Testing document x.x

SQUID

Helsinki 19th April 2005

Software Engineering Project

UNIVERSITY OF HELSINKI
Department of Computer Science

Course

581260 Software Engineering Project (6 cr)

Project Group

Mikko Jormalainen
Samuli Kaipiainen
Aki Korpua
Esko Luontola
Aki Sysmäläinen

Client

Lauri J. Pesonen
Fabio Donadini
Tomas Kohout

Project Masters

Juha Taina
Jenni Valorinta

Homepage

`http://www.cs.helsinki.fi/group/squid/`

Change Log

Version	Date	Modifications
0.1	31.3.2005	First version (Aki Korpua)
0.2	5.4.2005	Corrections, Use Case sample (Aki Korpua, Mikko Jormalainen)
0.3	12.4.2005	Use case split and now look (Aki Korpua)
0.4	13.4.2005	Integrity test cases (Aki Korpua)
0.5	13.4.2005	Some fixes (Aki Korpua, Mikko Jormalainen)
0.6	15.4.2005	Macros and fixes to layout (Aki Korpua)
0.7	18.4.2005	Some fixes and whacking latex (Samuli Kaipiainen)
		Project Explorer, Calibration and Measurement Controls test cases (Samuli Kaipiainen)

Contents

1 Introduction

This document describes how this software (Ikayaki) is planned to be tested properly. Mainly this document concentrates on describing methods used for testing and test cases. It is important that all members of team to make tests in same way. This lowers possibility of testing conflicts and helps on integration test phase.

2 Overview of testing the system

Because program will be used to control a magnetometer, testing will be more important than in normal software engineering student projects. We will do unit testing for each class, integrate testing to program and use separate squid emulator to test squid interface system.

In unit testing each class is tested independently. Unit testing will be done by using JUnit. Every programmer will test his own classes. Class should be tested when it is finished and corrected before integration test begins.

Integration testing tests interfaces between classes. It will be done by going through all user interface protos and checking that all sections in requirements document can be done. Some critical sequences which are done many times with program should be done too.

Squid interface integration testing is done simulating real system with emulator. It will be done using Squid-emulator before testing it with real magnetometer. Squid-emulator runs in different machine and is connected by few (2-3) Serial I/O cables. Squid-emulator will be tested with old program (2G) same way before testing Ikayaki-system so that it will have all same tested properties which old program have and both systems have same results with squid emulator.

To verify that old program and new program works same way, we will do critical measurement with old program and emulator, save emulators log file and then use emulator with that log file and do same critical measurement with new program and see that both have same results.

If Rita testing utility is easy enough to use it will be used in testing. Tests will be constructed in such way that every line of code is visited at least once.

3 Testing

3.1 Unit Testing

Unit Testing is done for each class separately. Classes tested with JUnit have it's own ClassNameTest.java class in test-directory and gui-components are tested manually. They should be done before and during coding class. All test cases must be executable after classes are ready. Every class should be tested succesfully before integration tests.

All test cases are listed in sections 4 and 5.

3.2 Integration Testing

Integration testing is mostly done after unit tests are passed for all classes. Testing is done using squid emulator at first and finally with real Squid system. Graphical User Interface is mainly tested in Unit testing and is only looked that all components work together as Squid Interface is tested.

Integration testing for GUI-components is done using top-down method. Testing is done partly during implementation phase, when new components are added to program. After adding component to program it is tested to work properly with other components.

Graphical User Interface is tested using all use cases from 6. If one fails, it is corrected immediatelly and all use cases must be done again. When all use cases are done without errors this phase is ready.

Squid Interface testing is done with Emulator. Use cases are Automatic Measurement, Thellier Measurement and Manual Measurement with all variations (see 6). In first phase we use old program with emulator, save its log file and take results. After that we run it with new program and compare results. If they are not same, corrections are made immediatelly to new program and test is run again. This is done until results are same for all use cases.

3.3 Squid Emulator

Squid emulator is tested with old program. Use cases are Automatic Measurement, Thellier Measurement and Manual Measurement with all variations (see Requirements Document). When all use cases can be done with emulator it is ready enough for integration testing.

This should be done so that Old Program has same results with SQUID-system and emulator for all use cases. But we don't have resources for this. So this test only tells us that Old and New program works same way with squid emulator.

4 JUnit test cases

Here are listed JUnit test cases for classes.

JUnit is simple java-based framework for testing your java classes. We use it in this project for Unit Testing. For more information visit http://junit.sourceforge.net/.

First you need to download JUnit from http://sourceforge.net/project/showfiles.php?group_id=15278 and extract it to directory (different than java directory) and set classpath for it.

Then you must write test class for every class to be tested. Test classes extend TestCase. They will have test methods, one for each Test Case. Test class also need suite()-method and Main for run. Sample test class:

```
  import java.util.*;
  import junit.framework.*;

  public class SimpleTest extends TestCase {

  //Simple Test Case

  public void testEmptySimple() {
        Simple simple = new Simple();
        //New Simple must be empty
        assertTrue(simple.isEmpty());

  }

  public static Test suite() {
        return new TestSuite(SimpleTest.class);
  }

  public static void main(String args[]) {
        junit.textui.TestRunner.run(suite());
  }
}
```

Type "java junit.swingui.TestRunner SimpleTest" to run test cases for Simple.

4.1 SampleClass

1. Test1
    - Cond1
    - Cond2
2. Test2
    - Cond1

Test results for SampleClass: 0/0

5 GUI-component test cases

Here are listed test cases for gui-classes, these are done manually just clicking and changing values.

5.1 ProjectExplorerPanel

1. Directory text field
    - Shows autocomplete popup when typing and typed text matches to one or more directory
      OK
    - Shows directory history on down-arrow click
      OK
    - Too long lines in popup are shortened
      OK
    - Tries to change directory when on enter press or mouse-click to popupmenu (UC21)
      OK
    - Only allows correct and existing directories (flashes red on change-attempt if invalid)
      OK
2. Browse-button
    - Opens a dir chooser dialog
      OK
    - Changes to chosen directory if "Open" selected; updates text field (UC21)
      OK
3. Project file table
    - Shows project files in current directory
      OK
    - Only shows project files (.ika and Calibration/AF/Thellier/Thermal)
      OK
    - Hilights open project
      OK
    - Hilights (with different color) currently measuring project
      OK
    - Loads project on project file mouseclick (UC20)
      OK

- Show export menu for clicked file on right-click
  OK
- Export to selected file if any chosen (UC13-UC15)
  OK
- Sorts data on header click
  OK
- Updates data if changed (modifield-column, that is)
  OK

4. "Create New" -components
   - Only allows correct, non-existing files; flashes red if invalid
     Test: / or \ **FAILED!**
   - If valid filename, creates it, updates table with created file selected (UC19)
     OK

Test results for ProjectExplorerPanel: 17/18

5.2 CalibrationPanel

1. Calibration project table
   - Shows (and only shows) calibration project files in current directory
     OK
   - Hilights open project
     OK
   - Hilights (with different color) currently measuring project
     OK
   - Loads project on project file mouseclick (UC20)
     OK
   - Doesn't show export menu for clicked file on right-click
     FAILED! :)
   - Sorts data on header click
     OK
   - Updates data if changed (measured- and elapsed-columns, that is)
     OK
2. "Calibrate" -button
   - Does whatever "Single Step" -button does (see MeasurementControlsPanel) (UC8-UC10)
     OK

Test results for CalibrationPanel: 7/8

5.3 MeasurementControlsPanel

1. Measuring buttons
   - All of them are enabled/disabled by good manners
     OK
   - On click, all of them execute their measuring command without unjustified complaints (UC1-UC7)
     OK
   - Flash red on click if action can’t be done
     OK
2. +Z/-Z info/radiobuttons
   - Shows correct orientation for current project
     OK
   - Changes orientation on +z or -z radiobutton click
     OK
3. Magnetometer status picture
   - Always shows handler positioin and rotation as supposed...
     OK
   - Updates smoothly ^^
     FAILED!
4. Handler manual controls (radiobuttons)
   - Disabled if "normal" measuring in action, enabled otherwise
     OK
   - Executes desired action (if possible; can’t tell) (UC11)
     OK
   - Stack move-radiobuttons correctly (never on top of each other or missing)
     Test: Any two handler positions are set (in Settings) to same location (number)... **FAILED!**
5. Magnetometer/demagnetizer manual controls (buttons)
   - Disabled if "normal" measuring in action, enabled otherwise if applicable
     OK
   - Enable/disable and update text according to current handler position and rotation
     FAILED!
   - Executes desired action (if possible; flash red if not) (UC11)
     OK

Test results for MeasurementControlsPanel: 10/13

5.4 SettingsPanel

1. Magnetometer, Handler and Degausser COM-port combobox
    - Shows all COM-ports on system in alphabetical order
      OK
    - Loads correct COM port from Settings
      OK
    - Doesn’t allow same value as in Handler and Magnetometer/Degausser COM-port comboboxes
      FAILED!
2. Save button
    - Only available if changes are made and values are permissible
      OK
    - Unavailable on start
      OK
    - On click closes window and saved data correctly
      OK
3. Cancel button
    - On click closes window and doesnt save changes
      OK
    - Always available
      OK
4. Calibration constants
    - Loads data correctly from Settings
      OK
    - Accepts positive and negative decimal numbers
      Test: -0.430002 OK
5. Degausser ramp
    - Loads data correctly from Settings
      OK
    - Has only elements: 3,5,7,9
      OK
6. Degausser delay
    - Loads data correctly from Settings
      OK

- Has only elements: 1..9
  OK

7. Acceleration and deceleration fields

   - Loads data correctly from Settings
     OK
   - Accepts only values in range of Integers 0..127
     Test: 1000 OK Test: -10 OK

8. Velocity and velocity in measurement fields

   - Loads data correctly from Settings
     OK
   - Accepts only values in range of Integers 50..20000
     Test: 0 OK Test: 2001 OK

9. Translation positions fields

   - Loads data correctly from Settings
     OK
   - Accepts only values in range of Integers 0..16777215
     Test: -1 OK

10. Right limit

    - Loads data correctly from Settings
      OK
    - Has only elements: plus, minus
      OK

11. Rotation field

    - Loads data correctly from Settings
      OK
    - Accepts positive and negative Integer numbers
      Test: -1 OK Test: 0.1 OK

Test results for SettingsPanel: 26/27

5.5 SampleClass

1. Use case1

   - Cond1
     Test: Description1 OK

- Cond2
  FAILED!

Test results for SampleClass: 1/2

6 Integration test cases

Here is test case which is done in integration phase to all components at same time. This is done in following order and must be repeated until done succesfully. Variations can be made to ensure that all works in different situations.

6.1 Automatic AF Measurement

1. Open software
   - Last project is opened, if available.
   - All components are in right place
2. Select Sample holder calibration
   - opens Sample holder calibration project in Measurement Sequence and Information
   - disables Measurement Controls
   - enables calibrate button
   - disables editing sequence
3. Click Calibration button
   - Adds new line in Sequence at bottom
   - Starts calibration, progress shown on measurement controls
   - Data is updated correctly in Measurement Details
   - Calibration button changed to stop button

4. Calibration finished
    - New data added to Measurement Sequence
    - Stop button is changed to calibration button
    - Last modified is updated Sample Holder
5. Select standard sample
    - opens Standard Sample calibration project in Measurement Sequence
6. Click Calibration button
    - Adds new line in Sequence at bottom
    - Starts calibration, progress shown on measurement controls
    - Data is updated correctly in Measurement Details
    - Calibration button changed to stop button
7. Calibration finished
    - New data added to Measurement Sequence
    - stop button is changed to calibration button
    - Last modified is updated in Standard Sample
8. Create new AF project
    - Calibration project is closed
    - new AF project is opened with given name (ProjectInfo,MeasurementSequence)
    - Measurement Controls has enabled buttons Measure,Single Step and disable Stop Now!
9. Add project Information

- All data is accepted

10. Load Set
    - Loaded set is correctly added to Sequence

11. Click Measure
    - Measure button changed to Pause, disables Single Step, enables Stop Now!
    - Rows are highlighted in Explorer and Sequence
    - Starts measuring first phase correctly (no demag now)
    - Picture of demagnetizer is up-to-date
    - Plots are drawn when one phase is over
    - Continues on next phase and does demag first
    - updates Details and Sequence
    - any sequence cannot be changed

12. Pause button
    - Finish current measure and stops after that
    - Pause button changed to Measure button
    - When finished disables Stop Now!

13. Sequence edit
    - Steps not yet measured, can be changed

14. Single step button
    - Performs Next phase on measure

- Disables Measure button and Single Step button, enables Stop Now!

15. Measure button
    - Measure button changed to Pause, disables Single Step, enables Stop Now!
    - Rows are highlighted in Explorer and Sequence
    - Continues from Next phase
    - Picture of demagnetizer is up-to-date
    - Plots are drawn when one phase is over
    - Continues on next phase
    - updates Details and Sequence
16. Stop Now! button
    - Stops measurement immediatelly
    - Disables buttons and enables Manual Control
    - What now? Is project Manual or is this only temporary?

Test results for Automatic AF Measurement: 0/0

6.2 Thellier Measument

1. Open software
    - Last project is opened, if available.
    - All components are in right place
2. Open existing Thellier project
    - Project is opened correctly in Information and Sequence

- Measure button is hidden, Single step enabled and Stop now Disabled

3. Add new line in sequence
    - It’s inserted correctly.

4. Click Single Step button
    - Performs Next phase on Sequence
    - Disables Single Step button, enables Stop Now!
    - Rows are highlighted in Explorer and Sequence
    - Picture of demagnetizer is up-to-date
    - Plots are drawn
    - updates Details and Sequence

5. Select next Project
    - Opens project correctly
    - Adds same phase to sequence (as in last project added)

6. Click Single Step button
    - Performs Next phase on Sequence
    - Disables Step button, enables Stop Now!
    - Rows are highlighted in Explorer and Sequence
    - Picture of demagnetizer is up-to-date
    - Plots are drawn
    - updates Details and Sequence

Test results for Thellier Measument: 0/0

6.3 Manual Measurement

1. Open software
    - Last project is opened, if available.
    - All components are in right place
2. Create new AF/Thellier project
    - new AF/thellier project is opened with given name (ProjectInfo,MeasurementSequence)
    - Measurement Controls has enabled buttons (Measure,) Single Step and disable Stop Now!
3. Adding project data and selecting manual
    - Data is accepted correctly
    - Manual is enabled
4. Rotate handler
    - Rotation changes on image of squid
    - Squid-system rotates
5. Move handler to specific place
    - Image of squid animates correctly and stops on right position
    - Squid-system moves to exact place
6. Demagnetize
    - Demagnetization is shown on squid image
    - Squid-system demagnetizes
7. Measure-all

- Measuring is shown on squid image
- Data is added to Details correctly (right rotation, background)
- Data is added to Sequence (only if all degrees done?)

8. Reset measure
    - Nothing shown, Magnetometer is reseted.

Test results for Manual Measurement: 0/0