T-76.115 Peer Test Plan
OpenLogbook

Ilkka Neuvonen ilkka.neuvonen@hut.fi
Mikko Fallenius Mikko.Fallenius@iki.fi
Ville Nenonen Ville.Nenonen@hut.fi

24th February 2003

Abstract

This document is a guide to testing the OpenLogbook software intended for the peer testing group within the HUT course T-76.115. It contains instructions on how to conduct the peer tests.

Contents

Version	Date	Author	Description
1.0	9.2.2003	Ilkka	First release version
1.1	24.2.2003	Mikko	Added the test cases
1.2	24.2.2003	Ville	Changed and added test cases

Table 1: Changelog

More detailed history log can be found from WebCVS of the project [5].

1 Introduction

1.1 Purpose and scope of this document

This test plan is a guideline for OpenLogbook's peer testing group for how the software is to be tested. All tasks and data that will be produced as a result of the testing are defined in this document.

1.2 Product and enviroment

OpenLogbook is a development project aiming at producing software with which a scientist can review an experiment and efficiently search through different types of data that has been recorded. These data might include video/audio streams, written comments and log files of different parameter combinations used - all information needed to reconstruct the experiment afterwards.

1.3 Terms and abbreviations

Definitions, terms and abbreviations are presented in their own document [3].

2 Responsibilities

Planning the peer testing is a responsibility of the OpenLogbook team, whereas executing the tests is a responsibility of the peer testing group. Should the need arise, the peer testing group may contact the OpenLogbook team via project manager Mikko Fallenius <mikko.fallenius@iki.fi>.

3 Reporting

3.1 Test report

Results of the peer testing will be retuned as a separate test report as defined by the model peer test report [7].

3.2 Fault reports

Fault reports will be entered into the Burana -system [6]. The following subsystems have been defined in Burana:

- Core
- Data
- Gui
- Timeline
- Utils
- openlogbook

Detailed descriptions of each subsystem can be found in the OpenLogbook technical specification document [4]. When the source of the fault is ambiguous the “openlogbook” -subsystem shall be used.

4 Testing tasks and procedures

4.1 Prerequisites

4.1.1 Provided by the OpenLogbook team

- An executable jar-file containing the Openlogbook Software.
- Media and data files used in testing.
- Access to OpenLogbook’s Burana-account.
- Peer test plan.
- Peer test report template.
- User guide.

4.1.2 Required from the peer testing group

- a Windows-computer capable of executing Java2 1.4.0 applications

4.2 Testing the installation instructions

The fist step in peer testing is testing the installation instructions in the Openlogbook User’s Guide [2] by installing the software. The testing should be executed as follows:

- The tester(s) install OpenLogbook as described in the Installation instructions -chapter of Openlogbook User’s Guide [2]
- The installation -section of the peer test report [7] is filled.

4.3 System testing

Having the software installed the peer testing group will start the system testing using the User’s guide as a reference. The tests will be performed as follows:

- The group will execute the test cases defined in Appendix 8 in the given order.
- Found faults are entered into the Burana-system [6].
- The faults are reported in the peer test report [7] referencing the burana fault id.

4.4 User’s Guide

As the last step of testing the User’s Guide -section of the peer test report is filled.

5 Test case and fault priorities

5.1 Test case priorities

The test cases are categorised with the priority levels shown in the following table:

Priority	Description
High	Functionality is essential for testing the major features of the program.
Medium	Functionality is important, but not essential for the system.
Low	Functionality has little effect on the system.

Table 2: Test case priority levels

5.2 Fault priorities

All found faults are given a priority depending on how critical the fault is. The following table describes different severity classes for faults:

Severity	Description
Critical	The fault is critical when it affects the whole program and testing cannot be continued.
Serious	The fault is serious when some parts of the testing cannot be finished.
Normal	The fault is normal when the fault affects functionality but doesn't stop testing.
Minor	The fault is minor when it doesn't affect functionality but is annoying or distractive.

Table 3: Fault priority levels

6 Test case pass/fail criteria

6.1 Pass/fail criteria

The following table represents the requisites for failing (and aborting) a test item. In the table X means the test case has failed.

Test case Priority	Critical fault	Serious Fault	Normal fault	Minor fault
High	X	X	X	X
Medium	X	X	X	
Low	X			

Table 4: Fail criteria

6.2 Critical faults

Should a critical fault arise during the testing, the testing is suspended. The OpenLogbook group must be informed immediately so that the problem may be resolved and the testing may continue.

6.3 Serious faults

Should a serious fault arise during the testing, the test cases related to the fault are suspended. The OpenLogbook group must be informed immediately so that the problem may be resolved and testing the related test cases may continue.

6.4 Finishing criteria

The testing is finished when all the test cases have been executed.

7 Risk management

All risks have three different parameters on a scale from 1 to 5:

- Probability factor. The probability that this risk will occur. This reflects the percentual chance of occurence, ie. if a risk is expected to occur with a 50% probability, it will be given a factor of 3.
- Severity factor. How severely this risk will affect the project.
- Impact factor. This is the total risk which is obtained by multiplying the probability factor with the severity factor.

The OpenLogbook team has identified the following risks in the peer testing:

Insufficient time for completing tests
The testing planned for the peer group cannot be completed due to lack of budgeted time.
Probability: 2
Seriousness: 2
Impact on testing: 4
Controlling actions:
The test cases have been ordered so that the most important and extensive test cases will be executed first.
Consequences:
All the data expected from the peer testing will not be acquired.

Technical difficulties prevent testing
Unexpected technical difficulties prevent completion of the planned test cases.
Probability: 2
Seriousness: 4
Impact on testing: 8
Controlling actions:
The OpenLogbook group has prepared to assist the peer test group in solving the problems.
Consequences:
Some of the budgeted time from the peer test group as well as some OpenLogbook project time is lost on the corrective measures.

Lack of testing motivation
The peer testing group is not motivated enough to perform proper testing.
Probability: 3
Seriousness: 4
Impact on testing: 12
Controlling actions: OpenLogbook-group will deliver good plans so the testing will be fun and easy to do.
Consequences:
The test data acquired will impaired.

8 Test cases

8.1 Test case PT01: Installing the software

Code:	PT01
Name:	Installing the software
Priority:	High
Description:	Installs the software using the installation instructions found in the User's Guide.
Focus:	Are the instructions sufficient? Are there any inconsistencies in the user's guide? How long did the installation procedure take?
Notes:	

Table 5: Test case PT01

8.2 Test case PT02: Recording an experiment

Code:	PT02
Name:	Recording an experiment
Priority:	High
Description:	Create and record a new experiment. Add hot spots and text comments during the experiment. Make the the test experiment last at least 15 minutes.
Focus:	Is the recording phase intuitive to use?
Notes:	This test case is a bit fruitless because there is no real experiment to record.

Table 6: Test case PT02

8.3 Test case PT03: Adding and removing material

Code:	PT03
Name:	Adding and removing material.
Priority:	High
Description:	Add and remove different video, audio, and parameter log files.
Focus:	
Notes:	After several addings and removings leave to the experiment at least 2 videos, 1 parameter log file, 1 sound file and 1 other log file.

Table 7: Test case PT03

8.4 Test case PT04: Save experiments

Code:	PT04
Name:	Save experiments
Priority:	High
Description:	Save an existing experiment created before. Make some changes to it and save it again with a different name.
Focus:	
Notes:	

Table 8: Test case PT04

8.5 Test case PT05: Load experiments

Code:	PT05
Name:	Load experiments
Priority:	High
Description:	Close the program and load both experiments saved before.
Focus:	Is everything that was saved also loaded?
Notes:	

Table 9: Test case PT05

8.6 Test case PT06: Review an experiment

Code:	PT06
Name:	Review an experiment
Priority:	High
Description:	Review an experiment by playing it. Move in time using the hot spots. After reviewing one experiment try to review two experiments at the same time.
Focus:	All data streams are played, and synchronisation works.
Notes:	

Table 10: Test case PT06

8.7 Test case PT07: Test the synchronisation

Code:	PT07
Name:	Test the synchronisation
Priority:	High
Description:	Set offset for every media file in both experiments. Play all data files in synchronisation. Then view just one file so that it gets out of synchronisation. Then restore the synchronisation and play the data files in synchronisation. Try to do the previous things to both experiments and play them at the same time.
Focus:	
Notes:	

Table 11: Test case PT07

8.8 Test case PT08: Arrange the GUI

Code:	PT08
Name:	Arrange the GUI
Priority:	Medium
Description:	Test the GUI by first opening an existing experiment with some data files, and then moving, resizing, hiding, bringing forth and closing the windows. Try also playing around with GUI when experiment is played. Try also on two experiments that are open.
Focus:	
Notes:	

Table 12: Test case PT08

8.9 Test case PT09: Modify hot spots

Code:	PT09
Name:	Modify hot spots
Priority:	Medium
Description:	Modify hot spots' descriptions and their time values.
Focus:	What happens with invalid values?
Notes:	

Table 13: Test case PT09

8.10 Test case PT10: Try to make the program fail/crash

Code:	PT10
Name:	Try to make the program fail/crash
Priority:	Low
Description:	Try everything that you can make up and try to make program fail or crash. Try also do something unexpected.
Focus:	
Notes:	If you find any faults, please also document how did you find them.

Table 14: Test case PT10

9 In the end

If you have any new ideas or any other comments on the program we are more than happy to see them.

References

[1] OpenLogbook project webpages for the course T-76.115
`http://motha.tky.hut.fi/openlogbook/`
Cited: 03.02.2003.

[2] OpenLogbook User's Guide `http://motha.tky.hut.fi/openlogbook/`
Cited: 09.02.2003.

[3] OpenLogbook Terminology document
`http://motha.tky.hut.fi/openlogbook/delivery/terminology_doc.pdf`
Cited: 03.02.2003.

[4] Openlogbook: Technical Specification
`http://motha.tky.hut.fi/openlogbook/delivery/technical_spec.pdf`
Cited 03.02.2003.

[5] OpenLogbook CVS archive on the web
http://motha.tky.hut.fi/cgi-bin/cvsweb/openlogbook/
Cited: 03.02.2003.

[6] Burana error reporting system
http://www.soberit.hut.fi/T-76.115/02-03/raportointi/burana/
Cited: 03.02.2003.

[7] A model Peer test report
http://motha.tky.hut.fi/openlogbook/delivery/peer_test_report.pdf
Cited:09.02.2003.

[8] OpenLogbook: Test plan
http://motha.tky.hut.fi/openlogbook/delivery/test_plan.pdf
Cited: 09.02.2003.