10.0 Unit Test & Integration Plan

10.1 Unit Test Plan

The purpose of this section is to both list and briefly describe the objective and contents of the tests our team plans to implement into our software.

10.1.1 Unit Test Descriptions

10.1.1.1 Unit Test 1 (Telemetry):

This test shall record numerous pieces of data that relate to spatial positioning using position vectors in three-dimensional space. Due to the difficulty of accessing the necessary API outside the SCS Software application, testing will largely be conducted via inspection of the output after a test session (a form of White Box testing). As a result, the distinction between unit and integration testing for this module is blurred.

10.1.1.2 Unit Test 2 (Test and Training Scenarios):

These tests shall demonstrate that both the test and training modules function as expected when provided with the correct parameters as input.

10.1.1.3 Unit Test 3 (Experiment Scheduler):

These tests shall ensure the functionality, integrity. and quality of the individual functions that comprise the user interface.

10.2 Integration Test Plan

10.2.1 Integration Test Descriptions

10.2.1.1 Integration Test 1 (Telemetry):

This test shall record numerous pieces of data that relate to spatial positioning using position vectors in three-dimensional space. Due to the difficulty of accessing the necessary API outside the SCS Software application, testing will largely be conducted via inspection of the output after a test session (a form of White Box testing). As a result, the distinction between unit and integration testing for this module is blurred.

10.2.1.2 Integration Test 2 (Test and Training Scenarios):

This section includes formalized guidelines for both the supervisor and user; which will provide a detailed description of how the experiment shall be conducted.

10.2.1.3 Integration Test 3 (Experiment Scheduler):

This test shall demonstrate both that the application properly records and stores each user's personal data and that it launches the expected sessions given correct inputs. As a GUI, comprehensive integration testing will be somewhat difficult. The bulk of integration testing, therefore, will likely have to take the form of Black Box user testing on the compiled product.

10.3 Module Dependencies

No warranty or guarantee is offered for nested or recursive dependencies of any external products and services, as external products may have additional dependencies that could change during the product lifecycle and/or vary from system-to-system.

10.3.1 Telemetry Subsystem:

Telemetry functionality requires, at a minimum, the following dependencies, roughly arranged in decreasing order of scope.

10.3.1.1 Development Dependencies

- Microsoft development stack
  - Windows 10
  - VisualStudio 2017 (or most recent)
  - Visual C++ 9.0 or higher, C standard numeric types (e.g., `stdint`) libraries
- SCS Software SDK v.1.9 or higher

10.3.1.2 Runtime Dependencies

- SimGear simulator hardware
- Microsoft Windows 10
- Sim Commander motion control software (by SimXperience)
- American Truck Simulator v.1.28 or higher (by SCS Software)

10.3.1.3 Test Dependencies

As the telemetry code is implemented as a DLL that can only be executed within the framework of the SCS Software API, not only must it be compiled and integrated with the application before it can be tested, but the Test and Training Scenarios should be integrated as well.

10.3.2 Test and Training Scenarios:

As development of these 2 packages contains a sufficiently substantial amount of overlap to be developed in parallel, they are here considered together. They require, at a minimum, the following dependencies, roughly arranged in decreasing order of scope.

10.3.2.1 Development Dependencies

- Everything included under section 10.3.1.1, as well as
- C standard library (`stdlib`), standard I/O (`stdio`)
- American Truck Simulator v.1.28 or higher (by SCS Software) and its Map Editor accessory

10.3.2.2 Runtime Dependencies

- Everything included under section 10.3.1.2
- Project Cars 2 (by Slightly Mad Studios)

10.3.2.3 Test Dependencies

Currently, there are no dependencies with respect to the order in which these tests should be run.

10.3.3 Experiment Scheduler:

While the client GUI is to be implemented as a Windows desktop app, implementation details are subject to change as the project progresses.

10.3.3.1 Development Dependencies

- Everything included under section 10.3.2.1
- Project Cars 2 (by Slightly Mad Studios)
- Unity 2018

10.3.3.2 Runtime Dependencies

- Everything included under section 10.3.2.2

10.3.3.3 Testing Dependencies

As a GUI that interacts with other applications, proper testing of the system requires all of the other dependent applications to be installed and running properly. Thus, while the tests can be run in relative isolation, the other tests should likely be run first.

10.3.4 Testing Dependency Diagram