UF UNIVERSITY of FLORIDA
Herbert Wertheim College of Engineering

Computer Engineering
CEN3908C

T1: Alpha Test Plan

Overview

The ***alpha test plan*** is a formal testing document teams will submit with the ***alpha build***. This plan will be used by the team to test the ***alpha build*** in the next phase of development; lessons learned will be applied in developing the ***beta build***. The team will be responsible for performing and documenting the results of the tests. The plan must be unambiguous, with sufficient clarity that it could be used by those outside of the team. Thus, it should be well-defined, organized, reproducible, and quantifiable.

Specification

The specifics of the test procedure will vary based on the content of the project but be specific enough that a technical reader could reproduce your tests and results without ambiguity. All plans must include these sections:

1. **Expected Behavior**: use tools such as flow charts, decision trees, source-code documentation, and written charts to define the expected behavior of your software and hardware.

2. **Test Procedures**: clear outline of tests to be performed to verify (or find flaws in) all expected behaviors. Unit test frameworks, electrical stress testing, and similar approaches be used to quantify performance.

   In future submissions, students will submit test results reporting testing findings and will conduct user testing based on the procedures outlined by the test plan.

Examples

Here are some examples of testable parameters and appropriate test methods:

Hardware	Software
• Generate a Bode Plot for a filter that confirms the cutoff frequency is as designed. • Create frequency and jitter specifications for timer-driven events, then use a logic analyzer to verify they are met. • Use firmware logging capabilities (e.g., SD card or UART logging) for microcontroller systems. • Measure an average power draw for battery powered devices as the device runs. Use this information to determine if the battery capacity is sufficient. Measure battery charge as device operates and extrapolate to estimate battery life.	• Test all functions to ensure proper return values for all acceptable parameters. • Develop automated unit test files and build systems. • Log user input and other events for debugging. • Use Docstrings to define expected behaviors and generate documentation. • Generate flowcharts that define user interface states and how to reach them. • Provide a decision tree for user interaction that allows for systematic evaluation if interfaces. • Define timeouts for asynchronous events, include timeout exception handling and logging in code.

Submission

Submissions will be on Canvas in the form of a PDF file containing the ***alpha test plan***.