CASE TOOLS

TEST SUITE DESIGN

Akhil Jain and Shubham Jain

Table of Contents

1. Introduction

This document is a high-level overview defining testing strategy for the Computer Aided Software Engineering Tool. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. Testing criteria under the white box, black box, and system-testing paradigm will be utilized. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation will be applied.

2. Test Objective

The objective of this test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, a broad range of tests will be exercised to achieve the goal. There will be many functions that can be performed by this application. The user interface to utilize those functions is designed to be user-friendly and provide easy access to all the functions.

3. Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Make specific test data for specific test cases.
4. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
5. Identify the expected results for each test.
6. Document the test case configuration, test data, and expected results.
7. Perform the test(s).
8. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
9. Successful unit testing is required before the unit is eligible for component integration/system testing.
10. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

11. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

4. Testing Process

The diagram above outlines the Test Process approach that will be followed.

1. Organize Project involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

2. Design/Build System Test involves identifying Test Cycles, Test Cases, , Expected Results, etc. Test Cases and the Data required are identified. The Test conditions are derived from the Software Requirement Specifications.

3. Design/Build Test Procedures includes setting up procedures such as Error Management systems and Status reporting.

4. Build Test Environment includes requesting/building hardware, software and data set-ups.

5. Execute System Tests – The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

5. Testing Strategy

Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the program's correctness. Some of these functions are:

Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Error guessing and Boundary Value Analysis testing on our application.

Below are major features tested:

- New
- Open
- Save
- Print
- View Data Dictionary
- Update Data Dictionary
- Check for Balancing Errors

- Go one level up in Hierarchy
- Create,Edit,delete shape and arrows.
- Create Hierarchy, Edit hierarchy,delete hierarchy
- Assign Bubble to a module or vice versa

White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test cases are generated that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once.

Integration Testing

Module - Graphic User Interface (GUI) Module

This module provides a simple GUI where the user can perform the different actions (functions). This module was tested separate from the backend to check if interface is functioning properly, and in general, to test if the mouse-event actions were working properly. The testing was performed by testing various functionalities.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case we focused only on function validation and performance. And in both cases we used the black-box method of testing.

Function Validation Testing

The integrated “SAS” was tested based on the requirements to ensure that we built the right application. In doing this test, we tried to find the errors in the inputs and outputs, that is, we tested each function to ensure that it properly implements the parsing procedures, and that the results are expected. The behavior of each function are contained in the Software Requirement Specification.

The interfaces to ensure they are functioning as desired (i.e. check if each interface is behaving as expected, specifically verifying the appropriate action is associated with each mouse_click event).

The interaction between the GUI and the backend controller classes. In this case the data will be inserted and check if the expected results are obtained.

Performance testing

This test was conducted to evaluate the fulfillment of a system with specified performance requirements. It was done using black-box testing method. Following things were tested:

Features not to be tested

We used Graphics2D library for drawing various shapes.
So there is no need to test these libraries separately as they are thoroughly tested and multiply used by many users.