Final Report 0.9

SQUID

Helsinki 5th 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
0.9	3.5.2005	First version (Esko Luontola)
0.95	4.5.2005	Improved debriefing, added project work flow (Esko Luontola)

Contents

1 Introduction

This document tells how the student project at the Department of Computer Science of University of Helsinki for building a new user interface for the SQUID magnetometer at the Department of Geophysics of the University of Helsinki went. The clients were Lauri Pesonen with his assistants Fabio Donadini and Tomas Kohout from the Department of Geophysics.

The Department of Geophysics uses a SQUID magnetometer for measuring the magnetism of rocks and meteorites. There was already a program for using the machine, but it had bad usability and the work process included the use of many external tools and Excel sheets to convert the measurement results to the desired format. The goal of this project was to make a better user interface for controlling the SQUID.

The name of the produced program is Ikayaki. The name comes from a japanese seafood - dried, grilled squid.

The project took place from 25.1.2005 to 6.5.2005.

2 Organization

The people related to this project are shown in Figure 1.

Name	**Role**	**E-Mail**
Mikko Jormalainen	Project Team	mtjormal@cc.helsinki.fi
Samuli Kaipiainen	Project Team	samuli.kaipiainen@cs.helsinki.fi
Aki Korpua	Project Team	aki.korpua@cs.helsinki.fi
Esko Luontola	Project Team (Manager)	esko.luontola@cs.helsinki.fi
Aki Sysmäläinen	Project Team	aki.sysmalainen@helsinki.fi
Lauri J. Pesonen	Client	lauri.pesonen@helsinki.fi
Tomas Kohout	Client	tomas.kohout@helsinki.fi
Fabio Donadini	Client	fabio.donadini@helsinki.fi
Juha Taina	Course Manager	taina@cs.helsinki.fi
Jenni Valorinta	Instructor	valorint@cs.helsinki.fi

Figure 1: The people who were part of this project

3 Project

This section discusses the progress of the project from its beginning to the end.

3.1 Work Flow

In this project was used the waterfall process model, where the stages were Project Plan, Definition, Design, Production, Testing and Installing. Here is how each of the stages went.

3.1.1 Project Plan

The project plan was written at the beginning of the project by the project manager (Esko Luontola), while others were working on the definition part of the project. The project plan was completed in two weeks without any problems. Right after that Esko wrote the HourParser program for managing the team's work hours, and the hours lists were up and running in a couple of days.

3.1.2 Definition

The project was started with a meeting with the client. They introduced to the project team the problem that they were having and gave the source code to the old program. The client had made some suggestions, how the old program could be improved, but it was right away understood by the team that the old program was totally fubar. Esko and Aki K were assigned to have a look at the source code.

The next meeting was held at the client's laboratory where they demonstrated the SQUID magnetometer. At this point the team had only a very faint idea of how the machine is being used. It might have been better to have a private meeting with the project team only before or between these meetings with the client, because during the next couple of meetings there were awfully many things that had to be went through, and it maybe slowed down the beginning of the project.

It was decided that the definition stage would include the creation of a user interface prototype, which made the stage longer than in usual projects, but on the other hand reduced the work in the design stage. The process model for designing the user interface was GUIDe (http://www.cs.helsinki.fi/u/salaakso/). There were held about three user observations with the client, and based on those was designed the user interface and three prototypes based on three goal-based use cases.

The user observations and initial UI designs were done primarily by Samuli, Aki K and Mikko. Aki S was sick for some weeks. It was agreed to have weekly work hours where everybody would work together in designing the UI. This was a successful arrangement. The final versions of the prototypes were made with Powerpoint and they were demonstrated to the client. The client had some problems in understanding the importance of the prototypes and why so much time was spent on making them. The team tried to explain that it is not possible to start writing code without making careful plans, or the end result will not be what the client wants. It would have been better, if the team had prepared better to demonstrate how the UI decisions made by the team were better than those that the client had suggested.

At the end of the definition stage had been produced three UI prototype use cases and written the requirements document. The programming language was decided to be Java instead of C (the project manager managed to push his will through :). There were plans to reuse parts of the old C code and link them to Java with Java Native Interface, but the old code was too incomprehensible. Late in the definition stage it was found out, that there were protocol specifications for the SQUID devices. When those were found, it was easy to dump the reuse of the old program code completely, especially so after Aki S had studied (in design stage) how to do serial communication in Java.

3.1.3 Design

The program was divided into subsystems and they were assigned for different persons to design. Esko designed the program architecture and Project Data classes, Samuli did the Project Explorer and Manual Controls, Aki K the SQUID Interface and Emulator and Device Configuration and Main Window components, Mikko the Measurement Sequence and Project Info and Details, Aki S the Serial Communication and Graphs. The file format for the program's project files was decided to be XML. The use of IntelliJ IDEA for generating Java GUI code was introduced.

The design document was written so that it would be easy to generate source code with Javadocs from it by using Latex macros. At the end of the design stage, there was held a FTR (Final Technical Review) for the design document.

3.1.4 Production

The production stage was started with a week of easter holiday. Some work was done during the holiday (Esko), but the coding started in full speed after that. Primarily everybody wrote the code for those parts that they had personally designed. In some cases it was necessary to share the work with others, so that the programming would be completed in time.

CVS proved to be an invaluable tool. IRC has been especially useful when writing and testing code, because there it is quick to ask for others to help. When creating the Java GUI, in some cases it was possible to use IntelliJ IDEA, with which it was easy to make complex gridbag-layouts in minutes.

The production stage became much longer than initially expected. Reasons for that could have been that the program was much larger than it was told to the team that typical student projects would be. Also some of the team members were not giving their full time to the project. Otherwise the production stage went according to plans without any unremovable obstacles.

3.1.5 Testing

At first the program was tested without the real SQUID machine. However, the use of an emulator was not successful, because the protocol documentation was not up to date with the reality. The documentation was inaccurate and there had been made some changes to the hardware after writing the documentation. What proved most beneficial in testing the controlling of the SQUID, was the monitoring of how the old program does things. A SerialProxy class was created to log the communication between the old program and the hardware, and between the new program and the hardware.

It had been planned, that JUnit would be used in testing the program. There was not enough time for doing that, so apparently only one JUnit test class was made (for the serial communication). It would be good if the program would be tested completely in the future. JUnit should be used to test the project data classes and the mathematical algorithms. More real measurement data is needed to be able to test that the measurements and calculations are working correctly.

During the testing stage the client gave some extra requirements, that were not included in the requirements document. It was possible to produce most of them, but it was time away from the testing. It would have been best to be able to test with the real machine every day some. The most beneficial way to work was, when changes to the program were made at the laboratory, so that they could be tested right away.

3.1.6 Installing

The program and its source code was handed to the client at the end of the project. The final version of the program was installed onto their computer. It was not necessary to teach them how to use it, because at least one of the clients had been using the program in the testing stage.

3.2 Amount of Work

The amount of work that the team did is shown in Figure 2.

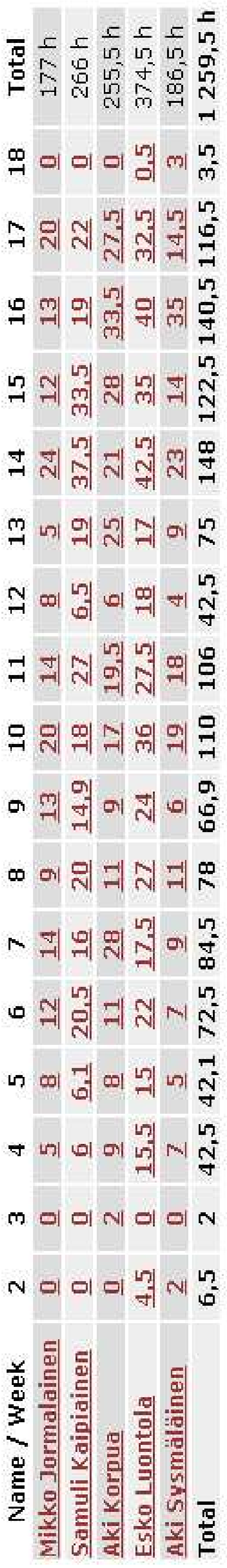
The estimated size of the program was a maximum of 13000 lines of code. The final size of the program is some 21500 lines of code. So the estimation that was made at the very beginning of the project (before the team even understood what they were doing ;) was more than 65% too low.

During the project was also produced HourParser, a program for managing the team's work hours. Its size is 1000 lines of code.

The final schedule for the project is in Figure 3.

4 Debriefi ng

The writings in this section are based on the answers that the project team members gave to a bunch of questions.



Name / Week	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Total
Mikko Jormalainen	0	0	5	8	12	14	9	13	20	14	8	5	24	12	13	20	0	177 h
Samuli Kaipiainen	0	0	6	6,1	20,5	16	20	14,9	18	27	6,5	19	37,5	33,5	19	22	0	266 h
Aki Korpua	0	2	9	8	11	28	11	9	17	19,5	6	25	21	28	33,5	27,5	0	255,5 h
Esko Luontola	4,5	0	15,5	15	22	17,5	27	24	36	27,5	18	17	42,5	35	40	32,5	0,5	374,5 h
Aki Sysmäläinen	2	0	7	5	7	9	11	6	19	18	4	9	23	14	35	14,5	3	186,5 h
Total	6,5	2	42,5	42,1	72,5	84,5	78	66,9	110	106	42,5	75	148	122,5	140,5	116,5	3,5	1 259,5 h

Figure 2: How many hours of work the team members did.

Task	Start	End	Days
Project Plan	25.1.2005	8.2.2005	14
HourParser	8.2.2005	9.2.2005	1
Definition	25.1.2005	1.3.2005	35
- Prototype		24.2.2005	
- Requirements Document		2.3.2005	
Design	1.3.2005	22.3.2005	21
- Design Document		22.3.2005	
Production	22.3.2005	14.4.2005	23
Testing	14.4.2005	28.4.2005	14
- Testing Plan		22.4.2005	
- Testing Report		29.4.2005	
Installing	28.4.2005	5.5.2005	7
- Final Report		3.5.2005	
- User Manual		xx.5.2005	
- Realization Document		xx.5.2005	

Figure 3: Final project schedule

4.1 Overview

The project has been a lot of work, some pain, some nice moments and reasonable enough results. Everybody in the team though that the theme of the program was both challenging and interesting. It was motivating to solve some real-world problems, even when it was out of our world. Nobody from the development team knew anything about the subject when the project got started started, and it took some weeks to understand what the people at Geophysics are actually doing and what it was that they really wanted. At start everything as a bit confusing and stressful, but towards the end the project team got more and more confident about the result.

The team was strong and people were supporting each other well. There were only five guys doing it all, and sicknesses and other courses took a huge amount of time from the project. The amount of work was much and it was shared unevenly. It took some time before the whole team was functional, but the team improved during the project and chemistry between the team members got better and better.

All team members learned working in a team. Some other things that were learned are: diplomatic negotiation skills with the client, the importance of meetings, the importance of design and that feeble feeling of things falling apart. Some technical things that were learned: Latex, CVS, more Java, new features of Java 1.5, IRC. The English of some students improved.

4.2 Project

The team has succeeded in making a program that works and the client appears to be pleased with it. The team mostly succeeded in everything, but failed to put the last effort and add a finishing touch to every step. The meetings with the client and communication could have been better, and as a result it was not possible to guess all of the requirements that the client would have wanted. The user requirements and the program could have been designed better. Testing the program was not as thorough as it should have been. The work did not keep up with the schedule, so in the end there was more work than in the beginning. The work load could have been distributed better.

What the project team thinks about the following:

- Goal-Derived UI Design:
  The UI would never have been even nearly as good if we would not have payed special attention to it. User observations gave us a better idea about client's workflow with the old software. The prototypes were also essential in designing the program.

- Meetings:
  Needed for the people to communicate. Things could work in plain irc/email, but not when everyone is committed enough to it, so regular meetings are needed.

- Hour management with HourParser:
  Great system, it's good that other members can see your hours right a way. The created program is much better than any previous ones. :) Other teams should try it.

- E-Mail:
  Good for coordination and communication. Mailing lists are good for communication within the project team. Much of the communication with the client was by e-mail.

- IRC:
  Was in important role when we were not working face-to-face, which was the case in about 90% of all work. Has been very helpful when many people are working on the same thing at the same time, especially so right before a deadline. It was also possible that while some are testing with the SQUID equipment, others stay home and program fixes for the found bugs, so that they can be tested right away.

- Java and Swing:
  Good choice here. It's a safe choice when the coders are not too experienced. It would never have been possible for all to learn C++ well enough to make a program half this complicated. Performance was not a program with current hardware.

- Latex:
  Hell and pain. Chainsaw internals massacre.

- Dia:
  At least the Windows version was buggy and the UI was designed to kill. Missed code generating features. A better tool for writing UML would be needed.

- CVS:
  Necessary for file management, even if a bit limited (can not rename/move files). The commandline version is clumsy - keep away from it. IDEA has a nice CVS front-end and propably so have many other IDEs.

- IntelliJ IDEA:
  Really well designed IDE for Java coding. The UI Designer makes the creation of complex layouts easy. For example it took for a first-timer only about 30 minutes to make the device configuration dialog's layout. Only stupid people write Java GUIs 100% by hand. :P

- JGoodies Looks:
  Looks better than the standard Swing look.

- Virtual serial ports:
  Helped at some point a lot. Was a necessity for the development of serial API.

- JUnit:
  Could have been used more. The team did not get too much in to it, but it worked well for serial testing.

- XML:
  Easy, effective and expandable. Was the best option for the new file format.

The waterfall project model that was used is a bit stiff, but it works. It is suitable for such short student projects as this. There would not have been time to use a more complicated model. The amount of documentation is a bit too much, though. A more flexible project model would be recommendable.

Those parts that were designed well, were produced according to the design. Those that were not designed, were produced more or less without plans. The user interface was produced accurately according to plans (use cases and prototypes) and so were also the Project data classes (ikayaki.*). The amount of time spent on designing could have been huge, but with this timetable the results were fair. If there had been time, the program requirements could have been more detailed (requires more communication with the client) and the program code could have been designed better (can be hard with GUI classes and their huge API).

Almost every phase was delayed, as expected. Maybe it would have hold better if the amount of work done by everyone were constant (20 h/week) all the time. Always someone (sometimes many) didn't do their jobs when supposed to; nothing much to help it, as everyone had other things than this project to do. Morale was low at times, which is inevitable in such a long project. The client also gave some extra requirements, which required some time to produce (luckily not too much). As a result, there was not enough time to test the program properly.

The legacy C code was a nightmare. If the team had known about the existense of protocol documentation, the old code could have been dumped sooner, because it was pretty hard to read and had no documentation. It was a good choice to start everything from ground. Using the old code would have created too many new risks and slowed down the project.

The protocol documentation was incomplete and did not mach the reality, so creating an emulator was not very useful. The created SerialProxy class gave much undocumented information about the protocol, so looking at how the old program does the things on protocol level made the day. Hardware was actually good and safe to use when you learned it, which helped a lot when testing and cleared errors in the documentation.

4.3 People

The client has been very interested in the project, which is good. There should have been more communication with the client, so that the team could have explained better things such as the process of software development. In the beginning the information from the client was sometimes inconsistent and the team did not always known who to follow, but this improved towards the end. The client was committed to project and ready to offer their help and time as much as they could. Special thanks to Fabio for being of much help.

Apparently most of the team members (4 out of 5) had never been talking that much English. The use of English slowed down the process in the beginning, but later on it was just a minor issue. Sometimes it was a bit hard to understand everything and sometimes it took time to find the right words, but it did not affect the project in the end. In overall the use of English has been good practice for the future.

The instructor did her work well, silently observing the project team when everything was going well and stepping in to direct when time was running out or the team was going to make a mistake. At very beginning there was some confusion about the authority between her and the project manager, but that was then sorted out. She kept the project and the group on trail and emphasized things that got less attention. She could have been a bit more relaxed on some issues when the internal pressure of the group was already high. But on the other side the pressure tolerances of the team got much better and more ready for real world challenges. In general she was nice and fair.

4.3.1 Team members' evaluations of themselves

Mikko Jormalainen
- Poor.

Samuli Kaipiainen
- Tried to do my jobs on time, some (but not many) failings to do so though, tried a couple of times to silently keep the project in one piece, lost some (or at few times, a lot) morale in the end.

Aki Korpua
- Lazy parasite. I was really working hard, but WoW and and other intrests took too much time.

Esko Luontola

- Maybe I worked a bit too much, but the work does not disappear by itself. Running a project team was new for me and what made it more difficult, was that everybody in the team were strangers at the beginning of the project. I'll try to improve in delegating work to others in the future. I know that I'm overconscientious.

Aki Sysmäläinen
- At the beginning my use of time for the project was minimal but towards the end it got better. I was eager to take the lead when it was quiet on that front. I at least tried to add some diplomacy to client-project group relations.

4.3.2 Team members' evaluations of each others

Mikko Jormalainen
- That one guy. Could have been more in contact with other members. Did his work well anyway.
- Didn't have that much interest in the project, or so it seemed at the beginning, but not so much towards the end. Was most always ready to have a meeting of some sort. Had some weird problems with cvs updating frequency x)
- Communication was lagging quite much. Does he even have 24/7 internet access at home?
- Did his part in documenting. Could have done more coding. He was also a bit distant from the group from time to time.

Samuli Kaipiainen
- Does excellent work when he is into it.
- Did what was assigned to him and did it well.
- Good.
- Did a great job with the project explorer which is an achievement of usability. But when he got full of the coding the whole group got a bit affected by that. His sense of humor and analytic attitude on problems were invaluable to group and the project.

Aki Korpua
- Fast worker, didn't care so much for perfection :) Did his part even when tired and out of morale. Had some good sympathy for the clients, which drove him to try and make a working final software.
- Was also good in what he did. Did a good job in digging into the SQUID and the old program. Had time for the project in spite of WoW. :)
- Good.
- He did a lot of work with interface and emulator. Some of his emotional reactions at the beginning distracted other group members. His support and hard work kept the group going during the black spots.

Esko Luontola
- Hero. Kept our project alive. Could have taken more leader role at the beginning.
- Kept the project in one piece. Did something like 80% of all coding, and was good at it too. Didn't care (or so it seemed) about work hours being accumulated to him. Made some vague changes to others' codings =)

- Excellent.
- At the start it took some time of him to take the lead but after that he’s work has been pretty convincing. His contribution to coding was huge and he kept the code together and fixed and added a lot to other guys coding.

Aki Sysmäläinen
- Sleepyhead, hehe. Dont try to do all courses at same time. Good social skills and does his work good.
- Had many other project going on at the same time 8) But, did his part in the end, such as the graphs for the program, which would have been a shame not to have. Took the lead sometimes, when things didn’t go forwards.
- Was a bit too busy with life outside the project. Was good in asking questions for example when designing the UI. Also good in communicating with the client.
- OK.

Appendix 1. Hours of Mikko Jormalainen

REPLACE WITH PDF

Appendix 2. Hours of Samuli Kaipiainen

REPLACE WITH PDF

Appendix 3. Hours of Aki Korpua

REPLACE WITH PDF

Appendix 4. Hours of Esko Luontola

REPLACE WITH PDF

Appendix 5. Hours of Aki Sysmäläinen

REPLACE WITH PDF