Testing Plan

Group 3

June 8, 2022

Contents

1 Introduction **4**
1.1 Overview . . . 4

2 Test Plan **4**
2.1 Functional Tests . . . 4
2.1.1 Testing Format . . . 5
2.1.2 Considerations . . . 5
2.2 Unit Tests . . . 5
2.2.1 Testing Format . . . 5
2.2.2 Considerations . . . 6

3 Functional Tests **6**
3.1 UI . . . 6
3.1.1 UI Setting . . . 6
3.1.2 Element Display . . . 6
3.2 Engine . . . 7
3.2.1 File Handling . . . 7
3.2.2 Action Handling . . . 8
3.2.3 Scripting . . . 8
3.2.4 Remote Interface . . . 9

4 Unit Tests **10**
4.1 Core . . . 10
4.1.1 App . . . 10
4.1.2 Engine . . . 10
4.1.3 Tool . . . 10
4.1.4 Tools . . . 10
4.1.5 ToolsFactory . . . 10
4.2 Elements . . . 11
4.2.1 Elementsfactory . . . 11
4.2.2 DocElement . . . 11
4.2.3 ScriptElement . . . 11
4.2.4 VisualElement . . . 11
4.3 UI . . . 11
4.3.1 MainController . . . 11
4.3.2 SizeObj . . . 12
4.4 XML I/O . . . 12
4.4.1 Ingestion . . . 12

5 Test Records **12**
5.1 Information . . . 12
5.2 Reports . . . 12

Revision History

Revision	Date	Author(s)	Description
0.1.0	14.03.22	SSP526	Doc created in GDocs
0.1.1	14.03.22	DM1306	Fit to LaTeXtemplate
0.2.0	14.03.22	DM1306	Add further information to all sections. Refocus on product goals. Change overall test strategy to bottom-up approach.
0.3.0	15.03.22	KYC527	Add Unit Test Section, re-organise sections and expend Unit Tests in Test Plan section

1 Introduction

1.1 Overview

This document describes our Testing Methodology that will be used through the development cycle of our product. It will define Functional Tests for User Stories as found in the Functional Specification document, and a broad overview of our methodology for generating and applying automated Unit Tests.

2 Test Plan

2.1 Functional Tests

At the highest level, our codebase can be split into two super-modules: The UI Controller and the Engine. This is a practical distinction, with the two running on seperate threads to prevent heavy processing blocking the UI; the two super-modules may be divided further into several modules each.
The UI Controller may be seen to minimally consist of:

- Menu Bar
- Tool List
- Element Properties Display
- Message Display
- 2D Graphics Display (Including Text and Tables)
- Image Display
- Video Display
- Audio Player Display

The Engine may be seen to minimally consist of:

- Document Parse
- Tool Parse
- Document Output
- Event Handling
- Tool Handling
- Script Engine
- Remote Interface

2.1.1 Testing Format

1. Refer to the appropriate Test if available.
2. Ensure that all testable dependancies have passed Unit Testing.
3. Build the software with the module to-be-tested included.
4. Enter the required input for the Test.
5. Compare the expected outcome with the actual outcome.
6. Record the result.

2.1.2 Considerations

The above modules may even be split several times further into their component parts; we shall start our testing with these, employing the common JUnit Test framework to run localised, automated Unit Tests ensuring that we have confidence in these parts as they become available, prior to further high-level Functional testing. In a word, our strategy is "bottom-up".

2.2 Unit Tests

Automated Unit Testing shall be applied to "lower-level" modules. Every public method on an Object should be tested for correctness of operation through the Unit Test suite using a combination of random, invalid, and valid inputs and a combination standard Unit Testing and "fuzzing" techniques. Unit tests should be implemented before the functional tests in order to eliminate chance of operation relies on faulty method(s).
JUnit And JaCoCo are to be used for preforming such technique, with the former verifies the functionality of individual method and the latter proves the effectiveness of the test code. Using such tools minimise the development effort and more importantly possible human errors which contribute to more reliable testing. As part of the QA of the unit testing, code coverage must reach at least 70% for the test code to be recognised.

2.2.1 Testing Format

1. Refer to the appropriate Test if available.
2. Define and code test fixture of JUnit Test Suites.
3. Build the software with the module to-be-tested include.
4. Run JUnit Test Runner.
5. Check console for failure flags.
6. Check console for test code coverage with JaCoCo.
7. Record the result.

2.2.2 Considerations

"Lower-level" modules have impacts on those above, and so flawed Unit Testing has the potential to invalidate further Functional Testing. This means that our Unit Test suite must be near-complete with high measured code coverage, to provide confidence in our semi-automatic and manual Functional Tests.

3 Functional Tests

3.1 UI

3.1.1 UI Setting

Menu Bar

Description	Post valid and invalid menu bar operations to the UI for display.
Purpose	The User requires access to operations from the menu bar. Test this function.
Inputs	Selection of menu bar items.
Expected Outcome	Available operations are displayed in the menu bars.

Tool List

Description	Post valid and invalid tools to the UI for display.
Purpose	The User requires access to tools from the Tools menu. Test this function.
Inputs	Selection of Tools.
Expected Outcome	Available tools are displayed in the Tools menu.

Element Properties Display

Description	Click on a visual element to show its' properties. Click off to hide them.
Purpose	The User shall select a visual element, which should reveal its' properties in the Properties menu. Test this function.
Inputs	Mouse clicks.
Expected Outcome	An object's properties are displayed in the Properties menu on click on the object.

3.1.2 Element Display

Message Display

Description	Post individual blocking and non-blocking messages to the UI for display.
Purpose	Users require notification about certain events within the program. Test this function.
Inputs	Blocking message (Action-required message). Non-blocking message (Information message).
Expected Outcome	Messages containing the input text of the correct type shall be displayed.

2D Graphics Display

Description	Post valid and invalid 2D graphics objects to the UI for display.
Purpose	The User may require that a certain 2D Graphical element be displayed. Test this function.
Inputs	Random 2D graphical object.
Expected Outcome	Valid objects should be displayed correctly. Invalid objects should not be displayed.

Image Display

Description	Post valid and invalid images to the UI for display.
Purpose	The User may require that a certain image be displayed. Test this function.
Inputs	Random images.
Expected Outcome	Valid images should be displayed correctly. Invalid images should not be displayed.

Video Display

Description	Post valid and invalid videos to the UI for display.
Purpose	The User may require that a certain video be displayed. Test this function.
Inputs	Random images.
Expected Outcome	Valid video should be displayed correctly. Invalid video should not be displayed.

3.2 Engine

3.2.1 File Handling

Document Parse

Description	Post valid and invalid documents to the engine to parse.
Purpose	The User shall open a document, and the engine will attempt to parse it. Test this function.
Inputs	Valid and Invalid presentation XML documents.
Expected Outcome	The parsed result of a valid document is returned. Invalid documents return nothing.

Tool Parse

Description	Post valid and invalid tool documents to the engine to parse.
Purpose	The User shall open the application which shall attempt to load a tool file. Test this function.
Inputs	Valid and Invalid tool documents.
Expected Outcome	Valid documents are correctly parsed. Invalid documents return nothing.

Document Output

Description	Try to save a presentation Stack Document.
Purpose	The User shall edit a document and then save it. Test this function.
Inputs	Graphically edited document.
Expected Outcome	Valid XML document is written to the User's specified location.

3.2.2 Action Handling

Event Handling

Description	Post valid and invalid events to the engine.
Purpose	The User shall perform an action and the engine should respond. Test this function.
Inputs	User actions.
Expected Outcome	Correct response to a valid event. No response to an invalid event.

Tool Handling

Description	Post valid and invalid actions for tools.
Purpose	The User shall select a tool from the UI and use it, triggering an action. Test this function.
Inputs	User tool input.
Expected Outcome	Tool handler is run correctly, posting the required items to the UI.

3.2.3 Scripting

Script Engine

Description	Ensure that the Script Engine can execute scripts and with correct access to program data.
Purpose	The User shall trigger an event associated with a script, which should execute as expected. Test this function.
Inputs	Test Script.
Expected Outcome	Script executes correctly.

3.2.4 Remote Interface

Remote Interface Connect

Description	Connect remote interface to remote
Purpose	The User may connect to a presenter. The User may present and wish to connect to others. Test these functions.
Inputs	Connect requests.
Expected Outcome	Interface connects to remote.

Remote Interface Actions

Description	Post actions into the Remote Interface. Receive output actions from the Remote interface
Purpose	The User may connect to a presenter, and wish to receive actions. The User may present and wish to transmit actions. Test these functions.
Inputs	Test Actions.
Expected Outcome	Actions are received correctly in the engine. Actions are output correctly.

4 Unit Tests

Test Plan is sectioned in same format as project tree.

4.1 Core

4.1.1 App

start

Description	Post actions into the Remote Interface. Receive output actions from the Remote interface
Purpose	Method is used to set-up scene with the given stage variable. Test these functions.
Inputs	Test Stage object.
Test Assert	assertNull

stop

4.1.2 Engine

start

stop

allowDraw

offerEvent

offerNewDoc

run

4.1.3 Tool

getName

getID

getScript

4.1.4 Tools

getTools

4.1.5 ToolsFactory

startMakingElement

4.2 Elements

4.2.1 Elementsfactory

startMakingElement

4.2.2 DocElement

getPages

4.2.3 ScriptElement

getScriptText

4.2.4 VisualElement

getLoc

setLoc

getID

setID

getZInd

setZInd

getSize

getFillColour

4.3 UI

4.3.1 MainController

gracefulExit

drawText

configPage

clearPage

addTool

showNonBlockingMessage

initialize

4.3.2 SizeObj

getX

getY

getRot

4.4 XML I/O

4.4.1 Ingestion

parseDocXML

parseGenericXML

5 Test Records

5.1 Information

5.2 Reports