Milestone 3 : Testing Document

Matthew Pike, 523355@swansea.ac.uk
Supervisor: Dr Max Wilson

Swansea University
Computer Science Department

Contents

1 Introduction

The purpose of this document is to detail the testing process applied to the project and review the outcome of the procedure. This document will also validate that the final version of the project fully fulfils the requirements set out in the Requirements Document. This will be realised through acceptance testing.

This document should serve as a milestone in the project detailing the final version and recording the project's conformity to the requirements and specifications set out in Milestone 1.

Term Definition

As with any technical project, there exists numerous technical abbreviations that are used in place of long descriptors. We present a table below of the various abbreviations used in this document, which should serve as an aid to the reader.

Term	Definition
Visualiser	The part of the system responsible for displaying the data visualisation.
Document	The web site or web based document being investigated in the user study.
Experiment	The study that is being conducted. Typically will contain a User and a Conductor.
Data Source	A device/software/location that is recorded by the Web Browser.
Recorded Instance	Since a single experiment on a single user can contain numerous conditions and tasks, we must distinguish what one unique combination of these properties are called. We have chosen 'Recorded Instance' to represent this. A Recorded Instance is an identifier for: Experiment -> User -> Condition -> Task.
Stack	A Stack is a part of the Visualiser UI that represents a single Recorded Instance. A Stack is therefore a UI component.
WPF	Windows Presentation Foundation - The visual framework used to develop the user interface(s) in this project.

ORM	Object Relational Mapping - a way of linking entries in a database to a usable format in application code.
SQLCE	SQL Compact Edition, the embedded database product we use to store application data.

Personas

In addition to technical abbreviations, the project features unique individuals that partake or feature somehow in the system. Below is a table documenting these individuals and their role in the system. Again, this table is meant as an aid to the reader.

Individual	Role
Conductor	The person responsible for performing the user study. This is typically a researcher who is aiming to prove a particular hypothesis.
Researcher	The person who is responsible for gaining insight from the user study. The Researcher and Conductor can typically be the same person, however this is not always the case, especially in a large research group. The Researcher is therefore a member of the research team who is wishing to use the data collected from the study.
User	The person who is sitting the user study. Their responsibility is to perform tasks provided to them by the conductor.
Client	The person who will be receiving the finished software project. In this case it is Pingar.

2 Testing Approach

In this section we document the testing strategy employed during the development of the User Study System (USS). We will provide brief detail on the testing procedures employed for different aspects of the project and comment on the success of our approach.

For detailed analysis on the testing methodology, we refer you to the Methodology and Requirements document in Milestone 1. This section provides an insight to the applied testing that occurred during the development of the USS.

2.1 Overview

In this section we will provide a brief overview of the User Study System (USS) and the subsections that form it. For a detailed account of the USS's design, please refer to the Design Document. The intention for this section is to provide an overview of the components that form the USS and their interactions with one another in order to see how certain strategies were developed for testing the USS.

The USS was designed as a combination of two major subsystems; Browser and the Visualiser. However, the subsystem Browser, can be further broken into two separate components; Setup and Web Browser. This relationship is shown in Figure 2.1.

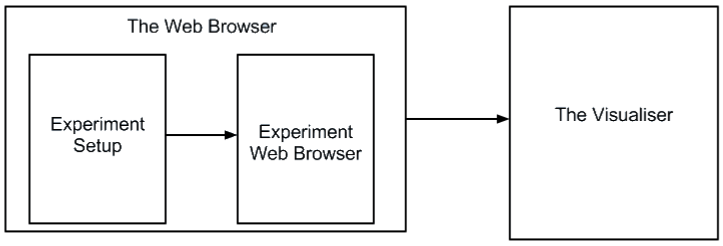

Figure 2.1: The Major subsystems and components that form the USS.

We see from Figure 2.1 that each component in the overall system is linearly dependant on the component before it. This gives the system a seemingly flat hierarchy. However, in terms of actual implementation, additional components are required in order for the system to operate correctly. As such, Figure 2.2 is an accurate representation of the major subsystems that exist in the USS.

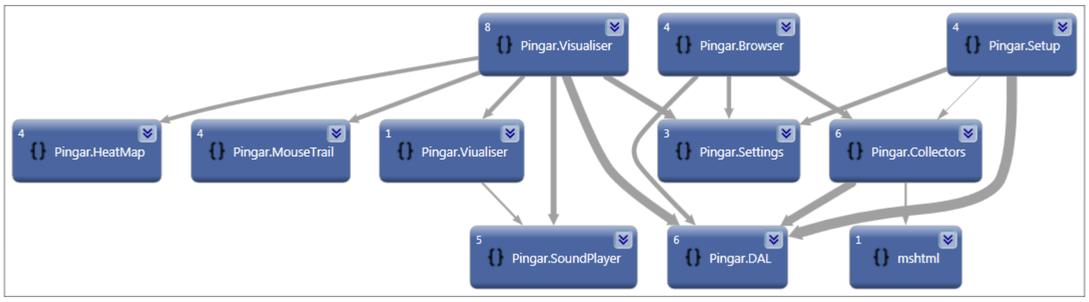

Figure 2.2: The components of the USS and their interaction/dependency with one another.

The tests developed for the USS were developed and executed within Visual Studio 2010 Ultimate Edition. We used the standard unit testing library provided with the IDE for unit tests. We also utilised the IDE for other forms of testing, such as performance testing. These tests are included with the project source code. We used the book "Professional application lifecycle management with Visual Studio 2010" by Gousset, M et al [GKKW10]. The book was of great assistance with the practical implementation of testing and also when and where testing should occur during the development of the system.

2.2 Testing Methods

In this section we will detail the testing method applied during the development of the User Study System (USS). We describe both the approach taken and the validation performed to ensure that the received results matched the expected value.

Our chosen testing methodology was complemented by our chosen development methodology - Prototyping. In some ways Prototyping is a form of testing, since it provides a test on the feasibility and relative success of a particular approach. This was however a positive side effect of our chosen methodology and acts only as the first wave of testing.

Our primary approach to testing was conducted through Unit Tests. Since the User Study System (USS) is a large, complex project, comprising of numerous smaller sub-systems, our testing was pre-dominantly local to each subsystem. To effectively describe the testing procedure, we have divided the proceeding sections into detailing the individual sub-systems, as they occur in the implementation of the USS itself. For an understanding of how these components form the larger system, refer to Figure 2.2 above.

2.2.1 Data Abstraction Layer (DAL)

The DAL is the component responsible for storing and retrieving all experiment information stored in the USS database. In testing this component, our interest was focussed primarily on assuring the consistency of the data stored in the system. Other factors such as performance were not formally tested since the application does not use the component intensively, rather it is used extensively by a number of components.

To confirm the consistency of the data being stored and retrieved, we developed a test project within the USS solution in Visual Studio. The project created a testing client which instantiates the DAL. From here the client performs consistency checks upon each of the entities exposed by the DAL. For each entity the consistency of CRUD operations is assured. A typical procedure follows these steps:

1. Add an instance of an entity to the DAL.
    a) Read that instance back from the DAL and assert it is equal to the original.
2. Modify the inserted entity with an updated one.
    a) Read that instance back from the DAL and assert it is equal to the updated entity.
3. Remove the inserted entity from the DAL.
    a) Attempt to read back the entity. Assert that the correct error message is returned.

We performed this procedure on each of the entities that the DAL publicly exposes to implementing classes.

To aid us in the testing of the DAL, we referred to the book **Programming Entity Framework** by Julia Larman [Ler10]. As well as using the book in the development of the DAL, the book proved to be an invaluable reference during the testing stages. Chapter 24 in the book is especially helpful, detailing exactly how one should approach testing a DAL. What was especially illuminating was Larman's recommendation of testing the individual entities directly, instead of attempting to test the entire context at once. This simplified the approach to testing and ultimately proved successful.

2.2.2 Setup

The Setup component is the part of the system where Conductors configure each user study. There was little testing required for this section as it was primarily just an interface to the data. The consistency of the data being stored is tested in the

DAL (documented above, subsection 2.2.1). We did investigate the possibility of performing automated UI testing on this component and other UI components, but ultimately decided against it due to time restrictions.

2.2.3 Browser

The Browser component and its associated sub-components (specifically the Collectors component) were subject to rigorous testing procedures. There were 2 forms of testing conducted:

Unit Tests These were used to ensure data consistency between the captured data and the data stored on disk. Since the Collectors component is responsible for collecting from a number of sources, we had to ensure that the synchronicity of these operations were not interfering with one another.

Performance Testing Again due to the massive synchronicity of the operations performed in this component, it was vital to ensure that the performance of the component was acceptable and did not interfere with the participant's web browsing.

For each of these tests we used "Software Testing with Visual Studio 2010" by Jeff Levinson [Lev11]. The book was especially useful for detailing performance testing applications using virtual users under a variety of different scenarios, outlined in chapters 6 & 7. These same chapters also outline the approach taken to automate the unit tests developed, to test the consistency of the data being collected versus the data being stored.

2.2.4 Visualiser

Similarly to the Setup component above, the Visualiser component deals mostly with displaying stored data. As such, the consistency of the data being stored is tested by the Browser component. However, additional tests were required to ensure that all components were synchronised with one another. This was performed through unit testing.

3 Acceptance Testing

In this section we aim to provide evidence of the system's conformity to the specifications set out in Milestone 1. We present this evidence in tabular format.

3.1 Setup & Web Browser

In this section we provide the acceptance testing for the initial part of the project, the web browser.

TEST_WB01

Asserts Specification(s)	WBSPEC1, WBSPEC2, WBSPEC3, WBSPEC4, WBSPEC5, WBSPEC20, WBSPEC21, WBSPEC22, WBSPEC23
Description	Tests whether the system allows conductors to specify participant, experiment, condition and task details. The system must also store the starting URL. The system should persist CRUD operations across system resets. The participant details should be stored in the system database.
Acceptance Criterion	• The system provides an intuitive user interface that allows the conductor to enter the necessary details. • The entered details should be validated. Validation errors should be displayed. • The system saves all valid input to the system database. • The stored data matches exactly to the entered data.

Result	**Pass**

TEST_WB02

Asserts Specification(s)	WBSPEC6, WBSPEC7, WBSPEC8, WBSPEC9, WBSPEC10
Description	The system must be capable of retrieving stored details (participants, experiments, conditions and tasks), and display them to the conductor in an intuitive interface. The conductor should be capable of selecting a distinct value for each detail.
Acceptance Criterion	• The system provides an intuitive user interface that allows the conductor to view the available details. • The displayed information matches the entered details. • The system allows for a single distinct value to be selected (e.g. Only one participant can be selected at a time) • The system stores this unique combination in the system database.
Result	**Pass**

TEST_WB03

Asserts Specification(s)	WBSPEC11, WBSPEC12, WBSPEC13, WBSPEC14, WBSPEC15, WBSPEC31, WBSPEC32 WBSPEC34, WBSPEC36, WBSPEC38, WBSPEC40

Description	The system should support the acquisition of data from the following sources: • Microphone • JavaScript based Web Events • The mouse • The web browser view (screenshots) • The Emotiv EPOC brain scanning device

Acceptance Criterion	• The system captures all mouse movement and clicks. • The mouse data is stored in a correctly formed XML format. • The system captures all JavaScript based events. • The web event data is stored in a correctly formed XML format. • The system captures all sound from the microphone. • The captured audio is stored in a correctly formed .WAV file format. • The system captures all screenshots at the specified interval. • The captured screenshot is stored in a correctly formed .BMP file format. • The system captures all brain based events. • The brain data is stored in a correctly formed XML format. • All captured information is stored in a standards compliant file format. • The recording process does not interfere with the participant's interaction with the web page they are currently viewing.
Result	**Pass**

TEST_WB04

Asserts Specification(s)	WBSPEC16, WBSPEC17, WBSPEC18, WBSPEC19,

Description	The system must allow the conductor to enable or disable the following collection sources prior to beginning the user study: • Microphone • JavaScript based Web Events • The mouse • The web browser view (screenshots) • The Emotiv EPOC brain scanning device
Acceptance Criterion	• The system provides a clear and intuitive interface for enabling or disabling these features. • The selection is stored to the system database and is enforced during the user study. • Collectors' settings persist application restarts.
Result	**Pass**

TEST_WB05

Asserts Specification(s)	WBSPEC24, WBSPEC25
Description	The setup utility must allow the conductor to configure the Emotiv EPOC and the audio input device, using the standard system utility for each.

Acceptance Criterion	• The interface features a shortcut that loads the Emotiv Control panel. • The interface features a shortcut that loads the windows input properties panel.
Result	**Pass**

TEST_WB06

Asserts Specification(s)	WBSPEC26, WBSPEC27, WBSPEC28, WBSPEC29, WBSPEC30, WBSPEC35
Description	The web browser must be familiar to a general user and must feature all the necessary capabilities that are expected from a modern web browser.
Acceptance Criterion	• The Web Browser has a back button which navigates to the previous page in history. • The Web Browser has a forward button which navigates to the next page in history. • The Web Browser has a go button which navigates to the specified page. • The Web Browser has an address bar which is modifiable. • The Web Browser renders standard compliant web pages correctly. • The layout of the Web Browser interface emulates Internet Explorer 6.
Result	**Pass**

TEST_WB07

Asserts Specification(s)	WBSPEC37, WBSPEC39
Description	The Setup utility and web browser must be well documented
Acceptance Criterion	• The source code is commented correctly and is supported by Doxygen documentation. • A detailed and understandable user manual is provided for non-technical users.
Result	**Pass**

TEST_WB08

Asserts Specification(s)	WBSPEC41, WBSPEC42
Description	The Setup utility and web browser should be extensible, allowing future collection sources and interface items to be added.
Acceptance Criterion	• The system provides public interface and hook methods to allow additional collection sources to be added. • The system provides interface pointers to the web browser's UI. • The system provides hook methods and interfaces to add additional document types to the system.
Result	**Fail**

Description	The test is at fault here. The original specification makes no reference to the document display being extensible, as such this functionality has not been implemented. We have revised this test in **TEST_WB09**.

TEST_WB09

Asserts Specification(s)	WBSPEC41, WBSPEC42
Description	The Setup utility and web browser should be extensible, allowing future collection sources and interface items to be added. (Revision of **TEST_WB08**)
Acceptance Criterion	• The system provides public interface and hook methods to allow additional collection sources to be added. • The system provides interface pointers to the web browser's UI.
Result	**Pass**

3.2 Visualiser

In this section we provide the acceptance testing for the final part of the project, the Visualiser.

TEST_VS01

Asserts Specification(s)	VSSPEC1
Description	The Visualiser must have a suitable hierarchy for the conductor to select a user study recording based on the study's settings (Experiment/Participant/Condition/Task)

Acceptance Criterion	• The interface features a hierarchical display for each recorded experiment in the system. • The interface must allow for hierarchical navigation of an experiment recording based on the features (Experiment/Participant/Condition/Task) • When selected, a recorded instance is loaded into the Visualiser. • The display transparently handles data unavailability from the system database without crashing the Visualiser.
Result	**Pass**

TEST_VS02

Asserts Specification(s)	VSSPEC2, VSSPEC3, VSSPEC14
Description	The Visualiser interface should present individual recordings in their own stack. The Visualiser should be capable of containing many stacks. The interface should be intuitive and allow for the comparison of multiple studies.
Acceptance Criterion	• A single recorded study is presented in a single interface pane. • The Visualiser interface allows for multiple panes to be contained and manipulated within the interface. • Each individual pane is resizeable. • It is possible to close a single pane, and the associated resources are freed appropriately.

Result	**Pass**

TEST_VS03

Asserts Specification(s)	VSSPEC4, VSSPEC5, VSSPEC6, VSSPEC7, VSSPEC8, VSSPEC9, VSSPEC1, VSSPEC12, VSSPEC15
Description	Each individual stack (display of a single recording) must display the following interface features: • Visualised emotion data from the Emotiv EPOC • Visualised web event data • Visualised audio data • Visualised mouse trail data • Screenshots displayed Each of these interfaces must be in sync with the central timeline.

Acceptance Criterion	• The Visualiser presents the associated study emotion data on a 2D graph. • The Visualiser allows for multiple emotions to be plotted on a single graph. • The Visualiser plots selected Web Events on the 2D graph according to their time of occurrence. • The Visualiser presents the audio data using a waveform visualisation. • The Visualiser displays the full and partial web page. • The Visualiser displays a heatmap visualisation of the participant's mouse trail overlayed on the full web page. • The Visualiser displays a mouse trail visualisation of the participant's mouse trail overlayed on the full web page. • All visualisations and controls conform to the time selected in the audio visualisation.
Result	**Pass**

TEST_VS04

Asserts Specification(s)	VSSPEC10, VSSPEC13
Description	The Visualiser must allow the researcher to align all open experiments by a common event in each of the recordings. This point in time should be automatically navigated to in each recording.

Acceptance Criterion	• The Visualiser will display a list of common events. • These events must occur in each of the open experiments. • The Visualiser will advance each open experiment to the point in time where the selected event occurs.
Result	**Pass**

TEST_VS05

Asserts Specification(s)	VSSPEC16, VSSPEC18
Description	The Visualiser must be well documented.
Acceptance Criterion	• The source code is commented correctly and is supported by Doxygen documentation. • A detailed and understandable user manual is provided for non-technical users.
Result	**Pass**

TEST_VS06

Asserts Specification(s)	VSSPEC17, VSSPEC20
Description	The Visualiser must be responsive, even when multiple recordings are being played simultaneously.

Acceptance Criterion	• The Visualiser is responsive to the researcher's input. • The Visualiser utilises all of the available CPU resources available to it when performing demanding or complex operations.
Result	**Pass**

TEST_VS07

Asserts Specification(s)	VSSPEC19
Description	The Visualiser must be run on appropriate hardware.
Acceptance Criterion	• The Visualiser runs on hardware that conforms to the specified requirements, or greater.
Result	**FAIL**
Description	When demonstrating the Visualiser at a recent fayre, the application was run on hardware that was below the specified requirements. As such, this test fails, but really the application benefits since it is efficient enough to run on hardware below the originally specified requirements. This is addressed in **TEST_VS08**.

TEST_VS08

Asserts Specification(s)	VSSPEC19
Description	The Visualiser must be run on appropriate hardware. (Addresses test **TEST_VS07**)

Acceptance Criterion	• The Visualiser runs on hardware that conforms to the specified requirements, or greater. The test is considered as passing if the application can run on hardware that is below the recommended requirement.
Result	**PASS**

Bibliography

[GKKW10] M. Gousset, B. Keller, A. Krishnamoorthy, and M. Woodward. *Professional application lifecycle management with Visual Studio 2010*. Wrox, 2010.

[Ler10] J. Lerman. *Programming Entity Framework: Building Data Centric Apps with the ADO. NET Entity Framework*. O'Reilly Media, Inc., 2010.

[Lev11] J. Levinson. *Software Testing with Visual Studio 2010*. Addison-Wesley Professional, 2011.