Versatile Autonomous Navigation Testing and Guidance Environment System Test Plan

Version 0.3

12/5/2024

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:

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	12/3/2024	Mai Evans, Jona Ortiz	Edited title, version, distribution list, 1.1, 1.2, 1.3, 1.7, 2.6
0.2	12/4/2024	Mai Evans, Jona Ortiz	Edited 1.7, 1.4, 1.5, 1.6, 2
0.3	12/4/2024	Jim Pamplona	Edited 2.6, 3, 4.1, 5

Table of Contents

Document Control ii
Distribution List ii
Change Summary ii
1. Introduction 1
1.1. Purpose 1
1.2. Scope 2
1.3. System Overview 2
1.4. Testing Approach Overview 4
1.5. Testing Entrance Criteria 4
1.6. Document Overview 5
1.7. References 5
2. Testing Approach 6
2.1. Testing Types 6
2.1.1. Usability Testing 6
2.1.2. Functionality Testing 6
2.2. Testing Suspension Criteria and Resumption Requirements 6
2.2.1. Suspension Criteria 7
2.2.2. Resumption Requirements 7
2.3. Testing Environment 7
2.4. Testing Assumptions 7
2.5. Testing Risks and Contingencies 7
2.6. Test Plan 9
3. Test Schedule 11
4. Traceability Matrix and Defect Tracking 12
4.1. Traceability Matrix 12
4.2. Defect Severity Definitions 13
5. Test Cases 14
5.1. Test Case TC-1 14
5.2. Test Case TC-2 14
5.3. Test Case TC-3 15
5.4. Test Case TC-4 16

1. Introduction

1.1. Purpose

The increasing use of uncrewed aerial systems (UAS) in commercial and military applications highlights the critical need for robust collision avoidance systems. These systems must enable UAS to navigate safely in complex environments, avoiding other aircraft, structures, and obstacles. Traditional testing methods for collision avoidance, however, are often resource-intensive, costly, and lack scalability, making it challenging to test a wide range of potential scenarios effectively.

The **Versatile Autonomous Navigation Testing and Guidance Environment (VANTAGE)** system addresses this challenge by providing an innovative framework that integrates artificial intelligence (AI) to enhance the testing and validation of UAS collision avoidance strategies. By leveraging AI, the system enables dynamic scenario generation, adaptive simulation control, and in-depth analysis, allowing researchers to explore a broader spectrum of conditions and behaviors with greater efficiency.

The purpose of this test plan document is to outline the structured approach to testing and validating the Versatile Autonomous Navigation Testing and Guidance Environment (VANTAGE) system, ensuring its effectiveness and reliability in supporting UAS collision avoidance strategies.

The document provides a comprehensive framework detailing the following:

- **Project Overview**: A brief description of the VANTAGE system, its objectives, and its role in advancing UAS collision avoidance testing.
- **Test Objectives**: The specific goals of the testing process, including validation of AI-driven scenario generation, adaptive simulation control, and performance analysis capabilities.
- **Organizational Responsibilities**: Clear description of roles and responsibilities among the teams involved in the testing process, ensuring accountability and efficient execution.
- **Test Approach**: A detailed methodology for conducting tests, including tools, techniques, and metrics to assess system performance under various conditions.
- **Test Environment and Resources**: Description of the required hardware, software, and personnel necessary to execute the tests.
- **Test Schedule**: A timeline outlining the sequence of testing activities, milestones, and deadlines.
- **Risk Management**: Identification and mitigation strategies for potential challenges and risks that could impact the testing process.

- **Reporting and Documentation**: Procedures for capturing test results, generating reports, and documenting findings to inform system improvements and ensure transparency.

By defining these elements, this document serves as a roadmap for evaluating the VANTAGE system's capabilities, ensuring that it meets its intended purpose of enhancing UAS collision avoidance through innovative AI-driven solutions.

1.2. Scope

For this semester, the scope of the VANTAGE system project encompasses the development and validation of a working skeleton of the system with the following features:

- **Simulation Controller**: A functional controller enabling the creation, management, and execution of simulations in both high-fidelity and low-fidelity environments.
- **Simulation Execution**: Testing the system's capability to run dynamic simulations accurately across the specified fidelity levels.

Out-of-Scope for This Semester:

- **AI Functionalities**: Integration of artificial intelligence-driven features such as dynamic scenario generation and adaptive simulation control will not be included in this release.
- **Hardware Integration**: Testing and incorporating hardware components will be deferred to subsequent project phases.

Future Versions:

- These excluded features will be introduced and tested in the next semester, expanding the system's capabilities to include AI and hardware support, aligning with the project's broader objectives.

This semester's release aims to establish a solid foundation for the VANTAGE system, focusing on its core simulation functionalities to ensure a reliable framework for further development and integration in future phases.

1.3. System Overview

For this semester, the VANTAGE system will focus on its core functionality, enabling the creation and execution of high-fidelity and low-fidelity simulations through a controller.

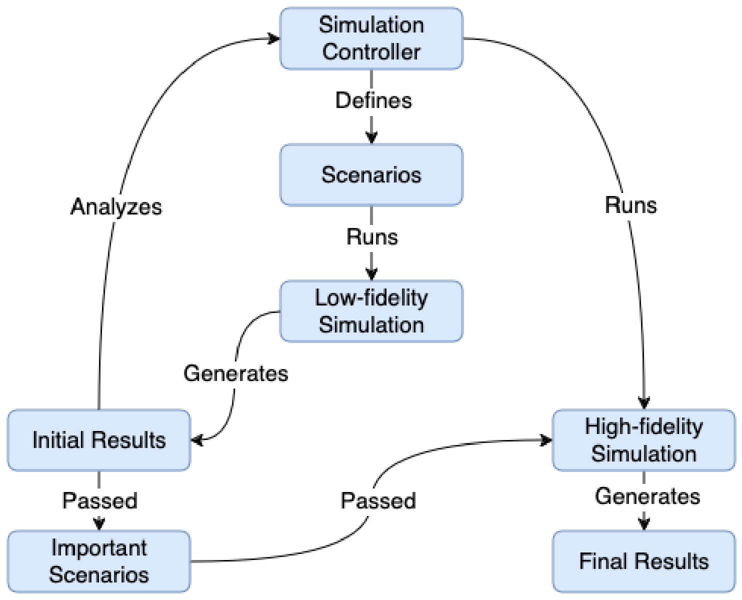

Figure 1. System Flow Chart

The system operates as follows:

- **Simulation Controller**:
    - Users interact with the system via a graphical user interface (GUI).
    - The controller manages the creation of simulation scenarios, defining environmental parameters, obstacles, and aircraft behaviors.
- **Simulation Environment**:
    - **High-Fidelity Simulations**: Provide detailed modeling for precise evaluation of UAS behaviors in complex conditions.
    - **Low-Fidelity Simulations**: Offer simplified scenarios for rapid testing and iteration.
- **Data Collection and Analysis**:
    - Outputs include performance metrics such as collision statistics and violations.

The testing approach for this semester will emphasize:

- **Controller Functionality**: Validating that the simulation controller reliably supports scenario creation and execution.
- **Simulation Fidelity**: Ensuring that both high-fidelity and low-fidelity environments perform as expected and generate consistent, usable outputs.

- **Data Integrity**: Verifying that simulation results are accurate, correctly formatted, and align with defined performance metrics.

Future testing phases will expand to include AI-driven features and hardware integrations, building upon the foundational capabilities established in this semester's scope.

1.4. Testing Approach Overview

The VANTAGE system's testing approach will follow a multilevel strategy to validate its functionality, usability, and reliability. Testing activities will cover all major components of the system, focusing on both core functionalities (e.g., simulation creation and execution) and interface interactions between subsystems. Key aspects include:

- **Usability Testing**: Verifies user interface accessibility, navigation, and output presentation.
- **Functionality Testing**: Ensures compliance with business and technical requirements for simulation control, execution, and data handling.
- **Integration Testing**: Confirms seamless interaction among components, including high-fidelity (Gazebo, ArduPilot) and low-fidelity (Julia-based) systems via the simulation controller.
- **Stress and Performance Testing**: Evaluates the system's ability to handle high volumes of simulations and large datasets without degradation in performance.

This plan will ensure that each component of the system performs as expected, providing a solid foundation for expanding functionalities in subsequent phases.

1.5. Testing Entrance Criteria

System testing for the VANTAGE system will commence when the following requirements for testing readiness are met:

- All components of the system are developed and integrated, including the low-fidelity (Julia-based) and high-fidelity (Gazebo and ArduPilot) simulation subsystems.
- The simulation controller is operational, facilitating communication between the high- and low-fidelity modules.
- The Ubuntu 22.04 testing environment is fully configured with all necessary dependencies, libraries, and software.
- The hardware environment meets the specified requirements for stable and efficient operation.
- Test plan, test cases, and expected outputs are reviewed and approved.

- All relevant system documentation, including installation guides, is complete.

1.6. Document Overview

This test plan document provides a structured approach to testing the VANTAGE system. It is designed to ensure thorough validation of system functionality, usability, and performance while addressing potential risks and contingencies. The document guides all testing activities, ensuring alignment with project objectives and stakeholder expectations.

The contents are organized as follows:

- **Section 1: Introduction**
    - Outlines the purpose, scope, and objectives of the test plan.
    - Provides an overview of the system, testing approach, and key references.
- **Section 2: Testing Approach**
    - Details the types of tests to be performed, criteria for suspension and resumption of testing, and the testing environment.
    - Highlights assumptions, risks, and mitigation strategies associated with the testing process.
- **Section 3: Test Schedule**
    - Defines the timeline and milestones for the execution of test activities.
- **Section 4: Traceability Matrix and Defect Tracking**
    - Describes the traceability matrix to ensure all requirements are tested.
    - Outlines the process for defect tracking, including severity definitions and resolution workflows.
- **Section 5: Test Cases**
    - Contains specific test cases with detailed scenarios, expected results, and success criteria.

This document serves as a blueprint for managing the testing lifecycle, ensuring that the VANTAGE system meets its requirements and delivers a robust solution.

1.7. References

Project Documentation:

- **System Requirements Specification (SRS)**: Details functional and non-functional requirements for the VANTAGE system, including simulation controller operations and performance criteria.

- **System Design Document (SDD)**: Describes the architecture, component interactions, and design of the high-fidelity and low-fidelity simulation environments.

Testing Standards and Guidelines:

- **IEEE Standard 829-2008**: Software and System Test Documentation, providing a framework for structuring the test plan and associated documents.

These references collectively ensure that the test plan aligns with project objectives, adheres to industry standards, and provides a solid foundation for evaluating the VANTAGE system's performance and reliability.

2. Testing Approach

2.1. Testing Types

2.1.1. Usability Testing

In order to verify that the UI of the simulators are accessible, user-friendly, and accurate the following tests shall be run:

- **UI Navigation Test:** Assess navigation through system menus and the ability to locate core functionalities.
- **Simulation Setup Test:** Confirm that users can configure simulation parameters without errors or ambiguities.
- **Output Presentation Test:** Ensure simulation results are presented in an easily interpretable format (e.g., graphs, tables).
- **Test Procedure Specification:** Test cases will detail interaction steps, expected outputs, and required UI conditions.
- **Test Incidents Reporting:** Incidents will be logged with screenshots, descriptions, and a severity level assigned.

2.1.2. Functionality Testing

To ensure that the functional requirements of the system are met, the following tests shall be run:

- **Collision Avoidance Test**: Validate the UAS's ability to detect and avoid obstacles in various scenarios.
- **Parameter Configuration Test:** Confirm that simulation settings can be modified and saved accurately.
- **Data Logging Test:** Ensure all simulation data is captured and logged correctly for analysis.

2.2. Testing Suspension Criteria and Resumption Requirements

This section will specify the criteria that will be used to suspend all or a portion of the testing activities on the items associated with this test plan.

2.2.1. Suspension Criteria

Testing will be suspended if the incidents found will not allow further testing of the system/application under-test. If testing is halted, and changes are made to the hardware, software, or database, it is up to the Testing Manager to determine whether the test plan will be re-executed, or part of the plan will be re-executed.

2.2.2. Resumption Requirements

Resumption of testing will be permitted once the issue that caused the suspension has been identified and resolved. The specific functionality must undergo successful retesting to confirm that it operates as expected and meets the defined requirements. Additionally, all dependent systems and processes affected by the suspension must also be validated to ensure no residual issues remain. Only after the successful completion of these steps will testing resume from the point of interruption.

2.3. Testing Environment

This test plan and the following test cases will be executed in a controlled testing environment. The testing environment will consist of an Ubuntu 22.04 system with all of the dependencies installed. The Ubuntu system will be standalone (e.g. a virtual machine or bare-metal installation).

2.4. Testing Assumptions

The testing process assumes that both JuliaSim and Gazebo ArduPilot will remain stable and updated, with hardware and software configured to meet performance needs. Input data is expected to be accurate, consistent, and representative of real-world scenarios, and communication between the simulators should function without disruptions. Test cases will align with system requirements, with predefined pass/fail criteria and clear documentation. It is assumed that necessary personnel and resources are available, and external disruptions, such as network issues, will not occur. The simulation environment is expected to mimic real-world conditions closely to ensure valid results.

2.5. Testing Risks and Contingencies

Here are a few risks that can happen:

- Simulator may become unstable during extended runs or under stress conditions. Periodic restarts and checkpoints may help minimize this data loss and downtime.
- The simulator may not simulate all real-world conditions fully. Documenting these limitations will enable the users to spot what they cannot do.

- Incorrect inputs can lead to unrealistic results. The users shall be able to address this by using the parameters they need for predictable results.
- Complex simulators may cause crashes. The user will be able to slowly scale up the simulations gradually to monitor the performance that best fits their system's specs.

2.6. Test Plan

Table 1: Test Plan

ID	Test	Status[1]	Date	Notes
TC-1	Verify parameter input interface for simulations			
TC-2	Confirm simulation rendering for both high- and low-fidelity modules.			
TC-3	Validate parallel execution of low-fidelity simulations			
TC-4	Validate compatibility of dependencies with Ubuntu 22.04 Desktop.			
TC-5	Test the MAVLink-based communication between Gazebo, ArduPilot, and ROS2 for high-fidelity simulation.			
TC-6	Ensure consistent data transfer between the high- and low-fidelity subsystems through the Simulation Controller.			
TC-7	Verify the simulation supports realistic quadcopter drone behaviors.			
TC-8	Confirm that obstacles influence navigation and response behaviors of agents in the simulation.			
TC-9	Verify that drones execute collision avoidance maneuvers when an obstacle enters their predefined violation radius.			
TC-10	Verify that agents collaboratively adjust flight paths in crowded airspaces to maintain separation.			
TC-11	Ensure that drones follow predefined waypoints during simulation.			
TC-12	Test the validation of input parameters and produce error messages upon an invalid input.			
TC-13	Verify the interactive button-based commands trigger specific simulation actions.			

[1] Unwritten, Incomplete, Pass, Fail

TC-14	Test the system's functionality execution upon button clicks.			
TC-15	Verify that simulation environment settings dynamically update based on parameters provided.			
TC-16	Validate that low-fidelity scenarios can be converted into high-fidelity simulations.			
TC-17	Ensure real-time telemetry and visualization of UAS flight paths in the high-fidelity module.			
TC-18	Confirm that LiDAR data is used for obstacle detection and flight dynamics in Gazebo.			
TC-19	Test that simulation data, including telemetry and parameters, is logged after each simulation.			
TC-20	Verify the system performs grid search testing of all parameter combinations.			
TC-21	Ensure the simulation supports multiple drones operating in the same environment.			
TC-22	Validate interaction simulations between drones and various obstacle types, including buildings and trees.			
TC-23	Test the system's ability to adjust drone autonomy levels from manual to fully autonomous.			
TC-24	Ensure the system runs 100+ low-fidelity simulations in parallel efficiently.			
TC-25	Ensure that the simulation can turn off the visuals for computational efficiency.			
TC-26	Validate that errors are logged and handled without compromising the entire simulation.			
TC-27	Ensure all logged data is saved as a CSV after each simulation.			

3. Test Schedule

Currently, there is no schedule for testing.

4. Traceability Matrix and Defect Tracking

4.1. Traceability Matrix

<< A list of requirements and their corresponding test cases >>

Table 2: Traceability Matrix

Req. ID	Test Case(s)
SRS-1	
SRS-2	TC-1
SRS-3	TC-2
SRS-4	TC-3
SRS-5	
SRS-6	TC-5
SRS-7	TC-7
SRS-8	
SRS-9	TC-4
SRS-10	TC-5
SRS-11	TC-19
SRS-12	TC-6
SRS-13	TC-7
SRS-14	TC-8
SRS-15	
SRS-16	TC-9
SRS-17	TC-10
SRS-18	
SRS-19	TC-11
SRS-20	
SRS-21	TC-12
SRS-22	
SRS-23	TC-13
SRS-24	TC-14
SRS-25	TC-12
SRS-26	TC-15
SRS-27	TC-16
SRS-28	TC-17
SRS-29	TC-18
SRS-30	TC-19
SRS-31	TC-20
SRS-32	TC-21
SRS-33	TC-22
SRS-34	TC-23
SRS-35	
SRS-36	TC-24
SRS-37	TC-25
SRS-38	TC-26
SRS-39	TC-27

SRS-40	
SRS-41	
SRS-42	
SRS-43	
SRS-44	
SRS-45	TC-5
SRS-46	

4.2. Defect Severity Definitions

Critical	The defect causes a catastrophic or severe error that results in major problems and the functionality is rendered unavailable to the user. A manual procedure is impossible to be implemented or high effort is required to remedy the defect. Examples of critical defects are as follows: • System abnormally terminates • Data cannot flow through a business function/lifecycle • Data is corrupted or cannot post to the database
Medium	The defect does not seriously impair system function can be categorized as a medium defect. A manual procedure requiring medium effort can be implemented to remedy the defect. Examples of medium defects are as follows: • Form navigation is incorrect • Field labels are not consistent with global terminology
Low	The defect is cosmetic or has little to no impact on system functionality. A manual procedure requiring low effort can be implemented to remedy the defect. Examples of low defects are as follows: • Repositioning of fields on screens • Text font on reports is incorrect

5. Test Cases

<< Describe each test as shown below. >>

5.1. Test Case TC-1

Objective: Ensure the parameter input interface allows users to retrieve all required simulation parameters.

Notes: This test verifies that users can enter and validate all necessary parameters for the simulation.

Test No.: TC-1		Current Status: << Passed / Failed / Pending >>		
Test title: Verify parameter input interface for simulations				
Testing approach: Manual testing by interacting with the input interface to validate fields for each maneuver type.				
STEP	**OPERATOR ACTION**	**PURPOSE**	**EXEPCTED RESULTS**	**COMMENTS**
1	Open the simulation interface.	To view the parameter fields.	Simulation interface portrays all parameter fields and a drop down consisting maneuver types.	
2	Check for all parameter input fields for each maneuver type.	To ensure that all fields are accurate for all maneuver types.	All maneuvers have specific parameters listed on the SRS.	
Concluding Remarks: << Filled in by the person who completed the test >>				
Testing Team: << List members of testing team and lead >>		Date Completed:		

5.2. Test Case TC-2

Objective: Validate that simulations can be displayed graphically using Julia and in 3D using Gazebo.

Notes: This test will confirm graphical rendering using Julia and 3D visualization in Gazebo.

Test No.: TC-2			Current Status: << Passed / Failed / Pending >>	
Test title: Confirm simulation rendering for both high- and low-fidelity modules.				
Testing approach: Run the simulation in both Julia and Gazebo to verify correct graphical output.				
STEP	**OPERATOR ACTION**	**PURPOSE**	**EXEPCTED RESULTS**	**COMMENTS**
1	Run the simulation using Julia to display the graphical output.	To check if Julia properly renders the simulation output.	The simulation should render without graphical errors in Julia.	
2	Run the simulation in Gazebo and verify 3D rendering.	To confirm Gazebo renders the simulation in 3D accurately.	The simulation should be displayed in 3D with no errors in Gazebo.	
Concluding Remarks: << Filled in by the person who completed the test >>				
Testing Team: << List members of testing team and lead >>			Date Completed:	

5.3. Test Case TC-3

Objective: Confirm that the low-fidelity simulations can be ran in parallel for rapid testing.

Notes: This test ensures that the low-fidelity simulation framework can process multiple simulations at once.

Test No.: TC-3			Current Status: << Passed / Failed / Pending >>	
Test title: Validate parallel execution of low-fidelity simulations.				
Testing approach: Run Julia commands to run low-fidelity simulations in parallel.				
STEP	**OPERATOR ACTION**	**PURPOSE**	**EXEPCTED RESULTS**	**COMMENTS**

1	Run known parameters to the Simulation Controller.	To have predefined results and prevent errors and changes in simulation duration.	The system should display the running of 1000+ scenarios at once with a progress bar showing, and move on to the next type of simulations.	
2	Monitor system performance and check for any errors.	To ensure the system handles parallel processing effectively.	No system crashes or simulation types running for over 15 minutes should occur.	
Concluding Remarks: << Filled in by the person who completed the test >>				
Testing Team: << List members of testing team and lead >>		Date Completed:		

5.4. Test Case TC-4

Objective: Confirm the system operates on Ubuntu 22.04 Desktop and supports all required software dependencies.

Notes: This test ensures compatibility with Ubuntu 22.04 Desktop and verifies that all necessary software dependencies are installed. Estimated test duration is 45 minutes.

Test No.: TC-4		Current Status: << Passed / Failed / Pending >>		
Test title: Validate compatibility of dependencies with Ubuntu 22.04 Desktop.				
Testing approach: Run Julia commands to run low-fidelity simulations in parallel.				
STEP	**OPERATOR ACTION**	**PURPOSE**	**EXEPCTED RESULTS**	**COMMENTS**
1	Install all required software dependencies on Ubuntu 22.04.	To confirm the compatibility of the system with Ubuntu 22.04.	All software dependencies should install without errors.	

2	Run the system on Ubuntu 22.04.	To verify that the system functions correctly on the operating system.	The system should operate without any issues.	
Concluding Remarks: << Filled in by the person who completed the test >>				
Testing Team: << List members of testing team and lead >>			Date Completed:	