Testing Document

Group: Canvas

Software Engineering Project
Department of Computer Science
University of Helsinki

02/09/2005
Document version: Final

Project Members:
Duku-Kaakyire Michael,
Karppinen Tony Henrik,
Lamsal Pragya,
Välimäki Niko Petteri

Instructor:
Raine Kauppinen

Customer:
Inkeri Verkamo

Supervisor:
Juha Taina

Table of Contents

Change log:

Date	Name	Description
2005-08-11	Niko	Updated document style and contents
2005-08-31	Pragya	Added test cases and test results
2005-09-02	Pragya	Final Version

1. Introduction

1.1. Overview of the system

This is a project for design and development of a diagram-drawing software under the course named Software Engineering Project at the department of Computer Science at University of Helsinki.

The aim of this project is to develop a functional application that can realize the core requirements of the customer, while leaving room for extension of the application in the future by other developers.

1.2 Purpose of this document

This document is to serve as the skeleton test design / test plan for the software engineering project Canvas.

Preparation for this test consists of three major stages:

* The Test Approach sets the scope of system testing, the overall strategy to be adopted, the activities to be completed, the general resources required and the methods and processes to be used to test the release. It also details the activities, dependencies and effort required to conduct the System Test.

* Test Planning details the activities, dependencies and effort required to conduct the System Test.

* Test Conditions/Cases documents the tests to be applied, the data to be processed, the testing coverage and the expected results.

1.3. Objectives of System Testing

At a high level, this system testing intends to prove that :

* The functionality is as specified by Design Specification Document and the Requirements Documentation.

* The software is of high quality; the software will support the intended functions and achieves the standards required by the customer for the development of new systems.

* Test harnesses(Automation of the testing process)
    * Test procedures.

* Test scripts.
* Test suites of test cases.
* Test data.

* maximize the effectiveness of testing
* Maximize the probability of tests causing failures that were previossly unidentified.
* Minimize the number of test suites and test cases.

The following are the preliminary steps we propose to follow

I - Identification of test suites

Identification of test suites using requirement documentation. The test suites identified so far are based on the functional and performance requirements specification. They are suites on
a.) Drawing
b.) Editing
c.) Execution
d.) Usability

II - Determination of the objectives of each test suite.

The objectives of each work suite outlined above is to test to uncover defects, identify and correct them.

III - Development of test procedures
-Coming at a later stage in project

IV - Determination of the test cases necessary to meet the objectives of the test suite. The test cases would be based on the use cases outlined in the requirement document

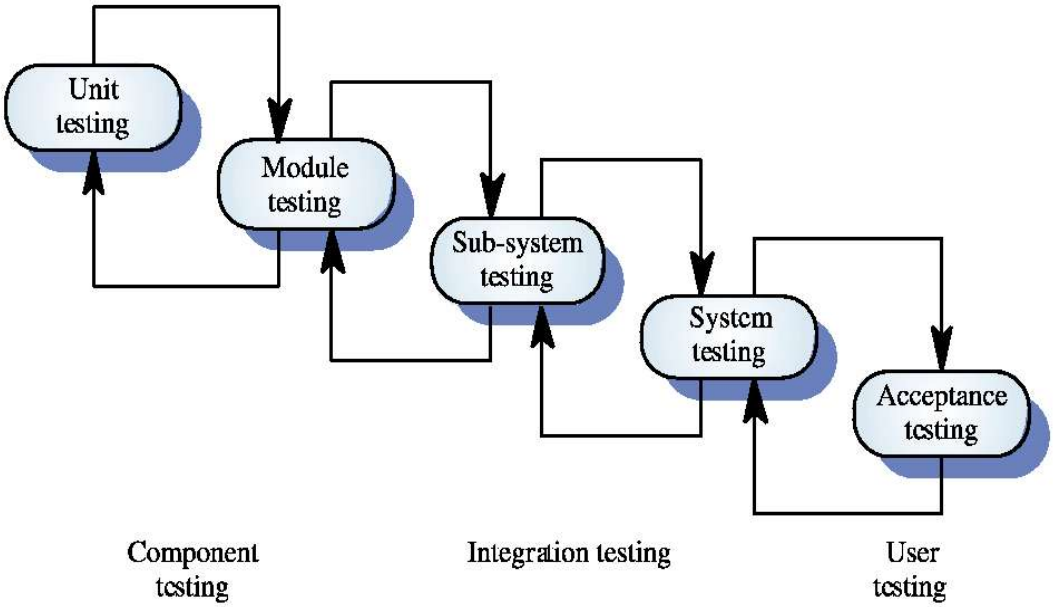
V - For each test case, determination and documentation in the associated test case the:
* Test preconditions
* Test steps
* Test postconditions
* Test schedule (for system tests)
* Required resources (for system tests. May require expansion of Gantt chart)

2. Scope and objectives

2.1. Testing Process

Below is a diagram of the testing process to be used starting from the smallest a module to the whole system and finally acceptance testing by the customer. The details are as

outlined in the next section.



2.2. Testing Scope

Outlined below are the main test types that will be performed for this release. All system test plans and conditions will be developed from the functional specification and the requirements catalogue.

2.2.1. Module testing

In this testing phase, testing is concerned with the testing of the smallest piece of software for which a separate specification exists.

2.2.2. Functional testing

This phase is build on test cases that are based on an analysis of the specification of the component without reference to its internal workings.

2.2.3. Integration testing

Goal for this testing phase is to expose faults in the interfaces and in the interaction between integrated components.

2.2.4. System testing

The process of testing an integrated system to verify that it meets specified requirements.

2.2.5. User acceptance testing

This test, which is planned and executed by the client, ensures that the system operates in the manner expected, and any supporting material such as procedures, forms etc. are accurate and suitable for the purpose intended. It is high level testing, ensuring that there are no gaps in functionality.

2.3. System Test Entrance/Exit Criteria

All high priority errors from system test must be fixes and tested before the software can be accepted.

3. Test Cases

3.1 System Testing

Test cases for system testing

Application

- Close the application from menu and close icon on title bar
- Resize the application
- Minimize/maximize the application
- Test the menu structure
- Close/minimize/maximize the internal frames
- Close/minimize/maximize the toolbar

Result: Application testing worked fine without any problem except for the visibility of the toolbar. Once the toolbar is manually closed, it could not be opened from the menu bar.

Fix: Problem with the toolbar fixed. Now it cannot be closed. It can only be made visible or invisible from the menubar.

Open a diagram

- Open a diagram using the shortkey for Open
- Open a diagram using the menu structure
- Open a large number of diagrams
- Exceptions
    - Open a diagram using illegal path and/or filename
        - Expected result is an error message
    - Open a file which is not a Canvas diagram file
        - Expected result is an error message
    - Open a diagram which is already open
        - Expected result is that the document is set active

Result: No error messages were shown in the case of exception. If an open diagram is tried to open again, it opens another instance of the same diagram. Others worked fine.

Save a diagram

- Save a diagram using the shortkey for Save
- Save a diagram using the menu structure
- Save a diagram with a large number of elements
- Save an empty diagram
- Save a diagram with a default name (new document with no given name)
- Exceptions
    - Save a diagram using illegal path and/or filename
        - Expected result is an error message
    - Save a diagram with a existing filename
        - Expected result is confirmation for overwriting

Result: No confirmation message was shown for overwriting an existing file and no error message for using illegal path and filename. Others worked fine.

Close a diagram

- Close a diagram with no changes
- Close a diagram with changes
    - Expected result is confirmation for closing
- Close the application using the "quit" icon on the title bar
- Close the application using the menu structure
- Close diagrams by closing the application with no changes in diagrams
- Close diagrams by closing the application with changes in diagrams

- Expected result is confirmation for closing the application
  and the reason for showing the confirmation
- Kill an application

Result: No confirmation message was shown for closing the modified diagrams without saving. However, it saves the modifications.

Insert a new element into a diagram

- Select all implemented elements one by one and insert those into a diagram
- Insert an element into a new diagram
- Insert an element into a diagram restrored for a file
- Insert an element with trying to set the ending point outside of the internal frame of diagram
  - Expected result is that user can not set the ending point outside the diagram frame (better yet, canvas will scroll when cursor hits the border of the current view port)
- Insert an element with the same starting and ending point
- Insert an element on top of another element
- Insert a large number of elements on the diagram

Result: No scrolling. Figure that goes outside the border cannot be seen but still exists. And, when an element is inserted with the same starting and end point, nothing is shown.

Editing an element

- Select an element and move it
- Select an element and move it outside the internal frame
  - Expected result is that the view port scrolls as the element is moved on the canvas
- Move multiple elements as above
- Insert text into an element
- Change color for element(s)
- Resize element(s)

Result: Moving an element outside the frame makes it disappear. No scrolling implemented. Multiple elements can be moved at the same time either by selecting all of them or grouping them.

Copy, Cut and Paste

- Cut an element which is connected to another element
- Cut and paste a group of elements
- Copy and paste text
- Cut/Copy text outside the application and paste it into an element

- Cut/Copy an element and paste it into another diagram
- Cut/Copy an element and paste it into the same diagram
- Paste a large number of elements in one diagram
- Copy object outside the application and paste in application
    - Expected result is that foreign objects can and should not be pasted to the diagram
- Paste without copying or cutting anything first
    - Expected result is that nothing happens
- Paste with no diagrams open or selected
    - Expected result is that nothing happens

Result: Copying text from outside the application does not work. Others work as expected.

Include text into the diagram

- Include text into the diagram itself

Result: Works as expected.

Define a new element by composition

- Define a new element selecting just one element
    - Expected result : should work like with several elements
- Define a new element selecting multiple elements
- Define same element twice or more
    - Expected result : should be possible even if not smart
- Define a new element with no selected elements
    - Expected result : It is not possible to select this function when no elemets are selected

Result: Defining a new element selecting just one element does not work with the built-in elements but only with user-defined elements.

Help

- Read the help
- Close/minimize/maximize help window

Result: Works as expected.

4. Test schedule

Test phase	Begin week	End week
Preparing test plan	25	28
Identifying test data	25	23

Test phase	Begin week	End week
Module testing	29	33
Functional testing	30	33
Integration testing	32	34
System testing	35	35
Testing documentation	24	35

5. Resources

5.1. Human

Title	No.	Date Req'd	Status
Test Controller	1	Week 23 – Week 35	Assigned
Testers	3	Week 29 - Week 35	Assigned

5.2. Hardware

Department of Computer Science will provide complete environment including hardware required for all phases of testing. The specifications for PC workstations are met when using the hardware the university provides.

5.3. Software

Test environment software

Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0-b64) Eclipse

6. Roles and responsibilities

Test Controller
- Ensure Phase 1 is delivered to schedule, budget & quality

- Produce High Level and Detailed Test Conditions
- Produce Expected Results
- Report progress at regular status reporting meetings
- Co-ordinate review & signoff of Test Conditions
- Manage individual test cycles & resolve tester queries/problems.
- Ensure Entrance criteria are achieved prior to System Test start.
- Ensure Exit criteria are achieved prior to System Test signoff.

Testers
- Identify Test Data
- Execute Test Conditions and Markoff results
- Raise Software Error Reports
- Administer Error Measurement System

7. Error management

During System Test, errors will be recorded as they are detected. The error review team will meet after each twice a week meeting to review and prioritise problems not yet solved. These will be assigned or dropped as appropriate. The team will consist of testers and test controller.

Errors, which are agreed as valid, will be categorised as follows by the error review Team :

Category	Description
A	Serious errors that prevent System test of a particular function continuing or serious data type error
B	Serious or missing data related errors that will not prevent implementation.
C	Minor errors that do not prevent or hinder functionality.

8. Status reporting

Test preparation and Testing progress will be formally reported during a weekly meeting. The whole project crew will attend these meetings. A status report will be prepared by the Test Controller to facilitate this meeting.

This report will contain the following information :

1. Current Status v. Plan (Ahead/Behind/On Schedule)
2. Progress of tasks planned for previous week

3. Tasks planned for next week including tasks carried from previous week
4. Error Statistics
5. Issues/Risks

9. Assumptions

9.1. Assumptions

- All "Show-Stopper" bugs receive immediate attention from the development team.
- All bugs found in a version of the software will be fixed and unit tested by the development team before the next version is released.
- Functionality is delivered to schedule.
- The automated test tool will function & interface correctly with the software.