Realization Document 1.0

SQUID

Helsinki 6th May 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	5.5.2005	Trunk (Samuli Kaipiainen)
0.2	5.5.2005	Trunk v2 (Samuli Kaipiainen)
0.25	5.5.2005	My changes (Samuli Kaipiainen)
0.3	5.5.2005	TODO-list-raw-trunk (Samuli Kaipiainen)
0.33	5.5.2005	TODO-items, graphs and some text (Aki Korpua)
0.35	5.5.2005	My TODO-items (Samuli Kaipiainen)
0.4	6.5.2005	TODO-items (Esko Luontola)
0.5	6.5.2005	Cleaning up (Samuli Kaipiainen)
1.0	6.5.2005	No time for any more tweaking (Samuli Kaipiainen) Some little additions (Esko Luontola)

Contents

1 Introduction

This document describes the changes from design to production (section 2), improvement suggestions for the program (section 3), as well as a TODO-list based on comments in source code (section 4).

2 Changes from Design document

Here we describe how the implementation/production of the software differs from that planned in Design document.

2.1 Data classes

Data classes are in packages ikayaki, ikayaki.squid and ikayaki.util.

Package ikayaki holds generic data classes, ikayaki.squid holds classes loosely related to the squid interface, and ikayaki.util has utilities.

2.1.1 Project data

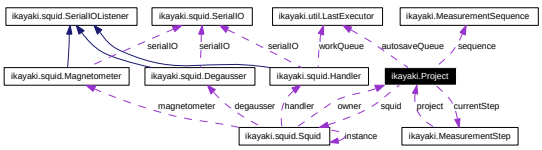

Figure 1: project-graph

Responsible for holding all the measurement data and controlling the SQUID. All of these classes were produced according to plans and there were no significant changes. Things such as susceptibility and some other data from the measurements was added. The way that the mathematical calculations are made, had some changes, such as the holder and background noise corrections.

2.1.2 Squid interface

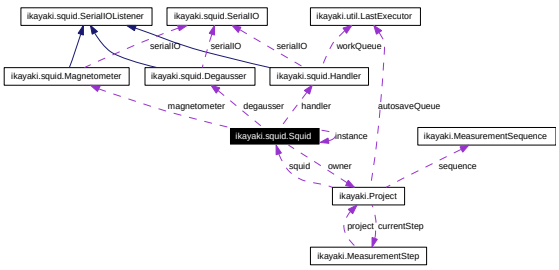

Figure 2: squid-graph

Squid Interface offers the Project class an interface to safely control the SQUID magnetometer. The Squid class holds three classes that handle communication to to three separate parts of the SQUID (Handler, Degausser and Magnetometer).

Classes are Squid, Handler, Magnetometer, Degausser.

These were not desinged well because it was hard to know how exactly we should use Squid at the time. Lots of changes were made and many methods were removed and added. Private status String is not used in any class and messageBuffer is not used like vise. Messages are catch on private SynchronousQueue<String> answerQueue = new SynchronousQueue<String>() variable and there is private boolean waitingForMessage variable indicating are we waiting any message from equipment.

Handler: Biggest change here is that we use WorkingQueue here. And EstimatedMovement are calculated here too. Many of the methods are change and most are removed or not used like planned.

Methods not used or removed:

poll(take)Message()

setPositionRegister(int r)

setPosition(int p)

setSteps(int s)

stop()

setBaseSpeed(int b)

setHoldTime(int h)

setCrystalFrequence(int cf)

getStatus()

Methods added or modified:

public int getEstimatedPosition() : return where is handler.

public int getEstimatedRotation() : return rotation of handler.

public boolean isMoving() : tells if we are moving.

public boolean isRotating() : tells if we are rotating.

public void moveToLeftLimit() : starts moving to left limit.

public void moveToRightLimit() : starts moving to right limit.

protected void moveSteps(int steps, int velocity) throws SerialIOException : moves steps as commanded.

protected void moveToPosition(int position) throws SerialIOException : moves handler to position, relative to home.

protected void seekHome() throws SerialIOException : seeks home position.

protected void selectMovement() throws SerialIOException : sets Handler on moving

phase.

protected void selectRotation() throws SerialIOException : sets Handler on rotating phase.

protected void setPosition(int position) : sets position for EstimatedMovement.

protected void setRotation(int rotationSteps) : sets rotation for EstimatedMovement.

protected void setUp() : sets Handler online.

protected void slewToLimit(boolean toRight) throws SerialIOException : sets Handler to go limit.

Degausser: Pretty much the same as planned. There is added blockingWrite(String command) that waits that command is sended, which is known when degausser answers it back.

Magnetometer: Pretty much the same but getter commands are not used.

2.1.3 Squid emulator

Squid Emulator is separate from the rest of the program and it is used only for testing that the Squid Interface works correctly. Biggest change was that this wasnt developed much and its not working as planned. Mainly was used to test that commands are sended correctly and Squid Interface gets answers.

2.1.4 Serial communication

SerialIO and classes related to it takes care of the harware layer of serial communication. Using these classes the program communicates with the Degausser, Samplehandler and Magnetometer. SerialIO represents one serial port and when it's created it reserves the port to itself. SerialProperties class includes all the configuration data for the serial port.

2.1.5 Global settings

Global properties that are used all around the program. There were many properties added to the Settings class. Not all of the methods are documented, so that should be done. All methods were converted to be static.

2.1.6 Utilities

Utility classes that are used in the program, but do not fit any of the other packages. There were a couple of classes added to the ikayaki.util package. The new classes are ComponentPrinter for printing, DocumentUtilities for reading and writing XML, LoggerPrintStream for adding timestamps to System.err and copying the output to a file, SerialProxy for monitoring the communication between two serial ports (use virtual serial ports to redirect the old program's communication through SerialProxy this way: Program -> Virtual COM -> SerialProxy -> Real COM -> SQUID Hardware).

2.2 GUI classes

This section is divided into subsections by gui components, each of which has one or more classes.

All gui classes are in package ikayaki.gui.

2.2.1 Generic GUI components

There were no changes to ProjectComponent.

2.2.2 Main window

Figure 3: main-view-panel-graph

Basically the same as was designed, but there were many methods added.

2.2.3 Configuration window

Separate window which is opened from menubar and it updates settings for Squid interface. Used usually only when system is installed to setup it. If there is something changed, program should be restarted as there is no quarantee that they affect directly.

2.2.4 Project Explorer

Nothing much here; some internal private field renames. Oh and a lot more stuff than planning suggests :)

ProjectExplorerTable is now its own class (as Calibration uses it too), and ProjectExplorerPopupMenu is its inner class. Also has inner class ProjectExplorerTableModel, which,

unlike normally (as in the standard way), is quite empty and most of the stuff is in ProjectExplorerTable. Uses StyledWrapper for table row background colors (and Calibration boldface-reminder).

NewProjectPanel is inner class of ProjectExplorerPanel.

2.2.5 Calibration

Uses ProjectExplorerTable, which makes CalibrationPanel.java really short.

2.2.6 Project information

Contains and allows the editing of the basic information of a sample. All plans were dumped and the classes were programmed XP-style.

2.2.7 Sequence and measurement data

Contains and allows the editing of the measurement sequence. All plans were dumped and the classes were programmed XP-style.

2.2.8 Measurement details

Contains the details of the selected row in the measurement sequence. All plans were dumped and the classes were programmed XP-style.

2.2.9 Measurement controls

Sadly, ManualControlsPanel had to go into MagnetometerStatusPanel as an inner class, since they share some data and move-radiobuttons.

MagnetometerStatusPanel's updateStatus takes no parameters, as it asks position and rotation directly from Handler. Also, status image is updated at all times once in 50 ms (positions asked with getEstimatedPosition and getEstimatedRotation in Handler).

Here too, a lot more stuff than what planning imply.

ManualControlsPanel's manual control components are disabled whenever there's *any* Squid action, to keep those fragile equipments from messing up.

2.2.10 Graphs

Graph panels visualize the measurement data. MeasurementGraphsPanel listens to MeasurementEvents to update the measurement data in plots. AbstractPlot is an abstract class which implements all the general features of graph plots. IntensityPlot and StereoPlot

extend the functionality of AbstractPlot and implement their special drawing features accordingly.

3 Improvement suggestions

What's left for the program to make it perfect :)

3.1 Adding more calculations

The algorithms for the measurement's mathematical calculations are spread around the program code in a perverted way. Look for these places when you want to change or add something:

- Magnetometer.readData() reads the raw measurement data from the SQUID.
- Project.Measurement.run() contains the routine for a measurement step. Project.ManualMeasure.run() is the same for manual measurements.
- MeasurementStep.addResult(MeasurementResult) adds the measurement data to a step, and gets the holder calibration from the Settings class when necessary.
- Project.updateTransforms() contains the transformation matrices for rotating the sample from sample coordinates to geographic coordinates.
- MeasurementStep.updateTransforms() applies the fixes and transformations to all of the MeasurementResult objects.
- MeasurementResult.applyFixes(MeasurementStep) rotates the sample to 0 angle and applies noise, +/-Z and sample holder fixes. This is the sampleVector.
- MeasurementResult.setTransform(Matrix3d) applies the transformation matrix for rotating the sample coordinates to geographic coordinates. This is the geographicVector.
- MeasurementValue classes contain the algorithms for calculating all kinds of mathematical numbers.
- SequenceColumn classes contain the information that how those numbers should be shown in the MeasurementSequenceTable. New columns are added to the table by listing them in SequenceColumn.getAllColumns(). Default columns are defined in Settings.getDefaultColumns().

3.2 Porting to Linux

There are a couple of things that need to be taken care of, if the program is ever ported to Linux or some other OS:

- Maximizing the program window (in Ikayaki's constructor) requires some native code, because Java does not have a way for doing it.
- Opening the user manual from the Help menu needs some OS-specific instructions to open the web browser.
- Serial communication will at least require the OS-specific version of javax.comm, or

maybe it needs to be done in some another way.
- Export-popup-menu (ProjectExplorerPopupMenu in ProjectExplorerPanel) uses new File("A:/") for disk drive.

3.3 Warning Signal for Degausser

As mentioned in the requirements document, the client wants that the program will make an alarm signal, if the degausser for some reason does not ramp down in time. The protocol specifications of the degausser says that it will automatically prevent that from happening, so this feature was dumped because there was no easy way to do it and it seemed unnecessary because of the hardware specifications.

Apparently Matti Leino has made such a system previously, so try to find out more from him.

Quote from Lauri Pesonen:

Siitä ambulanssivaroitussignaalista jos ramp down ei tapahdu.

Matti leino odottaa vietiä "done" noin 1 minnutin. Jos käskyä ei tule niin kelojen virtaan ERIKSEEN asennettu mikrokytkin kytkee virran pois jotta kelat eivät pala. Tämän voitte kertoa raportsissa mutta varsinainen homma tehdään syksyllä uuden ryhmän kanssa. Eli nyt tuskin ehditte mutta näin se on tehty. Soittakaa Matti Leino 0 205 50 2272 (GTK) ja hän kuvaa kuinka hän teki. Olisi tärkeätä että kirjaatte kuitenkin tuon Matti Leinon tekniikan ylös!

3.4 Documentation and Refactoring

The documents of this program could be better. Especially the user manual has been made in too much of a hurry. Rewrite them when necessary.

The structure of some parts of the source code is more or less messy. Refactor them to be more elegant, when necessary.

3.5 Accuracy of the Measurements

There appears to the some inaccuracy in either the measurements or then the calculations are not right. For some reason the measured Z values are accurate, but X and Y are not. It might be necessary to see more closely that how the old program does the measurements. The use of SerialProxy is recommended. The timing of the protocol commands might also be critical.

3.6 Graphs

Current graphs need some fixing. The stereo plot is now drawn with linear scale as it should be biased so that points are more packed together in the middle. The ratio is unknown. Zijderweld plots should be implemented as well.

4 TODO-list from source code

4.1 Ikayaki.java

The MainStatusBar is not implemented, but when it is ready, it should be added to the JFrame in this class. For now that part of the code is commented out.

4.2 MeasurementStep.java

For Thellier projects it would be necessary to store the notes for each measurement step in a better way. Now the decimals of a stepValue are used to convey information about the current measurement. Get rid of that ugly hack and add a proper property to store the information. Have a look at SequenceColumn.STEP, where this solution is explained in more detail.

4.3 MeasurementValue.java

In the calculation of THETA63 it was a bit unclear, if the length of the full vectors should be summed (like it is now), or if the vectors should be normalized first (length=1). Fabio said that the way it is right now should be OK, but I understood it differently when reading the documents that Lauri gave.

The algorithms SIGNAL_TO_NOISE, SIGNAL_TO_DRIFT and SIGNAL_TO_HOLDER are missing descriptions. Right now they are used only in the MeasurementDetailsPanel, but it would be nicer to be able to show a tooltip with explanations for them. They could also be added as columns to the measurement sequence table.

4.4 Project.java

The DEBUG field is for bypassing the SQUID hardware and running dummy measurements which generate random data. Useful for testing the user interface without the need to go to the laboratory. The program has crashed lately a couple of times when running the dummy measurement, but there was no need and time to fix it, because the real measurements are working fine.

It would be nice to create a DTD for the XML documents written by the Project class. Especially when there comes the need to make improvements to the file format, having a

DTD for each file format version will be useful in testing.

When changes to the project file format are made, the constructor Project(File,Document) should be made so, that when it encounters an old version, it will modify the Document object that was given as a parameter. When it has been modified to the new file format, the algorithms for the latest version will be used to create an instance of the Project class. This avoids the need to change the importing of older versions when new versions are made.

Exporting to a .SRM file is not implemented. There were no specifications as to what the file format is. It is not even sure, whether such a file format even exists or not.

In exportToTDT-method it is a bit unclear that how "the applied field value in mT" should be calculated. I have got the understanding that this program will not demagnetize the sample, so how will it then find out the "applied field value"?

In methods addSequence, addStep, removeStep and removeStep it might be better, that instead of returning a boolean value, they would throw an exception when somebody tries to modify the sequence when it is not allowed. This would help in finding bugs from the program, because nobody should call these methods in the first place when it is not allowed.

About the method doManualReset: it is not clear as to what it means to "reset the magnetometer". Find out what it means and make the method's documentation more detailed.

In doManualReset-method and Measurement-class there are some notes, that two methods in the Magnetometer class would be better when combined to just one method.

4.5 Settings.java

Many of the methods are undocumented, especially in the case of device settings. Write better documentation for those.

About the handler's positions: it should be possible to allow also negative values, because the Handler class should be smart enough (it calculates the relative positions itself and tells the handler only that how many steps to move).

If any changes are made to the Settings class, special care should be made to go through every place where those properties are used. Especially the ranges of acceptable values need to be checked, because the user interface already does some checking on them.

4.6 gui/DeviceSettingsPanel.java

There are many settings in Squid that apparently are not needed. So there is now only those which seems only to be needed.

There is no check if all systems have same port, only degausser and magnetometer can have same port.

Window cannot be closed with ESC-key as we wanted.

4.7 gui/IntensityPlot.java

X and Y axes have no ticks nor numbers.

4.8 gui/MagnetometerStatusPanel.java

If any two handler positions (read from Settings) are the same, the corresponding move-radiobuttons won't be moved correctly, as they are dumped into a TreeMap. There are no same positions in current lab setup, and don't know if there ever could be, so this won't matter. However, Device Configuration doesn't check what is inputted there, so let's just hope noone messes with those.

Call for updateButtonPositions(...) is in paintComponent, as otherwise the button positions won't stick; don't know what's the right place then for that method call.

Changing the "Measure BG"/"Measure XYZ" button text changes whole MagnetometerStatusPanel width; should prevent that somehow.

4.9 gui/MainMenuBar.java

The code for exporting SRM files is commented out, because there is not yet support for it in the Project class.

4.10 gui/MainStatusBar.java

This class has not been implemented. It could be used for displaying error messages and telling how many minutes the measurements will take (has also a progress bar). Tooltips and other help text could also be shown there.

One way might be to add a method to the Settings class, so that it would be easy to fire error messages from anywhere in the program, and they would be forwarded to the status bar. Use GUIDe to design the best way to use the status bar, if it is needed.

4.11 gui/MainViewPanel.java

There are no error messages created when communication with the SQUID can not be reached. Some message in status bar might be good. The old program gives a huge pile of popups, keep away from that anti-pattern. ;)

The button that is being used for hiding the project explorer tab, could be refactored to be an Action, so that it could be added to the MainMenuBar as a menu item.

4.12 gui/MeasurementDetailsPanel.java

Make tooltips for the table headers. Look at how they were done in MeasurementSequencePanel.

4.13 gui/MeasurementGraphsPanel.java

Make the button for opening a large version of the graphs look nicer. Maybe a small icon of a magnifying glass with a plus sign in the corner of the graphs panel.

4.14 gui/MeasurementSequencePanel.java

The actions in SequencePopupMenu should be refactored. They should be made so, that they do not depend on the parameters given to SequencePopupMenu, but instead they will themselves find out which of the rows are selected in the sequence table. They will need to listen to the ListSelectionModel of the sequence table, and enable/disable themselves whenever the operation is or is not permitted. The goal is to add these actions to the MainMenuBar and assign hotkeys for them, so that it would be possible to insert and remove rows from the sequence without the mouse.

4.15 gui/PrintPanel.java, util/ComponentPrinter.java

The way that it is decided, where to split the page when the print will not fit on one page, could be made more generic. The algorithm could be something like this:
- Go through all components in the panel with the help of Container.findComponentAt(int x, int y). Start from the upper left corner (x=0, y=0) and find out the lower right corner of the component at that position and save it somewhere. Make a note that everything that is from that point towards up and left has been "printed".
- Now look for the next component in order, which is closest to the top and has not been "printed", and do the same tricks for finding out its bottom right corner.
- This way you can iterate through the whole container, and find out where the bottom edges of the components are. When you find out that one component will not fit to the current page to be printed, write down somewhere that the page break will be set right above that component, so that the component will fit on the next page. - When you have gone through the panel to be printed, and have written down where the page breaks should be, rest of the printing will be easy. Just cut the pages at the right points.

4.16 gui/ProjectExplorerPanel.java

Cycling through popup menu list with up/down keys changes directory; it shouldn’t, but can’t recognize those changed-events from mouse clicks.

When mouseclicking autocomplete list item, textfield gets cleared because of setBrowserFieldPopup(...); no easy way around this, as other ways tried cause more other problems.

Many problems here arise from the fact that JComboBox isn't designed for recklessly changing popup menu contents (as we swap between directory history and autocomplete results). It might have been better to make an own component here, instead of using JComboBox, as now the whole thing has a lot of bubblegun stitching. But, it's good enough now.

4.17 gui/ProjectExplorerTable.java

Table columns have no indication for sort column, as Esko didn't like the *-indicator :)

ProjectExplorerPopupMenu uses new File("A:/") for disk drive; should be changed for any linux/etc porting.

SRM export is commented out, as it's not supported (in Project).

There are no messages telling if exporting was succesful or not (as I didn't want any popups for it)... Also, exporting overwrites any previous files with the same name; this might be just what the user wants, but it could also cause an unhappy surprise.

Also, *for some reason*, table selection for CalibrationPanel didn't work in the Squid lab last time we tried; it works everywhere else, so can't really say what's the problem. Could be related to US locale in lab computer. The calibration project is opened fine, but then someone clears table selection. Finally, don't know if it was a temporary bug, and now everything works fine, but even so there's something weird that could break it in the future.

4.18 gui/SequenceColumn.java

Remove the hack for Thellier files. See the TODO-list of MeasurementStep.java in this document.

Ask the client how they like to number format "0.000E0" and change it if necessary. It might be necessary to write a custom NumberFormat class, if Java's standard libraries are not enough (let's say that you want to have a "+" sign for positive exponents, and have the "E" as lowercase "e").

4.19 gui/StyledWrapper.java

It is possible to add more properties to be modified. Insets might be useful for some.

4.20 squid/Degausser.java

This class seems to work alright, but there is some issues which are uncertain. It have not been tested so that Degausser and Magnetometer are in same port, more likely there

will be many problems. This class trusts that everything goes alright, would be good to confirm status sometimes and check that everything is going well. ExecuteRampUp() and executeRampDown() are not used because there is risk that ramp stays up, executeRamp-Cycle is only used.

4.21 squid/Handler.java

Estimated movement does not calculate acceleration so its not exactly correct, but works well enough.

4.22 squid/Magnetometer.java

This class seems to work alright, but there is some issues which are uncertain. It have not been tested so that Degausser and Magnetometer are in same port, more likely there will be many problems. We only use filter 1x and range 1x and disable fast-slew, no idea if other options are needed for these. And we do not check status from magnetometer, we just hope all goes well. In measuring we do not Never use flux counting, never be needed tough.

4.23 squid/Squid.java

There is no functionality for UpdateSettings(), so when settings are changed program needs to be restarted. This should be corrected as it was planned. And there is no call from DeviceSettings for this.

4.24 squid/SquidEmulator.java

No further development needed. Was only little help on testing and wad dumped soon after testing in laboratorium.

4.25 squid/SquidFront.java

No further development needed. Have been used for testing some commands, but human control is too risky as we noticed.