations. ROS is a set of software libraries that empower developers to build robotics software. These technologies enable the developers of this project to create simulations, visualize them, and ultimately load them onto physical UAS to test them in the real world.

3.2. Product Features

The system provides a comprehensive testing environment for Uncrewed Aerial Systems (UAS) through a **two-tiered simulation approach.** The system is designed to optimize testing efficiency by seamlessly integrating low- and high-fidelity simulations. Users can transition between the two approaches for complementary testing, combining rapid evaluations with precise validations to enhance strategy development. A more high-level overview of the said features is defined in the list below:

Key Features:

1. **Low-Fidelity Simulation (JuliaSim):**
    - Enables fast evaluations of collision avoidance strategies.
    - Requires fewer resources, allowing rapid execution of numerous test cases in parallel.
    - Handles diverse scenarios to identify potential issues early in the development cycle.
    - Produces outputs to inform and refine high-fidelity simulations.
2. **High-Fidelity Simulation (Gazebo, ArduPilot, & ROS)**
    - Simulates real-world conditions with high accuracy.
    - Incorporates UAS sensors, cameras, and physical constraints for precise testing.
    - Provides a 3D view of simulation environments for better analysis.
    - **Final Strategy Validation:** Ensures that strategies are thoroughly tested before real-world deployment.
3. **Simulation Controller:**

- Acts as a central hub for managing simulation processes and parameters, supporting both predefined and custom testing scenarios.
- Integrates the low- and high-fidelity simulations to provide a unified interface for controlling both environments.
- Provides a graphical interface for setting up and running simulation scenarios, simplifying the input of telemetry parameters.
- Enables users to visualize simulation inputs and outputs in real-time, improving accessibility and usability.
- Allows users to store, load, and modify scenarios for iterative development.
- Enables reusability of scenarios to refine and test collision avoidance strategies efficiently.
- Logs telemetry data and simulation outputs for analysis, enabling users to track performance metrics and identify areas for improvement.

By integrating these features, the system ensures a simulation-driven approach for testing UAS behaviors, enabling users to explore new dynamics and refine strategies without physical limitations or regulatory constraints during the simulation phase.

3.3. User Classes and Characteristics

Commented [JP1]: Update

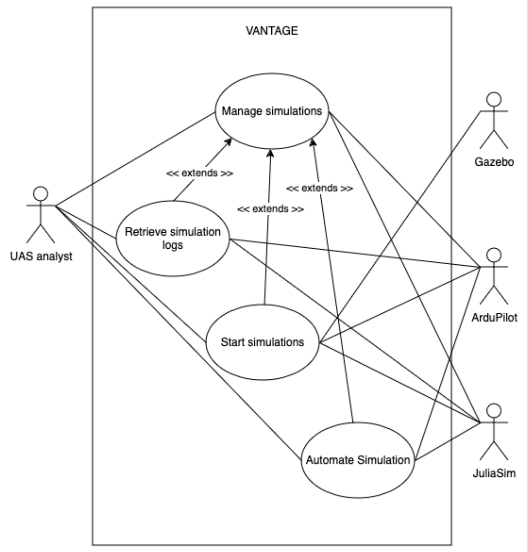


Figure 1. Use Case diagram of the System

3.3.1. Actors

This section presents the actors in the system.

- UAS Analyst
    - End user. Responsible for running, managing, and analyzing simulations completed on the system. Reviews logged data for trends, issues, and patterns. Provides insights for further development and optimization of UAS collision avoidance strategies.
- Gazebo
    - Displays the 3D environment in which the simulation is being run.
- ArduPilot
    - Acts as the brains of the high-fidelity simulations, as it provides real-time telemetry data for the said simulations.
- JuliaSim
    - Runs and displays low-fidelity simulations. It also provides telemetry data for the said simulations upon completion.

3.3.2. Use Cases

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

- Manage simulations
- Start simulation
- Retrieve simulation logs

3.3.3. Scenarios

Scenario 1: Manage Simulations

Description: The user creates, views, edits, or deletes a simulation

Actors: Current User (can be an UAS Analyst or Developer).

Precondition: The system has been launched successfully.

Trigger Condition: The user chooses to create, view, edit, or delete a simulation.

Steps:

1. The user selects the Create Simulation button (ALT 1, 2, and 3)
2. The system presents the user with a list of parameters and inputs
3. The user edits the parameters to create the simulation
4. The system saves the parameters to a simulation file
5. End of use Case.

(ALT 1: View Simulation)

1.1 The user selects a simulation
1.2 The system displays View, Edit, and Delete buttons
1.3 The user selects the View Simulation button
1.4 The system displays the simulations parameters
4.1 Use case continues at Step 5

(ALT 2: Edit Simulation)

1.1 The user selects a simulation
1.2 The system displays View, Edit, and Delete buttons
1.3 The user selects the Edit Simulation button
1.4 The selected simulation's parameters are displayed to the user
1.5 The user edits the parameters
1.6 Use case continues at Step 4

(ALT 3: Delete Simulation)

1.1 The user selects a simulation
1.2 The system displays Run, View, Edit, and Delete buttons
1.3 The user selects the Delete Simulation button
1.4 The selected simulation's file is deleted from the computer
1.5 Use case continues at Step 5

Scenario 2: Start Simulations

Description: The user starts a simulation

Actors: Current User (can be an UAS Analyst or Developer), JuliaSim, Gazebo, ArduPilot

Precondition: The system has been launched successfully.

Trigger Condition: None

Steps:

1. The user selects a simulation
2. The user selects the Run Simulation button
3. The system displays the Run Low-fidelity and Run High-fidelity buttons
4. The user clicks the Run Low-fidelity button (ALT 1)
5. The system sends the parameters to Julia Sim
6. JuliaSim creates the simulations and runs all combinations in parallel for rapid testing
7. The system logs all telemetry data in a log file
8. End of use case

(ALT 1: Run High-Fidelity)

4.1 The user clicks the Run High-fidelity button
4.2 The system prompts the user to select simulations to run
4.3 The user selects simulations to run
4.4 The system converts telemetry data to waypoint commands on ArduPilot
4.5 Gazebo and ArduPilot run the commands
4.6 The use case continues at Step 7

Scenario 3: Retrieve Simulation Logs

Description: The user views log history of a simulation

Actors: UAS Analyst

Precondition: The system has been launched successfully.

Trigger Condition: None

Steps:

1. The user selects a simulation
2. The user selects the View Logs button
3. The logs of past simulation runs are displayed to the user (ALT 1)
4. End of use case

(ALT 1: No logs are available)

3.1 A message indicating that no logs are available is displayed to the user
3.2 Use case continues at Step 4

3.4. General Constraints

Several key constraints will affect the project's progress.

1. System Accessibility: The system will be available in field applications and will be created as open-source software.

3. Framework and Middleware Compatibility:

- Since the **Gazebo version "11"** preserves compatibility with both **Ubuntu 20.04** and **ROS (Noetic)**, it must be used as the 3D simulation environment.

4. Programming Languages:

- The low-fidelity simulation is made with the **Julia programming language**.
- Controller integration is developed using **Python**, and the user interface utilizes the **PyQt5 Python package**.

5. Additional Software Requirements: The use of ArduPilot and Software in the Loop (SITL) simulation is mandatory, aligning with the project's open-source and field-accessibility goals.

6. Budget: The project will also be constrained to a currently unspecified budget.

3.5. Operating Environment

The system operates on a computer running the **Ubuntu 20.04** operating system. The Ubuntu 20.04 operating system may be loaded with or without a hypervisor. The computer should have **at least 16 Gigabytes of RAM and 50 Gigabytes of storage available**. To use the models, **Gazebo 11, ROS Noetic, and ArduPilot** need to be installed on the computer. The computer's processor should be supported by the forementioned software, and an **x86_64** architecture is recommended. The only physical environmental constraints are that the computer should be able to operate safely in the environment (e.g., not submerged in water or subjected to extreme temperatures).

3.6. User Documentation

The documentation includes:

- A guide on manually installing and setting up the software dependencies
- A script to automatically install the software dependencies
- A description of the system’s components and their functions

3.7. Assumptions and Dependencies

The development team has made the following assumptions:

- The source code will be provided to the user
- The software dependencies will be available for the user to download and install
- The project dependencies will remain open source
- All user interfaces may change depending on the requirements

4. Broader Impact

4.1. Public Health

This project will improve public health by reducing the number of aviation accidents involving UAS devices. The VANTAGE system is designed to make it easier for researchers to perform autonomous UAS simulations, which will improve UAS collision avoidance and navigation systems. A drone with improved navigation and collision avoidance will be less likely to collide with a person or inanimate object, which could cause debris to fall onto people below. The VANTAGE system is likely to reduce the number of collisions involving UAS, potentially saving lives. Drone swarms could also be developed for Search-And-Rescue (SAR) efforts after natural disasters like earthquakes or wildfires. The drone swarms would be able to quickly cover a large area and communicate the location of potential survivors to a human team, saving lives and speeding up rescue efforts.

4.2. Safety

The VANTAGE project will improve the safety of drone operators and other individuals within the area of drone operations. The simulations produced by the VANTAGE system will help to improve collision avoidance and UAS swarming technology for enhanced safety for everybody involved. Drone swarms could also be used for SAR operations, improving disaster recovery efforts and increasing the likelihood that victims of natural disasters survive. The improvement of public safety and of SAR operations are two of the main targets of UAS research, and the VANTAGE system aims to assist researchers in achieving these goals.

4.3. Welfare

UAS are likely to become more integrated into society as time and technology progress. VANTAGE will play a key role in the development of safer drone navigation, collision avoidance, and swarming algorithms which will greatly improve autonomous UAS safety. The potential benefits to the general welfare of individuals in society are innumerable. Drones could be used to help deliver items such as food and medical supplies to areas in need, or even to pick up online shopping or food orders. One of the key requirements of the VANTAGE system is its Open-Source nature. This will allow anybody to perform complex UAS simulations, allowing for fair access to the technology. This could even potentially drive more economic competition, creating job opportunities and driving down the cost of basic UAS equipment.

4.4. Global Effects

VANTAGE is an open-source project, so there are no licensing restrictions for users. Anybody across the globe should be able to access and use the software. However, foreign governments could censor or limit access to the software within their own internet systems if they decide to. Because the VANTAGE system is completely software-based, the only environmental requirements are that the computer running the system is kept in a cool environment. Most locations should be able to accommodate this need. UAS research could become more popular in underrepresented areas of the world because the VANTAGE project is free and open source. This could drive more corporate interest into the UAS industry as well, increasing the involvement of autonomous UAS in everyday life.

4.5. Cultural Effects

The VANTAGE project could foster public trust in drone technology by demonstrating the effectiveness of the app such as air navigation aid, how drones can be tested beforehand, and many other ways. This emphasis could encourage developers to adopt similar practices encouraging accountability and responsibility in the field. Beyond regulatory benefits, VANTAGE aims to optimize drone operations, reducing the inefficiencies and indirectly promoting environmental awareness. This project could set a precedent for future innovations with sustainable system design. Furthermore, its cost-effective approach may encourage smaller airports to adopt the application offering an alternative approach to airspace monitoring. The VANTAGE system could have as small implications as changing the public's view on drone safety to massive implications such as FAA compliance in the airports.

4.6. Societal Effects

The impacts of drones in society are overwhelming and do exist in many aspects other than military application. Agriculturally, they are used for crop monitoring, pesticide spraying, and soil analysis, improving efficiency and sustainability. In disaster response, drones assist in search-and-rescue missions, delivering aid to hard-to-reach areas, and assessing damage after natural disasters. They also play a crucial role in infrastructure inspection, helping to maintain bridges, power lines, and pipelines with reduced risk to human workers. UAS are even transforming industries such as filmmaking, journalism, and delivery services, enabling new perspectives and faster logistics. In healthcare, they are being used to transport medicine and medical supplies to remote or disaster-stricken areas, improving access to critical care. Wildlife conservation efforts also benefit from UAS technology, as they help track endangered species and monitor habitats without disturbing the environment. However, their widespread use raises concerns about privacy, air traffic regulation, and security. This calls for a proper balance between innovation and policy. That is why a system like VANTAGE is necessary in streamlining how UAS are tested to meet safety-critical standards.

4.7. Environmental Effects

The VANTAGE system employs preventative and protective measures that can be utilized by UAV researchers to enhance safety and reliability. By analyzing telemetry data to identify patterns leading to collisions and violations, the system enables proactive avoidance strategies. This predictive capability helps mitigate the risk of mid-air incidents, reducing potential debris caused by collisions and ensuring safer UAV operations. Overall, this system indirectly helps environments where UAVs are operated to be more sustainable.

4.8. Economic Effects

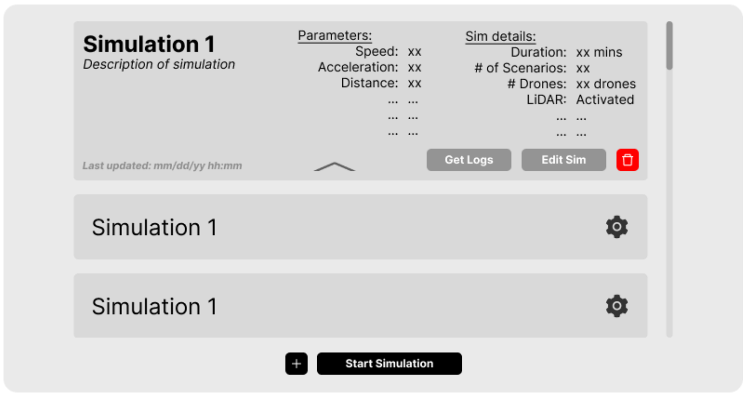
While the VANTAGE system does not primarily aim to affect economic factors, there are notable potential side effects from the improvement and widespread adoption of autonomous UAS with advanced collision avoidance. Namely, the increased use of drones as a delivery solution. Drones could be used to deliver food, online purchases, medical equipment, military equipment, and just about anything. The drone delivery industry is likely to create technical competition, which will create jobs and lessen the cost of basic drone equipment for the average person to begin learning about UAS. This would also drive more government spending and research into UAS improvements, which could allow autonomous UAS to fulfill even more roles. Jobs such as delivery drivers or couriers could be negatively impacted by the improvement in UAS driven by the VANTAGE system, however the losses in those employment areas will likely be overshadowed by the job gains in the technology and UAS development sectors.

5. External Interface Requirements

5.1. User Interfaces

SRS-1. The system shall display a user interface that implements all, but not limited to, the features demonstrated within the open-source Figma prototype for this system.

https://www.figma.com/design/3LFUXZGglFq9kiKeomucW9/VANTAGE?node-id=0-1&node-type=canvas&t=xTfRp2XFzhZyfPz2-0



Simulation 1
Description of simulation

Parameters:
Speed: xx
Acceleration: xx
Distance: xx
... ...
... ...
... ...

Sim details:
Duration: xx mins
# of Scenarios: xx
Drones: xx drones
LiDAR: Activated
... ...
... ...

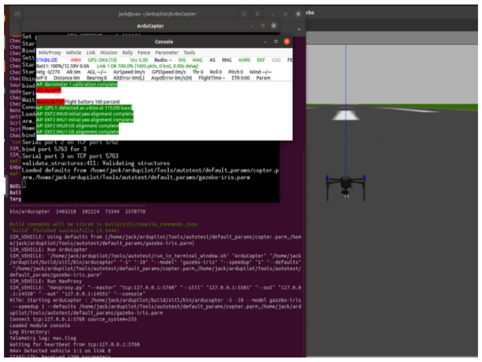
Last updated: mm/dd/yy hh:mm

Figure 2. Home display – Figma Prototype

SRS-2. The system shall have a simulation details interface.
SRS-3. The simulation details interface shall display all details regarding the simulation found in the corresponding CSV file, e.g. parameters, duration, start time, end time.
SRS-4. The system shall have a logs interface.
SRS-5. The logs interface shall categorize all logs by number, date, duration, total simulations.
SRS-6. The logs interface shall have a button to change between displaying the high- or low-fidelity logs.
SRS-7. The system shall have a parameter input interface.
SRS-8. The parameter input interface shall provide a dropdown menu to select one of the simulation maneuvers listed below:

a. Vertical
   i. Adjustments in altitude
b. Horizontal
   i. Adjustments in lateral position
c. Right of Way (RoW)
   i. Compliance with predefined FAA priority rules as reported in requirement SRS-23.

SRS-9. The system shall display simulations on Gazebo for real-time 3D visualization.


jack@uas: ~/ardupilot/ArduCopter
ArduCopter
Console
MAVProxy Vehicle Link Mission Rally Fence Parameter Tools
STABILIZE ARM GPS: OK6 (10) Vcc 5.00 Radio: -- INS MAG AS RNG AHRS EKF LOG
Batt1: 100%/12.59V 0.0A Link 1 OK 100.0% (1695 pkts, 0 lost, 0.00s delay)
Hdg 0/270 Alt 0m AGL --/-- AirSpeed 0m/s GPSSpeed 0m/s Thr 0 Roll 0 Pitch 0 Wind --/--
WP 0 Distance 0m Bearing 0 AltError 0m(L) AspdError 0m/s(H) FlightTime -- ETR 0:00 Param

```
AP: Barometer 1 calibration complete
Init Gyro***

Ready to FLY Flight battery 100 percent
AP: GPS 1: detected as u-blox at 115200 baud
AP: EKF2 IMU0 initial yaw alignment complete
AP: EKF2 IMU1 initial yaw alignment complete
AP: EKF2 IMU0 tilt alignment complete
AP: EKF2 IMU1 tilt alignment complete
```

```
Serial port 2 on TCP port 5762
bind port 5763 for 3
Serial port 3 on TCP port 5763
validate_structures:411: Validating structures
Loaded defaults from /home/jack/ardupilot/Tools/autotest/default_params/copter.p
arm,/home/jack/ardupilot/Tools/autotest/default_params/gazebo-iris.parm
```

```
----------------------------------------------
bin/arducopter  2403210  102224  73344  2578778

Build commands will be stored in build/sitl/compile_commands.json
'build' finished successfully (4.544s)
SIM_VEHICLE: Using defaults from (/home/jack/ardupilot/Tools/autotest/default_params/copter.parm,/hom
e/jack/ardupilot/Tools/autotest/default_params/gazebo-iris.parm)
SIM_VEHICLE: Run ArduCopter
SIM_VEHICLE: "/home/jack/ardupilot/Tools/autotest/run_in_terminal_window.sh" "ArduCopter" "/home/jack
/ardupilot/build/sitl/bin/arducopter" "-S" "-I0" "--model" "gazebo-iris" "--speedup" "1" "--defaults"
 "/home/jack/ardupilot/Tools/autotest/default_params/copter.parm,/home/jack/ardupilot/Tools/autotest/
default_params/gazebo-iris.parm"
SIM_VEHICLE: Run MavProxy
SIM_VEHICLE: "mavproxy.py" "--master" "tcp:127.0.0.1:5760" "--sitl" "127.0.0.1:5501" "--out" "127.0.0
.1:14550" "--out" "127.0.0.1:14551" "--console"
RiTW: Starting ArduCopter : /home/jack/ardupilot/build/sitl/bin/arducopter -S -I0 --model gazebo-iris
 --speedup 1 --defaults /home/jack/ardupilot/Tools/autotest/default_params/copter.parm,/home/jack/ard
upilot/Tools/autotest/default_params/gazebo-iris.parm
Connect tcp:127.0.0.1:5760 source_system=255
Loaded module console
Log Directory:
Telemetry log: mav.tlog
Waiting for heartbeat from tcp:127.0.0.1:5760
MAV> Detected vehicle 1:1 on link 0
```

Figure 3. Gazebo and Ardupilot

5.2. Software Interfaces

Software interfaces that included in the project are as follows:

SRS-10. The system shall utilize **Julia** for low-fidelity simulations, providing the necessary computational framework for rapid low-fidelity testing and analysis of UAS collision avoidance systems.

Version Number: Latest stable release
Source: https://docs.julialang.org

SRS-11. The system shall integrate with **ROS** to enable robotic control and communication between simulation components and hardware systems, ensuring proper interaction between drones and enabling utility enhancements provided by the program.

Version: Noetic
Source: https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html

SRS-12. The system shall utilize **Ardupilot** for flight control, allowing for the testing and development of autonomous flight behaviors within the Gazebo environment.

Version: Latest stable release
Source: https://ardupilot.org/

SRS-13. The system shall incorporate **Gazebo** for high-fidelity simulation of UAS environments, providing realistic virtual environments for testing collision avoidance strategies with accurate physics and environmental models.

Version Number: 11
Source: https://gazebosim.org/docs/harmonic/install_ubuntu/

SRS-14. The system shall support **Python** for scripting, automation, and integration tasks, enabling the development of tools for data processing, system control, and interactions with external systems.

Version Number: 3.x or higher
Source: https://www.python.org/downloads/release/python-3130/

SRS-15. The system shall run on **Ubuntu** operating system to provide a stable and efficient development environment, supporting the required software packages and dependencies for the project.

Version Number: 20.04 LTS
Source: https://releases.ubuntu.com/focal/

5.3. Communications Interfaces

SRS-16. **:** The system shall interface **Gazebo, Ardupilot, and ROS** as the high-fidelity subsystem, with all components communicating via MAVLink.
SRS-17. The system shall export simulation results as a CSV file for analysis and reporting.
SRS-18. The system shall transfer and communicate consistent data between the high-fidelity and low-fidelity subsystems via the Simulation Controller.

6. Behavioral Requirements

6.1. Same Class of User

There are no different access levels for any users.

6.2. Related Real-world Objects

SRS-19. The system shall simulate real-world quad-copter drones.
SRS-20. Obstacles in the simulation shall influence agent navigation and response behaviors.
SRS-21. The system shall simulate an isolated environment in which the entities will be tested within.
SRS-22. The Right of Way maneuver shall be defined, as per the FAA:

a. 14 CFR § 91.113 - Right-of-way rules: Except water operations. (e) “When aircraft are approaching each other head-on, or nearly so, each pilot of each aircraft shall alter course to the right.”

6.3. Stimulus

SRS-23. The system shall execute the corresponding functionality upon a button click (e.g., start, create, edit, delete simulations).
SRS-24. The system shall expand the simulation details when a simulation is clicked.
SRS-25. The system shall have a deletion confirmation popup when the Trash Can button is clicked.
SRS-26. The system shall delete the simulation when the Delete button is clicked in the deletion confirmation popup.
SRS-27. The system shall exit the deletion confirmation popup when the cancel button is clicked.
SRS-28. The system shall open the parameter input interface when the "New Simulation (+)" button is clicked.
SRS-29. The system shall validate all input parameters upon user input.
SRS-30. When in the parameter input interface, the system shall dynamically change the required parameters based on the selected maneuver type.
SRS-31. The parameter input interface shall display the following parameters when the Vertical maneuver is selected.

Parameter Name	**Unit**
Drone speed	Miles per hour
Heli speed	Miles per hour
Drone X position	Feet
Drone X position	Feet
Drone Direction	Degrees
Drone response distance	Feet
Ascent rate	Feet per second

Figure 4. Vertical Maneuver Parameter list

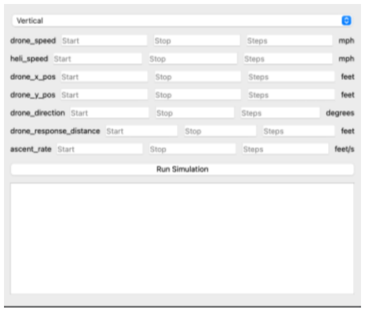

Vertical
drone_speed Start Stop Steps mph
heli_speed Start Stop Steps mph
drone_x_pos Start Stop Steps feet
drone_y_pos Start Stop Steps feet
drone_direction Start Stop Steps degrees
drone_response_distance Start Stop Steps feet
ascent_rate Start Stop Steps feet/s
Run Simulation

Figure 5. Parameter input Interface (Vertical Maneuver)

SRS-32. The parameter input interface shall display the following parameters when the Horizontal maneuver is selected.

Parameter Name	**Unit**
Drone speed	Miles per hour
Heli speed	Miles per hour
Drone X position	Feet
Drone X position	Feet
Drone Direction	Degrees
Drone response distance	Feet
Drone horizontal turn rate	Degrees (per second)
Drone horizontal turn angle	Degrees (per second)

Figure 6. Horizontal Maneuver Parameter list

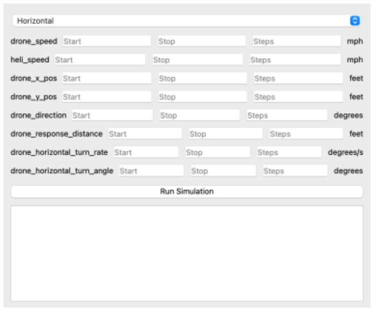

Horizontal
drone_speed Start Stop Steps mph
heli_speed Start Stop Steps mph
drone_x_pos Start Stop Steps feet
drone_y_pos Start Stop Steps feet
drone_direction Start Stop Steps degrees
drone_response_distance Start Stop Steps feet
drone_horizontal_turn_rate Start Stop Steps degrees/s
drone_horizontal_turn_angle Start Stop Steps degrees
Run Simulation

Figure 7. Parameter input Interface (Horizontal Maneuver)

SRS-33. The parameter input interface shall display the following parameters when the RoW maneuver is selected.

Parameter Name	**Unit**
Drone speed	Miles per hour
Heli speed	Miles per hour
Drone X position	Feet
Drone X position	Feet
Drone Direction	Degrees
Drone response distance	Feet
Drone horizontal turn rate	Degrees (per second)
Drone horizontal turn angle	Degrees (per second)
Force right turn	Boolean

Figure 8. RoW Maneuver Parameter list

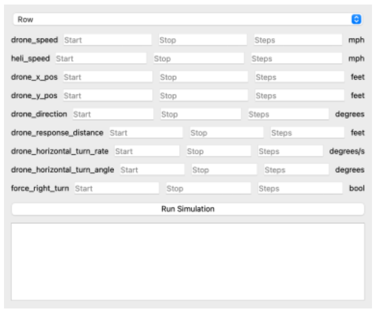

Row
drone_speed Start Stop Steps mph
heli_speed Start Stop Steps mph
drone_x_pos Start Stop Steps feet
drone_y_pos Start Stop Steps feet
drone_direction Start Stop Steps degrees
drone_response_distance Start Stop Steps feet
drone_horizontal_turn_rate Start Stop Steps degrees/s
drone_horizontal_turn_angle Start Stop Steps degrees
force_right_turn Start Stop Steps bool
Run Simulation

Figure 9. Parameter input Interface (RoW Maneuver)

SRS-34. The system shall open the simulation interface when the edit sim button is clicked.
SRS-35. The system shall enable editing of parameters when the simulation interface is clicked.
SRS-36. The system shall save the parameters to the simulation file, log the time and date, and exit the simulation interface when the save button is clicked.
SRS-37. The system shall revert all changes to the original and exit the simulation interface when the cancel button is clicked.
SRS-38. The system shall open the logs interface when the Get Logs button is clicked.
SRS-39. The system shall open the operating system's download popup when the save button is clicked in the logs interface.
SRS-40. The system shall allow users to sort data in ascending or descending order when clicking on a column header in the logs interface.
SRS-41. The system shall exit the logs interface when the done button is clicked.
SRS-42. The system shall log a violation if an obstacle enters the LiDAR scanning radius of a drone.
SRS-43. The system shall log a collision if the drone collides with an obstacle.

6.4. Related Features

SRS-44. The LiDAR device on the drone shall continuously monitor for other drones and obstacles at the user's selected radius.
SRS-45. The pathing feature on Ardupilot shall allow drones to follow predetermined waypoints and commands.

6.5. Functional

SRS-46. The system shall enable the simulation of two drones (agents) operating in the same environment.
SRS-47. The system shall simulate head-on simulations utilizing two drones.
SRS-48. The system shall allow the user to select specific scenarios from the low-fidelity simulations and run them on the high-fidelity module.

SRS-49. The system shall save all collision or violation occurrences in a csv log file that consists of:

a. Telemetry data
b. Time of occurrence (simulation time)
c. Minimum distance

SRS-50. The system shall validate the parameters based on the following:

a. Non-negative decimals for start value and end value
b. Non-negative integers for steps

SRS-51. The system shall update the high-fidelity simulation environments' settings automatically based on the provided parameters.

SRS-52. The high-fidelity module shall provide real-time telemetry and visualization of UAS flight paths.

SRS-53. The system shall use LiDAR data from Gazebo's sensor system to determine violations and obstacle distance.

SRS-54. The system shall create combinations via a grid search of given parameters to find results for each scenario possible.

SRS-55. The low-fidelity module shall run all scenarios possible for each simulation.

SRS-56. The system shall convert parameters to waypoint commands in the high-fidelity module.

7. Non-behavioral Requirements

7.1. Performance Requirements

SRS-57. The system shall be ran within a device that meets the specs specified by all dependencies of the system, notably:

a. Gazebo
b. Ardupilot
c. ROS
d. JuliaSim
e. Python

SRS-58. The system shall run all low-fidelity simulations in parallel depending on their maneuver type.

SRS-59. The system shall allow non-visual execution modes for computational efficiency in the low-fidelity module.

7.2. Qualitative Requirements

7.2.1. Availability

SRS-60. The system shall detect and handle errors by ending the simulation with an error log without compromising the entire simulation.

SRS-61. The system shall ensure that all logged data is saved after every simulation.

7.2.2. Maintainability

SRS-62. The system shall have an installation guide that dictates how to install all dependencies of the system.
SRS-63. The system shall have script(s) that installs all dependencies required for the system.
SRS-64. The system shall remain open source to ensure accessibility for the research and developer community.

7.2.3. Portability

SRS-65. The system shall operate on a UNIX-based environment that supports the installation of all dependencies.

7.3. Design and Implementation Constraints

SRS-66. The system shall support Python as the primary language for controlling simulations for ease of use and compatibility with a wide range of operating systems.
SRS-67. The system shall utilize Gazebo and Ardupilot as the primary software utilized for high-fidelity simulation testing.
SRS-68. The system shall utilize Julia as the primary software utilized for low-fidelity simulation testing but may be ported into Rust at a later iteration.

8. Analysis Models

8.1. Data Flow Model

8.1.1. Data Sources

The data sources and their inputs to the system identified in the data flow model are as follows:

- UAS Analyst
  - Command inputs
  - Parameters
  - System issue alerts
- Gazebo
  - 3D rendering
- ArduPilot
  - Simulation data
- JuliaSim
  - Simulation data

8.1.2. Data Sinks

The data sinks and their system outputs identified in the data flow model are as follows:

- UAS Analyst
    - Simulation logs
    - Telemetry data
- ArduPilot
    - Sim parameters
- JuliaSim
    - Sim parameters

8.1.3. Data Dictionary

The data types are described in the data dictionary below. This section includes the name of the data type; a description of the contained data; how the data is structured; and the range of values.

Name	***Description***	***Structure***	***Range***
Command input	Interactive buttons that trigger specific commands or actions within the simulation. Each button corresponds to a predefined function, allowing the user to activate or deactivate features easily.	*Bool*	*0-1*
Parameters	A string input used to specify configuration settings or operational parameters for the simulation.	*String*	*0<*
3D render display	A visual output that provides a real-time, three-dimensional representation of the simulation environment. It shows the physical behaviors and interactions of objects in the simulation.	*Display*	*1*
JuliaSim render display	A visual output dedicated to rendering results or processes within the JuliaSim framework. It focuses on showing system dynamics, mathematical models, or any visualization specific to the JuliaSim platform.	*Display*	*1*
Sim Parameters	Parsed parameters specific to the simulation framework that utilizes them.	*CSV*	*0<*
Simulation data	Refers to the raw output data generated by the simulation. It may include information like event logs, performance metrics, or scenario results.	*CSV*	*0<*

Telemetry Data	Refers to parsed logs that contains the analyzed data that states whether violations or collisions have been made.	*CSV*	*0<*
Real-time Sim display	A visual output that monitors progress of both simulation environments.	*Display*	*1*

Figure 6. Data Dictionary

8.1.4. Context Diagram (Level 0 Data Flow Diagram)

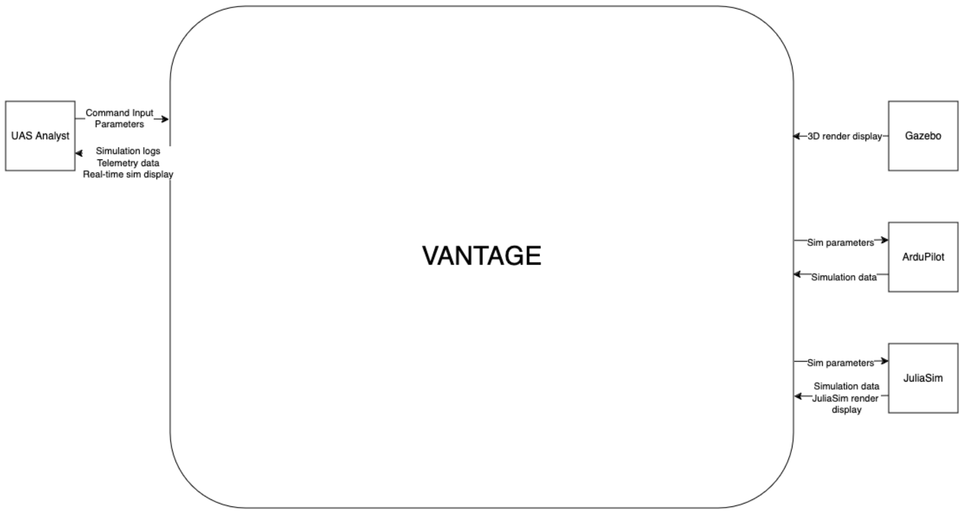


Figure 7. Context Diagram

8.1.5. Level 1 Data Flow Diagram

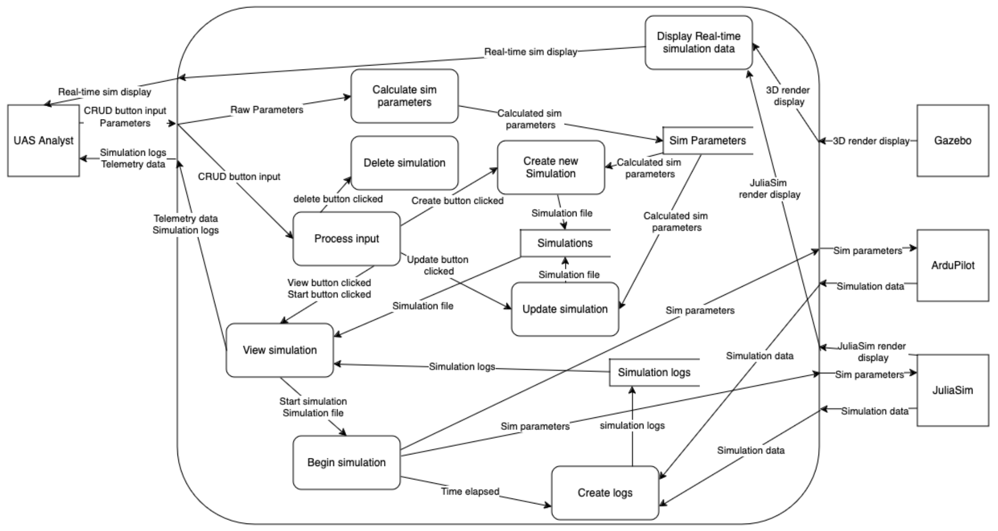


Figure 8. DFD Level 1

9. To Be Determined List

At the current iteration, there is nothing listed that is To Be Determined.