Pekka Mäkiaho and Timo Poranen (eds.)

Software projects 2012-2013

UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 23

TAMPERE 2013

UNIVERSITY OF TAMPERE
SCHOOL OF INFORMATION SCIENCES
REPORTS IN INFORMATION SCIENCES 23
OCTOBER 2013

Pekka Mäkiaho and Timo Poranen (eds.)

Software projects 2012-2013

SCHOOL OF INFORMATION SCIENCES
FIN-33014 UNIVERSITY OF TAMPERE

ISBN 978-951-44-9238-9

ISSN-L 1799-8158
ISSN 1799-8158

Preface

This report contains project stories of 13 software development projects from academic year 2012-2013. The students came from Project Work (TIEA4) and Software Project Management (TIETS19) courses. The stories describe what kind of experiences groups got during their project and what was the software product that came out from the project. In the end of each story there are project statistics.

Table 1: General project statistics.

Project	Type	Client	Dev. Mod.	Group	Hours
Avainsiirto	WWW	Other	Scrum	3+3	1004
Alkeismatematiikka	Appl.	University	Scrum	3+4	1040
Eedu	Android	Company	Scrum	4+4	1229
Gesture Recognition	Appl.	University	Scrum	4+4	927
Logistiikkaketjun kirjaus	Android	Company	Scrum	3+4	1084
Majava4	WWW	University	Scrum	2+4	847
Math.fi	WWW	Other	Scrum	3+4	1028
MobSec	Android	Other	Waterfall	3+4	1152
Pricing Tool	WWW	Company	Scrum	2+5	1163
Meta Review Tool	WWW	University	Scrum	2+4	1118
Smart Lightning	Appl.	University	Scrum	3+5	1266
Virtual Patient	WWW	University	Scrum	4+5	1484
VirPro	Appl.	Company	Scrum	3+4	1077

Table 1 gives an overview of the projects. For each project, there is project type (WWW, Android or standalone application), client (University, Company or Other non-commercial client), development model, group size (number of managers + number of developers) and working hours of the project.

Although 12 projects applied Scrum ("Scrum-but") development model, there was usually some differences when compared to standard Scrum: no face-to-face daily meetings, no fixed length iterations, many managers in the team, no sprint retrospective…

Table 2 contains general course statistics (number of projects and usability teams, number of students in the courses and average project size in working hours) starting from year 2005.

Table 2: Course statistics 2005-2013.

Academic year	Projects	Usability teams	PW students	SPM students	Average project size
2005-6	19	1	98	8	1008
2006-7	18	2	87	34	1089
2007-8	14	1	70	29	997
2008-9	10	1	60	39	1643
2009-10	15	1	80	34	1151
2010-11	13	1	70	27	1230
2011-12	14	0	67	30	1331
2012-13	13	0	54	39	1109

School of Information Sciences (SIS) offered version control services (subversion) and Redmine-project management tool for all projects. Balsamiq was used in many projects for user interface design. SIS's projectWiki was used to maintain course and project related documentation and guidelines: https://projectwiki.sis.uta.fi. The wiki also contains some articles on project management and project management tools, including lists of end-products currently in use, course related publications and course related videos:

- https://projectwiki.sis.uta.fi/wiki/Finished_projects
- https://projectwiki.sis.uta.fi/wiki/Course_publications
- https://projectwiki.sis.uta.fi/wiki/List_of_project_videos

Course staff thanks our clients and students for great projects!

Pekka Mäkiaho and Timo Poranen

Tampere, September 2013

Interactive website for teaching chess - Avainsiirto

Overview

Avainsiirto is an interactive Finnish web service aiming to teach the game of chess. Avainsiirto contains a set of chess problems the users try to solve. While solving the problems the users learn the rules of the game, how to play chess and also how to make smart tactical choices.

Users aren't forced to register to the service to be able to solve the chess problems. If they choose to register, the service will track their progress and calculate a strength number for each user depending how well the user has solved the problems.

Avainsiirto is mainly aimed at young students still in primary school. While designing the service it was also important to make sure the service feels welcoming to older students and adults.

Image 1: *The main chess problem interface*

Since the client Shakkilinna made all the necessary materials for the actual chess problems, it was up to the project team to design both the visual and technical elements of the service, according to customer's needs and requests. The main requirement involving the visuality of the site was to remember the user target group.

The technical requirements sprung from the versatility of the chess problems: the need to be able to add new problems (as seen in image 2) as well as modify the existing ones, and the need to track the user's progress.

The first key feature of the site is the chess problem interface (image 1). The problem includes all necessary texts, the chess board and the chess pieces. Texts include the description of the problem, the hint and feedback to both correct and wrong answers. The chess board has chess pieces on it and one of the pieces must be moveable when solving the problem, both by click-and-click and drag-and-drop. The board and the pieces must understand the basic rules of how each chess piece moves on the chessboard.

Image 2: *Creating chess problems has never been easier!*

After the user moves one of the pieces feedback is given: colors on the board may change and the exercise helper piece gives information if the user succeeded or failed the exercise. If the user is registered to the service, one must get points according to how well one has solved the problem. The service must remember which problems are already solved and which are not, as well as the strength number of the user.

The second key feature is the ability to make new problems and follow the progress of the users (as seen in image 2). The customer, or administrator, must be able to sign in to the service and access the html-based problem editor.

Organisation and management

The group was initially split into two different teams. The first team handled the coding part of the project. This included the JavaScript section that made it possible to have interactive content on the site, such as the interactive chess board. The code team was also responsible for the backend logic of the site. The second team handled the design and implementation of the user interface and product testing.

The project decided on a one meeting per week minimum. The whole project team would participate in the weekly meeting which was to serve as a synchronization point. At the beginning of the project there were UI meetings, but after the main guidelines were set they ceased to exist. In the later phases of the project there were also code workshops.

Our organisation is explained in the next table. Lauri Maasola has been our administrative manager in charge of common tasks. Panu Hokkanen was mainly responsible for the user interface implementation. Along with Mikko Kokotti, all the managers observed the coding process and helped the development team when it was necessary.

Management	**Development**
● Lauri Maasola	● Tommi Perkola
● Mikko Kokotti	● Jukka Suorsa
● Panu Hokkanen	● Siiri Tammisto

Table 1: *The team*

Lauri Maasola

Lauri worked as the administrative project manager, which included things like keeping everyone in touch with each other, reserving meeting rooms and organizing weekly meetings and review sessions. Lauri also helped coding the actual chess game and implemented the Kinetic JS framework into the project.

Mikko Kokotti

At the beginning of the project, Mikko introduced and taught the basics of CakePHP to team developers. Mikko built a prototype of the problem editor with Java, which helped to design and develop the actual editor with assistance of Tommi Perkola.

Manager Panu Hokkanen

Panu’s main responsibility has been UI designing and implementation. After the initial start of the project, Panu helped getting the actual chess game working properly. He has also helped with general CakePHP problems the team encountered and

implemented the Javascript editor to the site (since it was a separate piece of Javascript before).

Tommi Perkola

Tommi wrote the code to the games javascript basics at the start of the project, which gave a good support to write the rest of the code to the game. Most of the final editor is developed by Tommi.

Jukka Suorsa

Jukka was the main programmer responsible for the backend part of our project. He learned to use the CakePHP framework and implemented most of the features concerning user management, news section and general needs of the site.

Siiri Tammisto

Since Siiri was the only team member without programming skills, her main responsibility was to test the site while others programmed it. Siiri tried to look at the site from the user's perspective at all times, and to make remarks about possible user interface problems.

Methods and tools

Name	Version	Purpose	Usefulness
Eclipse IDE	Indigo/ Juno	Main development	Extremely useful
CakePHP	2.2.2	Framework for PHP	Extremely useful
Kinetic JS	4.3.3	HTML5 Canvas JavaScript framework	Extremely useful
jQuery	1.8.3	JavaScript library	Extremely useful
XAMPP	3.1.0	Apache server and MySQL support	Extremely useful
Subversion		Version control	Extremely useful
Facebook		Communication	Very useful
Redmine	1.3.1	Project information sharing, e.g. features, meeting minutes, etc.	Very useful
Photoshop	CS5	Creating sketches and UI	Very useful

		elements	
Google Docs		Document creation	Very useful
Balsamiq Mockup		UI design	Very useful
Internet Relay Chat		Communication	Useful when people were actually using it

Table 2: *Methods and tools*

Project phases and development model

Our development model was Modified Scrum, since we didn't have any daily meetings. Instead, we had weekly meetings and additional workshops during the project. Workshops were arranged when they were needed and usually were about sharing information or coding some feature with more than one people.

At the start of the project the main way of communication was the forum Redmine supported and of course Email. After a while, the team found it to be too slow and incoherent, so we changed our primary communication method to Facebook, where we created a new group for our project. Facebook proved to be quite useful, mainly because of it's addicting features. Since everyone is using Facebook on a daily basis, team members could quickly communicate with each other. Our secondary method of communication was the Internet Relay Chat (IRC), which proved to be useful for those that tended to use it regularly outside the project.

Our project's phases for the code team were mostly practicing, implementation and integration. The project was "kickstarted" with the code camp, where the initial project was made and we started to implement the first main features, the user system, the news section and the interactive chessboard. The following large milestones included first working prototype of the solvable chess problem and the chess problem editor. These large features included a lot of integration, since they were developed separately outside of the main site. The process of learning new technologies and integrating ultimately lead to the lack of proper unit tests, but on the other hand, we managed to achieve more than the minimum goal of our project.

Experiences

Foreseen risks

RISK	**ANALYSIS**
Technology problems / Tools and skills	Countermeasure Studying new techniques and teaching others.

	Analysis The required technologies were learned. However, required tools have been hard to install for some members, especially XAMPP and Eclipse.
Quitting team members	Countermeasure Motivating team members. Delegating working hours as equally as possible. Analysis One team member quit and the reason remains unknown. After the disappearance, the managers participated more in the coding process to compensate loss of working hours.
Motivation problems	Countermeasure Motivating the team members. Analysis It was hard to keep the team motivated at the times, since we didn't have any real incentives. In a university project like this, it is important that the person attending the course is open and motivated to learn new technologies and working with the team.
Working and studying during project	Countermeasure Personal schedule planning. Analysis Countermeasure for this risk would have been better delegation of jobs.

***Table 3**: Foreseen risks*

Unforeseen risks

RISK	**ANALYSIS**
Lack of testing	Things did not proceed as fast as we had initially hoped, so the software testing phase was delayed. Since the CakePHP was new framework for every team member, we thought we would postpone testing so people could learn the framework first. In hindsight, managers should have set the testing environment

	to the project and teach it to the project members.
Different team members working different hours	Problems occurred mostly on the integration phase, because there was some lack of communication. To counter this, we started to have regular code meetings.
Scheduling difficulties	Since some project members had jobs and working hours scheduling weekly team meetings was sometimes hard.

Table 4: *Unforeseen risks*

Statistics

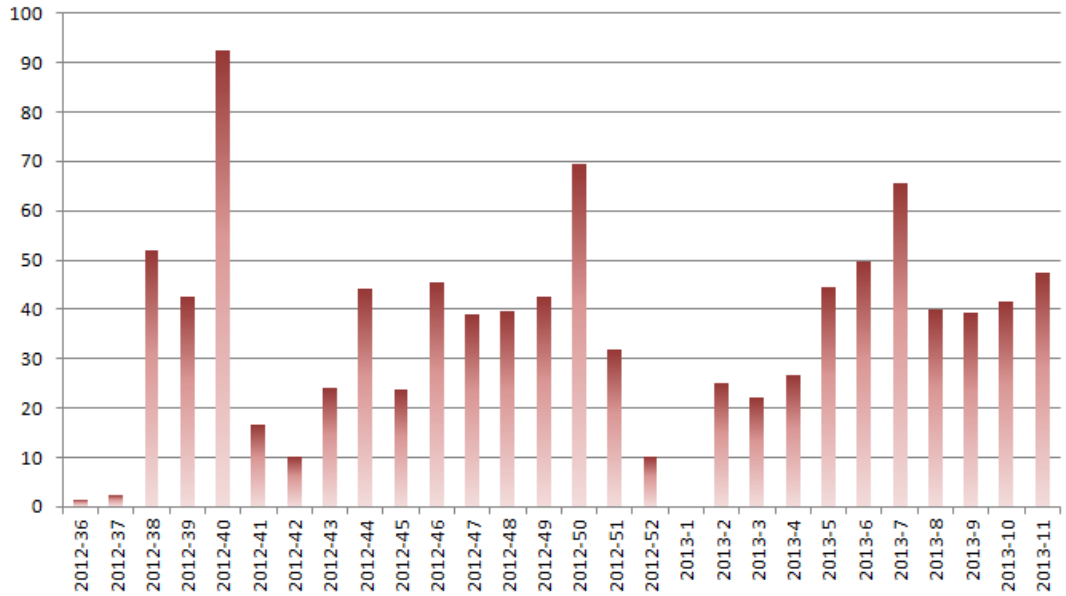


Diagram 1: *Working hours by week.*

Team size	Dev. model	Start date	End data	Days	Hours
3+3	Modified Scrum	19.9.2012	17.3.2013	180	1003.50

Table 5: *General project information.*

Activity	Planning and management	Req. specification.	Design	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	384.25	3.00	65.00	413.25	14.00	5.00	7.75	83.50	27.75	1003.5
%	38%	1%	6%	41%	1%	1%	1%	8%	3%	100%

Table 6: *Group effort by activity.*

Requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
-	-	20	21	-	9

Table 7: *Requirements and high-level design outcomes.*

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
-	-	1	-	-	-

Table 8: *Design outcomes.*

Document	Pages	Versions
Preliminary analysis	6	1
Project Plan	12	3
Test plan	4	1
Final report	X	1
Project's story	X	1
Weekly reports	X	1

Table 9: *Documents.*

Language	PHP, JavaScript

LOC	7567
Classes	16
Functions	91
Code revisions	220

Table 10: *Codelines.*

Alkeismatematiikan oppimisympäristö - Learning environment for elementary mathematics

Overview

Learning environment for elementary mathematics (AlkMat) is for children age 6 to 8 (from pre-school to 2nd grade) to help children learn quantities of objects, numerical symbols and basic arithmetics. AlkMat offers a new method to learn mathematics by offering Kinect motion sensing input. AlkMat can also be used by mouse.

AlkMat saves results of the tests and exercises and based on that information it generates exercises based on children's current skill level.

AlkMat helps teachers to identify children who need special attention in learning numbers and early mathematics. Teachers can also test children's level of skills and later follow how children develop their skills in mathematics.

Our client was research center TAUCHI (Tampere Unit for Computer-Human Interaction) in University of Tampere. The client representative was Jussi Okkonen. Pedagogical expert in mathematics was Sari Yrjänäinen from University of Tampere.

Illustration 1: *Main menu*

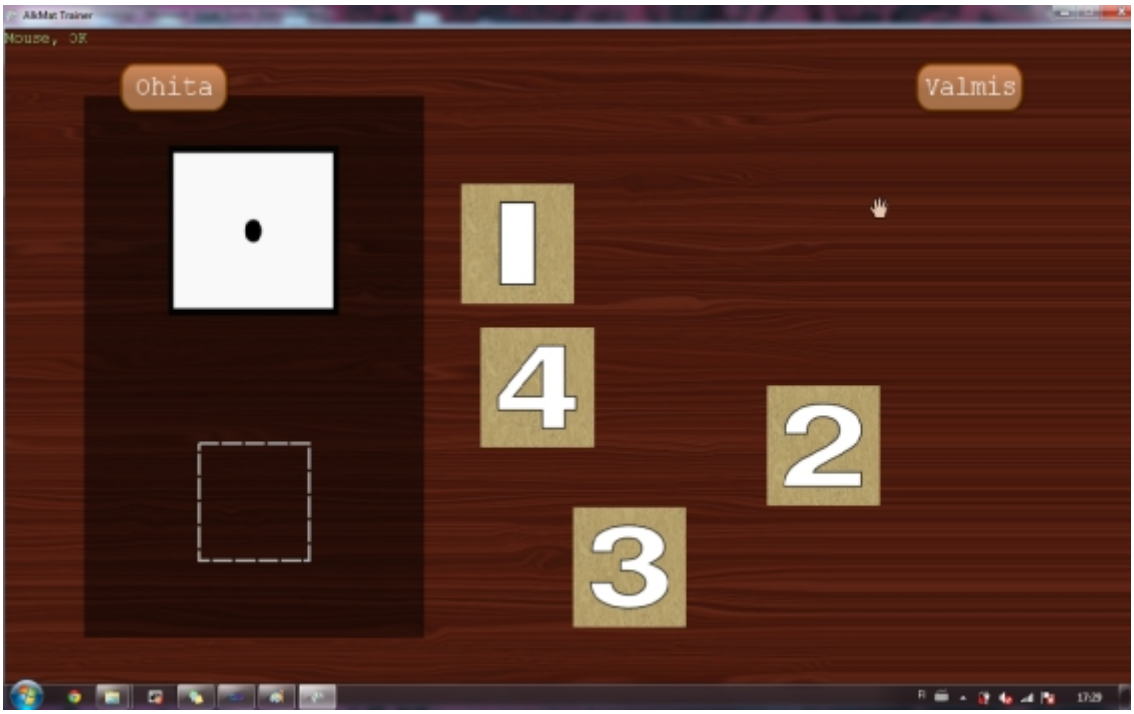


Illustration 2: *Object exercise*

Organization and management

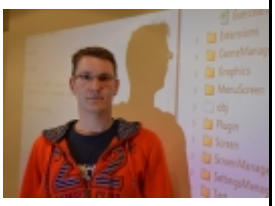
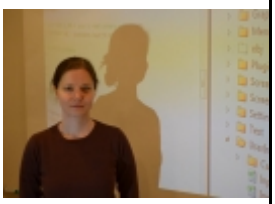
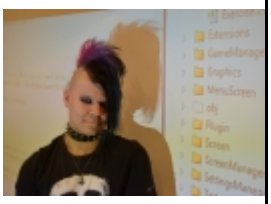
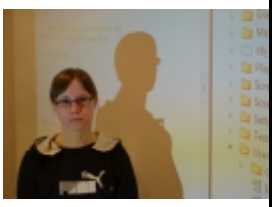
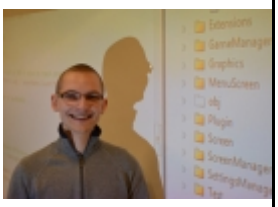
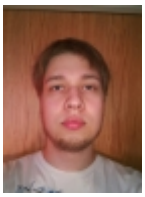
 Johan Björn project manager	 Teemu Jyrkämä project manager	 Anu Kauppi project manager	 Jude Laine database, audio
 Krista Talvio design, usability, graphical resources	 Joel Luukka design, code	 Ville Valtonen design, code	

Illustration 3: *Team members*

Methods and tools

We used mainly IRC and UTA email for communication. IRC wasn't really a perfect solution for communication because everybody didn't use it. Also it's impossible to use it 24/7 because it doesn't save messages in the screen and one has to use a separate log file. Although during the last four or five sprints, we had quite a lot of conversation among the core coding team on IRC. It was really useful way to communicate and ask little questions concerning features. Documents were stored in Google Drive which worked very well. We used Redmine for project management tool and sharing files and Doodle for scheduling.

For the development we used these tools:

- Microsoft Visual Studio 2010 Professional SP1
- Microsoft XNA Framework 4.0
- Kinect SDK 1.0.3.191
- Microsoft .NET Framework 4
- Microsoft SQL
- Subversion

For the software project we used Scrum methodology with some modifications. We had weekly meetings on Tuesdays. In these meetings we discussed tasks of previous week and planned next tasks for next week.

Project phases and development model

We planned 9 two week sprints and one 3 week sprint at the beginning of the project. Later we added one sprint because we decided to continue coding for 2 weeks to get everything ready. Adding one sprint didn't change the deadline of the project. It worked very well to have more time for finalizing the code.

We had weekly meetings on Tuesdays. We also had some workshops especially in the beginning of the project. In these workshops we planned, designed and coded together which was very good way to work.

Project timetables

First project meeting	18.9.2012
Preliminary analysis	26.9.2012
Project plan	12.10.2012
Sprint 1	24.9.- 7.10.2012
Sprint 2	8.10.-21.10.2012

Sprint 3	22.10. - 4.11.2012
Personal report I	7.11.2012
Sprint 4	5.11. - 8.11.2012
Sprint 5	19.11. - 2.12.2012
Midterm presentations	28.11.2012
First review	28.11.2012
Sprint 6	3.12.2012 - 16.12.2012
Sprint 7	17.12.2012 - 6.1.2013
Personal report II	15.1.2013
Sprint 8	7.1. - 20.1.2013
Sprint 9	21.1.2013 - 3.2.2013
Sprint 10	4.2.2013 - 17.2.2013
Sprint 11	18.2.2013 - 3.3.2013
Finishing coding	17.2.2013 - 13.3.2013
Documentation and finishing	18.2.- 15.3.2013
Final presentation	13.3.2013
Project story	15.3.2013
Project CD	15.3.2013
Final meeting	28.3.2013
Personal report III	03.2013

Experiences

Due to staff changes our requirement specification phase delayed. It created a slow start for the project because we got requirements as late as in November. Hence we had to prioritize the requirements carefully. There could have been more testing before the end of the coding phase if we had more time or more coding resources available.

Some minor problems occurred when project team members could not attend to meetings due to illness, work or just because it is always hard to pick a suitable time slot for everybody.

It would have been good to have one other communication tool for discussing about meeting times and other practical issues. We used email and it wasn't flexible enough for that kind of purpose. Good meeting memos are also important to have so that everybody can check what they should do next.

Illustration 4: *Weekly meeting*

Statistics

Team size	Dev. model	Start date	End date	Days	Hours
3+4	Mod. scrum	18.9.2012	28.3.2013	192	1040

Table 1: *General project information*

Activity	2012-9	2012-10	2012-11	2012-12	2013-1	2013-2	2013-3	**Total**
Planning and management	54.50	76.50	79.50	30.00	43.50	47.50	29.50	**361.00**
Requirements specification	1.00	3.00	20.50	14.50	1.50			**40.50**
Studying	41.00	18.00	18.00	6.50	14.00	14.50	5.00	**117.00**
Code		26.00	55.00	6.50	50.00	35.00	19.50	**192.00**
Other	10.00	10.50	26.75	18.00	12.50	5.00	12.50	**95.25**
Design		6.00	36.00	25.00	12.50	14.00	20.00	**103.50**
Integration and testing			4.00	4.00	10.00	2.00	52.50	**72.50**
Review		4.00	10.00		5.00	7.00	8.00	**34.00**
Repair					0.50			**0.50**
Total	**106.50**	**144.00**	**249.75**	**104.50**	**149.50**	**125.00**	**160.00**	**1042.25**

Table 2: *Hours by activity*

Week	Hours
2012-36	13
2012-37	14
2012-38	35,5
2012-39	44
2012-40	47
2012-41	33,5
2012-42	23
2012-43	29,5

2012-44	30,5
2012-45	54,5
2012-46	48
2012-47	52,5
2012-48	78,25
2012-49	32
2012-50	57,5
2012-51	6
2012-52	6
2013-1	15
2013-2	41,5
2013-3	43,5
2013-4	31
2013-5	24,5
2013-6	29
2013-7	51,5
2013-8	8
2013-9	51,5
2013-10	77
2013-11	56
Total	1038,25

Table 3: *Weekly hours*

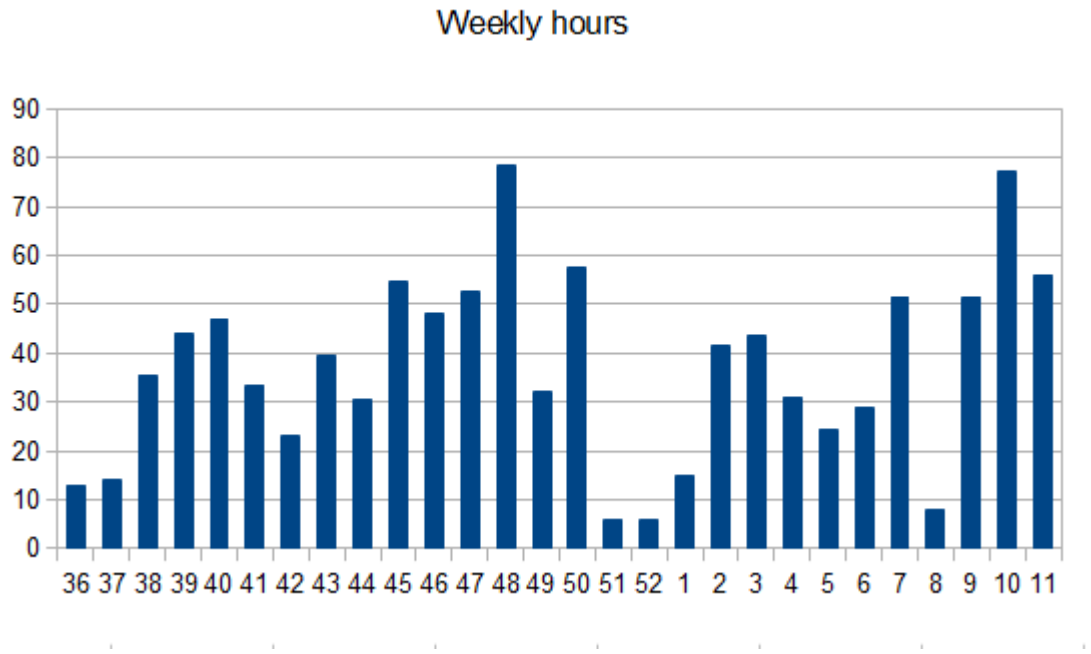


Illustration 5: *Weekly hours*

Document	Pages	Versions
Preliminary analysis	5	1
Project plan	23	10
Requirements specification	17	6
Test plan	16	5
Test report	8	1
Exercises and exercise generator	3	1
User manual	7	1
Project story	8	1
Final report	19	4
Demo videos	2	2

Table 4: *Documents*

Code Revisions	LOC	Comments	Source and property files	Features	Support	Bugs
206	10 856 C#: 10107 XML: 749	2208	155	81	37	5

Table 5: *Code*

Project Eedu

Overview

We made a Java library that allows running Android application to change device where it is running. If application is running on device 1 it can be "moved" to device 2 and shut down in device 1. Library was targeted to be used in Math Elements game made by Eedu Ltd and thus we also made a simulator to simulate library related functionalities of the game. Library also allows to transfer required information of Math Elements game to start the game in the same state as it was when running on the original device.

Our product contains two major parts: Framework and PhoneGap application to simulate Math Elements game. Framework is split into two minor parts: The core of Framework and Socket Server communication. PhoneGap application also splits into two minor parts: The application itself and a PhoneGap plugin. Google App Engine ChannelAPI socket server is also needed but it is not core parts of our product.

PhoneGap application is a simple Android application made by using PhoneGap framework. It has a few functionalities: 1. Show QR code to change device, 2. Show QR code to start multiplayer game and 3. Send example message to other device.

Image 1: PhoneGap application running on Android Emulator. QR code is used to pair devices.

Organization and management

Name	*Role*	*Main tasks*
Markus Leinonen	*Project Manager*	*general management, programming and testing, contact person with client, PhoneGap & Java mentoring*

Pyry Kallio	*Project Manager*	*general management, SVN, programming and testing*
Da Ke	*Project Manager*	*general management, projectwiki, redmine stuff, user interface, documents and testing.*
Yanzhao Wen	*Project Manager*	*general management, UML and architecture*
Ville Siltala	*Project Member*	*QR code coding*
Eashan Salhotra	*Project Member*	*PhoneGap coding and Testing documentation*
Golnaz Sabet Nejad	*Project Member*	*IntentFilter/Launcher coding*
Pengfei Lv	*Project Member*	*Connection with Google App Engine(including socket server and socket client)*

Table 1: Team members' roles and main responsibilities.

All of the team also took part in documentation and design/planning of the project.

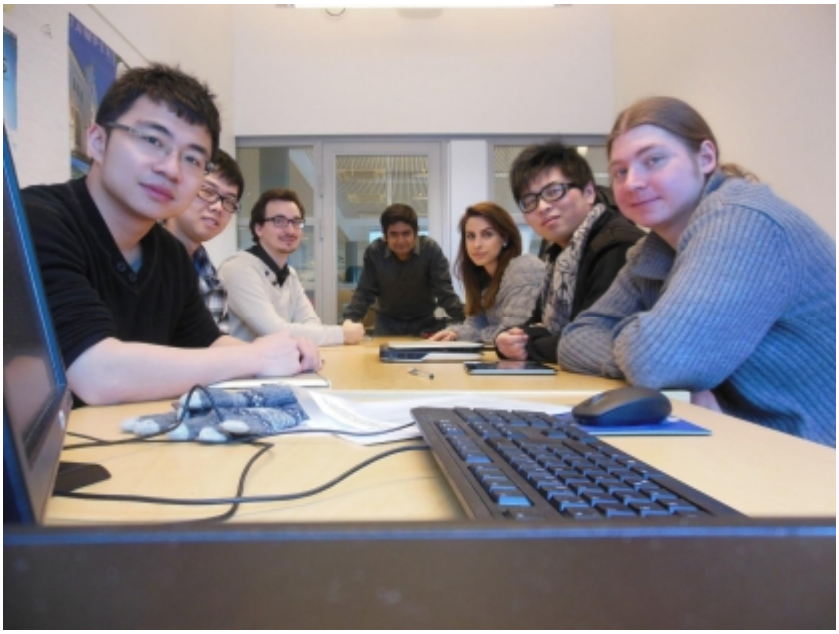


Image 2: Project Team. From left to right: Yanzhao Wen, Pengfei Lv, Markus Leinonen, Eashan Salhotra, Golnaz Sabet Nejad, Da Ke and Pyry Kallio. Not in this picture: Ville Siltala.

Methods and tools

Tool	Purpose	Usefulness
Eclipse IDE	Main development	Very useful
Android SDK	Tools needed to compile and develop Android code	Very useful
GAE SDK	Tools needed to develop Google App Engine applications	Very useful
Subversion	Version control	Very useful
Redmine	Project management and information sharing, e.g. features, meeting minutes, etc.	Useful
Skype	Real-time communication between team	Useful
Google Group	Communication between team	Useful
Google Drive	Document creation and storing	Useful

Table 2: Methods and tools.

Project phases and development model

Development model for this project was modified Scrum. As a student project it is impossible to use "pure" Scrum. Project team had weekly meetings but not daily Scrum meetings. Project schedule was divided into phases and phases further into shorter sprints, but sprints were actually only written in schedule but not supervised as the management concentrated on phases. Review meetings with project supervisor were scheduled in the ends of development phases, and single sprints were not reviewed at all. At the very beginning of the project a lot of time was be used to studying and learning new programming environments and there were not sprints at all in the beginning. We did a lot of demos as we didn't know if and how all requirements could be achieved.

Our project was divided into four phases. First phase was studying: as none of our project members was not familiar with Android development and not all too with Java, we had to study new technologies to start the development. Second phase was a demo phase: during it each team member had her/his own area of expertise of which she/he developed a working demo. Third phase was integration and upgrading phase: during this phase we integrated separated demos into a single working framework.

Fourth phase was testing and documentation: this phase was meant to test the functionality of the framework and write needed documents.

Preliminary Analysis Meeting	Review of the Preliminary Analysis	28.9.2012
Project Plan Inspection	Review of the Project plan with Pekka Mäkiaho	9.11.2012
Review I (Checkpoint I)	Review of the updated project plan and project progress Pekka Mäkiaho	4.12.2012
Review II (Checkpoint II)	Review of the project progress with Pekka Mäkiaho	8.2.2013
Review III (Checkpoint III)	Review of the project progress with Pekka Mäkiaho	7.3.2013
Final Report Meeting	Review of the Final Report with Pekka Mäkiaho	21.3.2013

Table 3: Milestones and checkpoints

Document name	**Revisions**	**Pages in latest revision**
Preliminary analysis	1	16
Project Plan	6	21
User stories document	1	6
Testing template	4	50

Table 4: Course documents

Experiences

Foreseen risks

RISK	ANALYSIS
Developers' lack of experience in used tools and methods	Counter measure: Adequate time to let developers introduce themselves in used technology. Possibly organize workshops. Analysis: All of the team members learned necessary skills.
Quitting team members	Counter measure: Good team atmosphere, fair management, motivating developers by changing tasks if needed and avoiding overburding any developer. Analysis: No member of the group quitted and a group spirit became even stronger towards the end of the project.
Communicating problems	Counter measure: Using of clear language, using different communication methods efficiently. Analysis: Lack of communication was some kind of a problem for the whole project, both between managers and developers and between managers.
Lack of testing devices	Counter measure: Trying to share available testing devices fairly, so no one need to develop the whole project by using Android emulator. Single developer may have a test device for one week at time, and then it is given to the next one. Analysis: The problem was quite marginal as not all

	team members developed Android projects in the start of the project. At some phase Eedu Ltd took devices away but returned them for testing phase.
Lack of testing	Counter measure: Introducing the team into unit testing early enough. Keeping statistics of untested features and code Analysis: It took for a while before team members started to write unit tests, but they are however written. Not all necessary tests could be written because we run out of time, but most of the functionality is however tested adequately and the client is satisfied.
Managers' lack of experience in management	Counter measure: Prepare for managemental problems. Receive extra time for managemental issues. Analysis: Managers had sometimes Skype meetings, and meetings with course staff were prepared well. As there were four managers, it was possible to share managing tasks between managers, but in many cases Markus was the only manager able to solve some problems and he was also that manager who had most obstacles to attend course meetings, so extra time was needed for managemental issues.
Things might be impossible to implement	Counter measure: Use modular development model. Plan modules in a way they don't interfere each other. Be prepared for problems. Analysis: No such things emerged which had prevented to reach the minimum goals of the project.

Table 5: Foreseen risks

Risks not foreseen

RISK	ANALYSIS
Different team members working different hours	Even though all team members will get minimum working hours, the work was divided unequally as some members have got lot of hours in the late phase of the project and some others have worked much throughout the whole project. Some of the developers are getting maximum hours as some are getting minimum, but this was also predicted when estimating working hours in preliminary analysis phase. Among the managers, Markus has got more working hours than other managers, as he has been the critical manager who has the best knowledge of the application usage and context, and technologies.
Team members getting work	Working during the project has caused problems when scheduling meetings. As mentioned above, Markus has had the best knowledge of the project, but he has also had difficulties in attending meetings as the distance between Tampere and his home town Pori is quite long.

Table 6: Unforeseen risks

Statistics

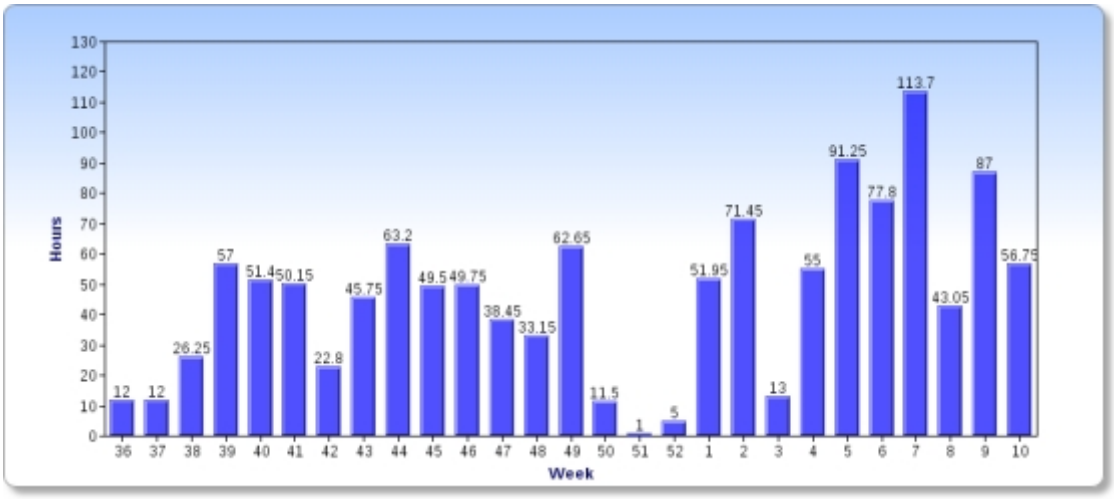


Chart 1: Weekly working hours.

Team size	Dev. model	Start date	End date	Days	Hours
4 + 4	Scrumbut	12.9.2012	15.3.2013	124	1252.50

Table 7: General project information.

Code	Planning and manage-ment	Other	Studying	Integration and Testing	Review	Design	Repair	Requirements specification	Total
182.40	**449.80**	**59.50**	**235.15**	**158.10**	**26.30**	**80.00**	**1.00**	**36.50**	**1228.75**
15 %	36 %	5 %	19 %	13 %	2 %	7 %	0 %	3 %	100%

Table 8: Group effort by activity.

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
22	-	4	-	-	-

Table 9: Requirements and high-level design outcomes.

Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
2	2	1	2	2

Table 10: Design outcomes.

Document name	Revisions	Pages in latest revision
Preliminary analysis	1	16
Project Plan	6	21
User stories document	1	6
Testing template	4	50
Final report	3	16
Project story	1	9

Table 11: Documents.

Language	Java
LOC	947
Classes	6
Interfaces	8
Code revisions	23

Table 12: Codelines.

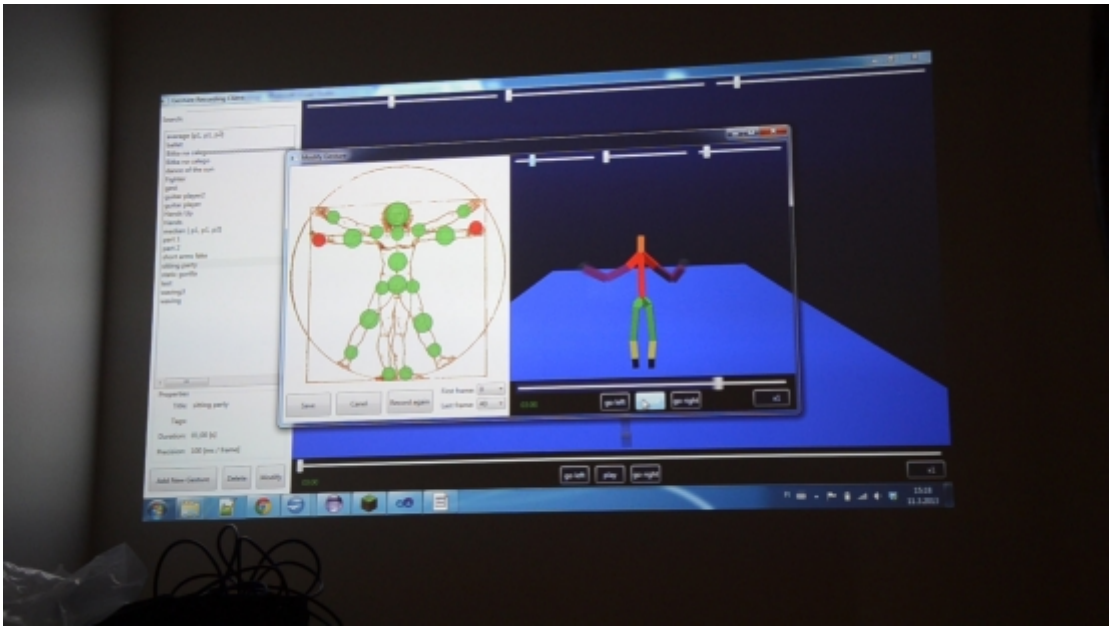
Gesture Recognition

Overview

The basic idea of the project was to develop general gesture recognition software, which can be integrated to different applications through an interface with minimal coding effort. This project intends to reduce the time & effort by the researchers by focusing on gesture research instead of learning new tools & SDKs. The client's current system requires that a gesture recognizer is developed from the beginning for each new project, as the server only gives the application user data but does not interpret it in anyway. This is not efficient as it requires the client to do almost the same work for every project.
There is an existing gesture recognition system that the client uses. It's based on Microsoft Kinect hardware. Existing system uses a client / server architecture paradigm where the server provides gesture data to the client through socket connection. The project team has created a skeleton data manger, which connects to skeleton server and polls for up to date data all the time. We have also created hesture distributor which contains recognition engine which uses data gained from skeleton data manager. And we also create gesture recorder, which is a separate application used for recording new gestures.

In this project, we have created an interface which is connected to an existing server for Microsoft Kinect. We have created a recognition system that uses predefined gestures that can be used in any application We have also created an application which allows the user to record their own gestures which can then be used in different applications. This gesture recording application has a very GUI which makes it easy to use even for people with no programming experience.

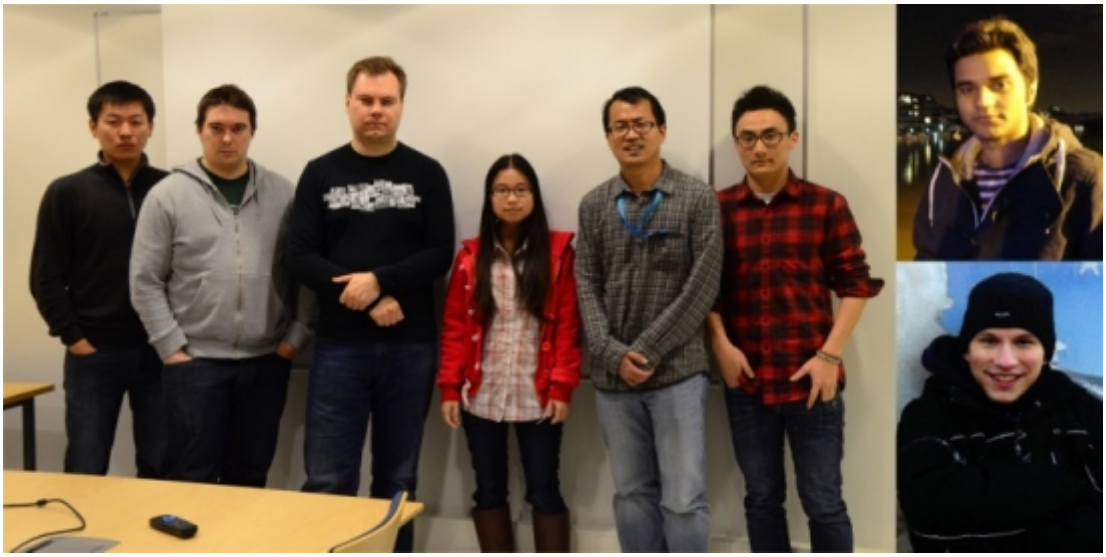
New gesture recorded with gesture recording software



UI for replaying and editing recorded gestures

Organization and management

Our group consisted of Eight members, four managers and four developers. The details of the group members are given below:
Santeri Saarinen, *Development Manager*
Mika Pesonen, *Survey Manager*
Muhammand Faisal, *Scrum Master*
Yameng Qin, *Team Communications Manager*
Maciej Kachniarz, *Developer*
Jiadong Liu, *Developer*
Zhao Hanning, *Developer*
Byambadorj Dulamsuren, *Developer*



Methods and tools

The development in this project was done with Microsoft Visual Studio 2010. We used C# as our programming language. For Kinect, we used Microsoft Kinect SDK v.1.5, and coding4fun toolkit. Socket programming was used to connect to the gesture recognition server and client provided by the client. For user interface design, the project team used Balsamiq software.

Project also used Redmine for tracking project progress via tickets and wiki page of the Redmine page was used to communicate other material. Hours were tracked using the Redmine. Redmine also contained a project wiki (https://redmine.sis.uta.fi/projects/gesture/wiki), which was used to save any important information about the project and partly also used for communication. The team also used Facebook group for daily communication between members. Skype and Gtalk were used for person to person communication and long-range meetings.

SVN source code repository was used and there was a benefit how SVN integrates to Redmine. Also the client used svn, so we could easily move the code between our and their repository when needed.

All documentation for the project was created with Google Docs, as this allowed the team members to edit texts simultaneously. For personal reports, other text editing tools, such as Microsoft Word were used.

Project phases and development model

Modified scrum model was used during project. Sprints were defined; backlog was created based on the requirements. Unfortunately was not possible to organize daily scrum meeting as developers were not able to work every day 8 hours. However, developers reported what they worked on, what were they going to do next and if they had any problems. Same questions were asked during the weekly meetings. This helped to understand if there was anything blocking the development.

Project sprints were three weeks long, except during the Christmas, as the members were not required to work during that time. Software was tested as it was created by the developers and each feature will be marked when it passed required tests.

Task	Start Date	End Date	Completed	Corresponding Person
Project Management	24.9.2012	15.3.2013	Yes	Project Managers
Preliminary Analysis	24.9.2012	5.10.2012	Yes	Project Managers
Survey	24.9.2012	15.12.2012	Yes	Mika Pesonen
Architecture Design	24.9.2012	4.11.2012	Yes	Santeri Saarinen
Implementation	15.10.2012	10.3.2013	Yes	Project Managers
UI Design	5.11.2012	16.12.2012	Yes	Project Managers

Table 2: Project Phases

During the project we had 3 reviews with the client, which were held in November, January and February. During these reviews we presented the current results of the project, and planned further requirements and goals for the group.

Experiences

Foreseen Risks

Risk	Analysis
Keeping Schedule	Keeping schedule is always a problem in student projects as there are other courses students are participating at the same time. Analysis/Countermeasure Workload was properly divided amongst the group members and any change in schedule was properly handled and adjusted.
Communication Problems	In university projects, mostly students are not working in the same space at the same time, and there might be situations where students need some code fix or other delivery from each other, and communication is important at this stage, and failing to do so could lead to significant delay. Analysis/Countermeasure Different communication channels were used to ensure that everyone is connected to each other and if anyone is unavailable, they informed before time to avoid any inconvenience.
Technology Problems	Technology was especially challenging in this project as new technologies were needed to be learned in short time and some gesture recognition algorithms were hard to understand and implement. Analysis/Countermeasure As most of the group members had no experience in the field of gesture recognition, and the tools which were being used, so it was made sure that adequate learning time in the project hours was included. The group members learned the new tools and technologies, and wherever they had any problem, they were able to resolve the issues with the help of experienced group members.
Architecture Design	As the delivered software had to support several gesture based technologies, there might have been problems if we designed the architecture wrong way, or if we designed the APIs so that adding new technology was not easy once the system was complete.

	Analysis/Countermeasure This risk was properly handled and was overcome by spending plenty of time on the architecture design and refining it so that the final architecture was without any problems.
Wrong Group Balance	In the project, we had a wrong balance of managers and developers. We had 4 managers and 4 developers. Also, it was in the beginning known that one member will not be able to work until the end of this project. Analysis/Countermeasure To cater this problem, managers also actively participated in different development tasks, so sharing the workload of the developers.
Team Member Quitting	Luckily, none of our members quit, but some were away for longer periods of time, and one had to leave back to his home country in the middle of the project. Analysis/Countermeasure To prevent this becoming a problem, we reorganized the tasks to members who were available during these periods of time. And we also mailed a Kinect abroad to enable one project member to continue working from home.

Table 1: Foreseen Risks

Our experiences

Arranging meeting times which were good for the whole group was difficult as people had busy schedules. At one point we gave up on finding good times that fit everyone and agreed that if people are not able to join, they will report their progress by email. We also used Facebook to keep everyone up to date.

Still we did not have enough communication within the group. We only had discussions maybe once or twice a week, and this was not enough to follow everyone's progress, especially if someone was not able to join the conversation.

Our group was not correctly balanced, so the workload between members could not be divided correctly. Because of this, the managers also took some tasks of the developers.

Also, some of our members were away for longer periods of time, which caused them to be unable to help with the project. We tried to divide tasks again if this happened, but because of this, the tasks were not evenly spread to all members of the team, and some worked more than others.

Luckily, we also had some good experiences. Our group was diverse with lot of different skills, which meant that even with new technologies and lot of need for studying, people still managed to complete tasks well.

Also our subject was very interesting and challenging, and all members learned something new during the project.

Statistics

Team size	Dev. model	Start date	End data	Days	Hours
2+5+1	Modified scrum	24.09.12	18.03.13	176	

Table 1: General project information.

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	300	20	34	234	46	48,5	0	221,5	22,5	926,5
%	32,38%	2,15%	3,67%	25,26%	4,96%	5,23%	0,00%	23,91 %	2,43%	100%

Table 2: Group effort by activity.

Number of requirements	Use-cases	UI screens	Database diagrams
57	0	3	0

Table 3: Requirements and high-level design outcomes.

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
5	3	3	0	0	0

Table 4: Design outcomes.

Document	Pages	Versions
Preliminary analysis	9	3
Project Plan	17	3
Requirements specification	1	10
Design plan	5	3
Test report	6	2
Final report	8	5
Project's story	8	2
Weekly reports	21	21
Survey	21	4

Table 5: Documents.

Language	C#
LOC	3306
SLOC	4577
Reused code	300
Reused and modified	300
Classes	50
Code revisions	49

Table 6: Codelines.

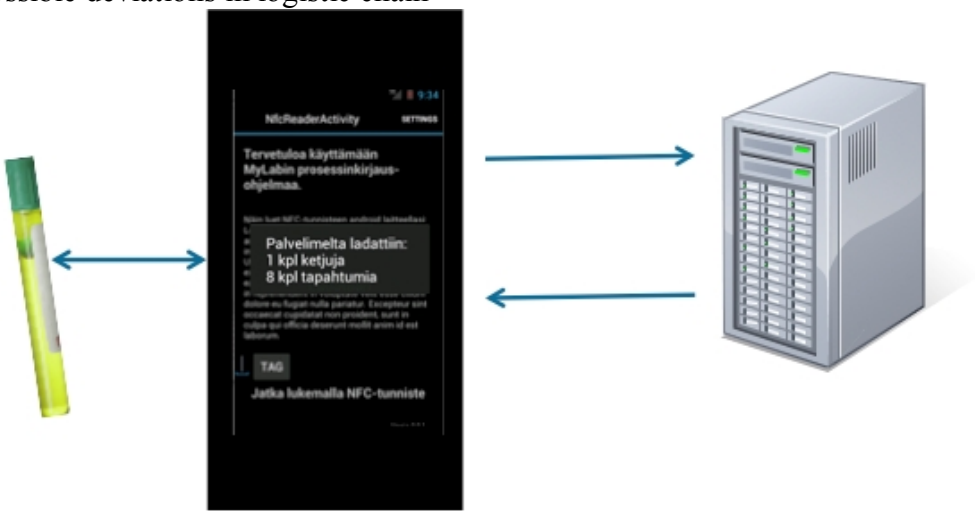
Logistiikkaketjun kirjaus

Overview

Our project's goal was to implement a software to the Android mobile phone. The software is a prototype, which client can introduce a trade shows and customer events.

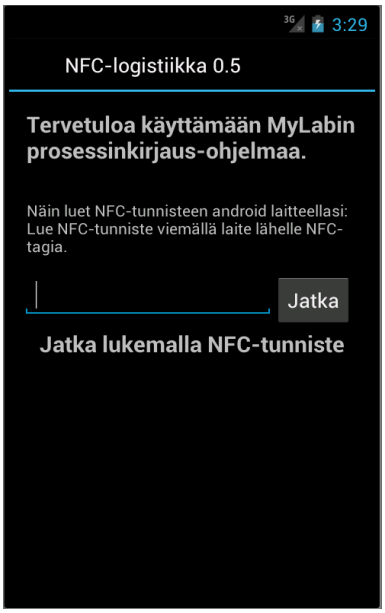
Software's main functionalities are:

- Reading NFC-tag
- Registering new logistics transactions
- Seeking logistics transactions
- Possible deviations in logistic chain

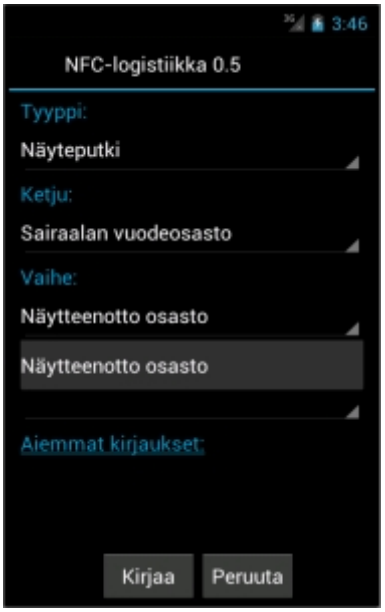


-

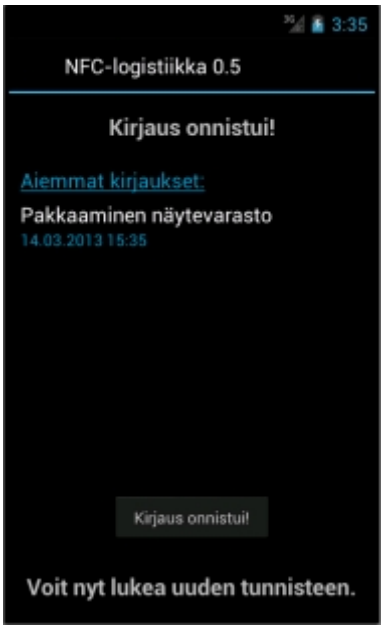
Picture 1. Mobile phone read NFC tag and is connected to the server.



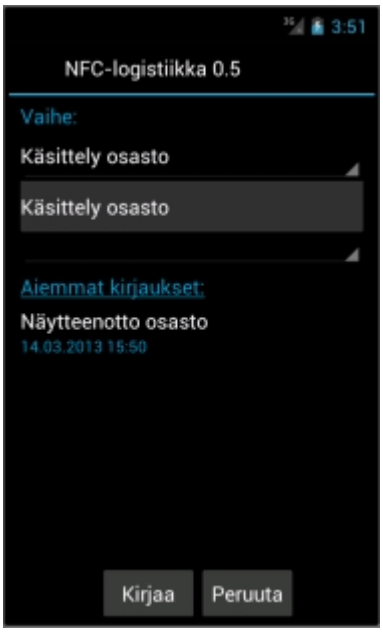
Picture 2. Welcome page. User can read NFC –tag or input it by keyboard or using bar code.



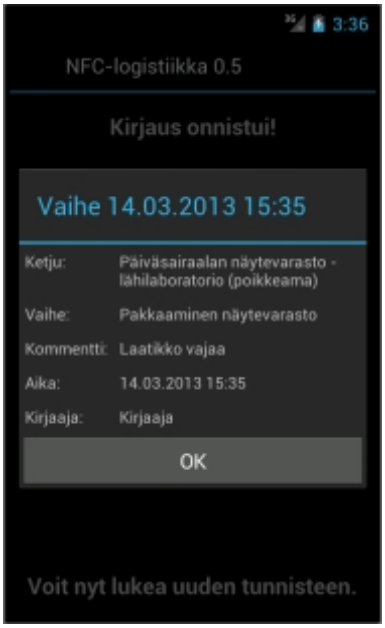
Picture 3. User select information: logistics type, chain, phase and comment.



Picture 4. Feedback page. New phase has been successfully saved.



Picture 5. Registration of the new phase.



Picture 6. Phase information dialog.

Organization and management

Our team had 3 Project Managers and 4 Developers.

Project Managers and their responsibilities:

- Tuomas Granlund (Technical Project Manager)
    - Development environment
    - Version control

    - Technical guide
    - Planning document
    - Architecture plan
    - Installation plan
    - Code auditing
    - Training
- Elina Koivulampi
    - Course Schedule
    - Hourly accounting
    - Facebook group
    - Deployment plan
    - Testing plan
    - Testing
    - Project Manager meetings
- Ari Varpenius
    - Project plan
    - Project rules
    - Weekly meetings
    - Specification document
    - Project presentations

Developers and their responsibilities:

- Juha Kaura
    - Application server installation and configuration
    - XML message structure
    - Server application implementation
- Heini Pylkkönen
    - User Interface plan
    - User Interface implementation
- Kaj Torrkulla
    - Database plan
    - Database implementation
- Lauri Vene
    - Android application implementation

Course supervisor:

- Pekka Mäkiaho

Client and our contact person:

- Mylab, Lari Pelkonen

We had weekly meetings where Project Manager describe status of the project and developers describe what they have done after previous meeting and do they have any kind of problems or questions. Then Project Manager and developers made plans for next week works.

Project Managers had meetings every two or four weeks and in those meetings Project Managers planned big picture of the project.

We also had Trello as a project management tool and we had good experiences about that system.

We met our client 4 times and Tuomas discussed regularly with the client's contact person.

Methods and tools

Android and application server

Eclipse SDK 4.2.1 (Juno)

SDK Platform Android 4.1.2-update1, API level 16,

Tomcat 7.0.26.

Java: 1.6.0_24.

Database

MySQL Community Server 5.5.28

(Emacs 24.2)

Version control

Git , version 1.7.11

Task and time management

Trello (http://www.trello.com)

Documentation

Project course's wiki: https://projectwiki.cs.uta.fi/wiki/Logistiikkaketjun_kirjaus

Google Drive

Testing tools

No tools

Project phases and development model

Our project had iterative model.

1. Specification
2. Design
3. Implementation

   I Release 11.11.2012

   II Release 2.12.2012

   III Release 23.12.2012
4. Testing
5. Production starts 7.2.2012

First we specified all client's requirements and designed the whole software. Then we split implementation to three releases. In the testing phase we tested whole software, client made acceptance testing and after that the software was ready for the production phase. We chose this kind of project model because we think that by using this model we can exactly be sure that production phase starts in due time.

There is a list of project's tasks:

Task	Start	End
Project presentation	19.9.2012	19.9.2012
Project's kickoff	21.9.2012	21.9.2012
Meeting the client	27.9.2012	27.9.2012
Specification workshop	27.9.2012	27.9.2012
Preliminary analysis meeting	28.9.2012	28.9.2012
Project plan implementation	24.9.2012	8.10.2012
Specification document's implementation	27.9.2012	28.10.2012
Planning phase starts	8.10.2012	8.10.2012
Project plan review	15.10.2012	15.10.2012
Implementation starts	15.10.2012	15.10.2012
I release	11.11.2012	12.11.2012
II release	2.12.2012	2.12.2012
III release	21.12.2012	21.12.2012
Deployment plan	1.11.2012	7.1.2013
Finishing phase	6.1.2013	18.1.2013
Testing phase	19.1.2013	27.1.2013
Installation phase	23.1.2013	27.1.2013
Customer training	28.1.2013	28.1.2013
Acceptance testing	28.1.2013	1.2.2013
Finishing phase	2.2.2013	4.2.2013
Deployment phase	5.2.2013	6.2.2013
Production phase starts	7.2.2013	7.2.2013
Finishing the documentation	11.2.2013	20.3.2013

I release 11.11.2012

Tasks
Database planning and implementation
Application server installation
Model of application / Android

Logistics Application class implementation / Android
Request class implementation / Android
HTTPUtils class implementation / Android
Response class implementation / Android
ResponseFactory class implementation / Android
Welcome page implementation / Android
The initial setup query / Application server
War packaging / Application server

II release 2.12.2012

Tasks
Logistics Application class implementation continue / Android
NFC Reader Activity class implementation / Android
Request class implementation continue / Android
HTTPUtils class implementation continue / Android
NFCUtils class implementation / Android
Response class implementation / Android
ResponseFactory class implementation continue / Android
Event logging page implementation / Android
Tag information from Android and analyse and information to Android / Application server

III release 21.12.2012

Tasks
NFC Reader Activity class implementation continue / Android
FormSenderActivity class implementation / Android
EventDialogFragment class implementation / Android
Feedback page implementation / Android
Phase information dialog implementation / Android
Event logging input and output / Application server

Experiences

Our project didn't met any foreseen risks.

We should have add extra phase to verify that all the functionalities works correctly before the actual test phase.

Bad experience was that we had too many ways to communicate (IRC, Facebook, email, Google Drive, Trello, weekly meetings).

Statistics

Team size	Dev. model	Start date	End data	Days	Hours
3+4	Iterative	16.9.2012	22.3.2013	187	1084

Table 1: General project information.

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	415	17	63	325	16	23	21	103	101	1084
%	38,3	1,57	5,81	30,0	1,48	2,12	1,94	9,51	9,28	100%
Total										**1084**

Table 2: Group effort by activity.

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Week 37	12	0	2	0	0	0	0	2	3	19
Week 38	31	0	0	0	0	4	0	9	11	55

Week 39	27	9	0	0	1	2	0	11	7	57
Week 40	23	0	0	0	3	0	0	10	27	63
Week 41	22	4	4	0	0	2	0	9	3	44
Week 42	25	4	14	4	0	3	1	3	5	59
Week 43	24	0	12	11	0	1	1	6	2.5	57.5
Week 44	11	0	1	33	0	1	1	12	3	62
Week 45	17	0	2	31	0	2	1	8	2	63
Week 46	39	0	1	7	2	0	2	3	4	58
Week 47	14	0	12	33	0	1	0	6	4.5	70.5
Week 48	27.5	0	4	74	0	1	0	19	2	127.5
Week 49	17	0	2	16	0	0	1	0	2	38
Week 50	13	0	0	20	0	1	0	1	1	36
Week 51	14	0	2	30	0	0	0	4	2	52
Week 52	1	0	0	2	0	0	0	0	1	4
Week 1	11	0	0	0	0	0	0	0	0	11
Week 2	12	0	0	24	0	0	0	0	2	38
Week 3	21	0	0	31	3	0	4	0	2	61
Week 4	22.5	0	7	9	7	3	5	0	1	54.5
Week 5	8.5	0	0	0	0	2	3	0	1	14.5
Week 6	6.5	0	0	0	0	0	0	0	0	6.5

Week 7	2	0	0	0	0	0	2	0	0	4
Week 8	0	0	0	0	0	0	0	0	0	0
Week 9	4	0	0	0	0	0	0	0	0	4
Week 10	2	0	0	0	0	0	0	0	8	10
Week 11	8	0	0	0	0	0	0	0	7	15
Total	415	17	63	325	16	23	21	103	101	1084

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
3	8	3	9	0	0

Table 3: Requirements and high-level design outcomes.

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
14	3	3	5	1	3

Table 4: Design outcomes.

Document	Pages	Versions
Preliminary analysis	5	1
Project Plan	20	9
Usability analysis	-	
Requirements specification	8	2
Design plan	-	
User interface document	1	1
Test plan	16	5
Test report	2	1

Usability test report	-	
Final report		1
Project's story	11	1
Weekly reports	23	1

Table 5: Documents.

Language	JAVA, XML, SQL
LOC	16 186
SLOC	6 319
Reused code	-
Reused and modified	-
Classes	67
Functions	502
Code revisions	306

Table 6: Codelines.

Majava 4

Kuva 1: Majava-sivuston etusivu.

Yleiskuva projektista

Majava-kilpailu on koululaisille suunnattu tietotekniikka-aiheinen kilpailu, joka järjestettiin Liettuassa ensimmäisen kerran vuonna 2004. Viime vuonna kilpailu järjestettiin kymmenessä maassa eri puolilla Eurooppaa. Kilpailua alettiin järjestää, koska ympäri Eurooppaa alettiin huolestua tietotekniikan aseman heikkenemisestä koulujen opetussuunnitelmissa. Tämän seurauksena koululaisten tietotekninen osaaminen heikkeni. Majava-kilpailun perimmäisenä tarkoituksena on tutustuttaa peruskoululaisia ja lukiolaisia tietotekniikkaan hauskalla mutta opettavaisella tavalla sekä lisätä opiskelijoiden kiinnostusta tietotekniikkaa kohtaan. Tavoitteena on myös tukea opettajia tietotekniikan opetuksessa.

Majava-kilpailussa kysymykset on jaettu ikäryhmittäin neljään ryhmään. Kysymyksissä on pyritty korostamaan yleisiä ongelmanratkaisutaitoja. Kysymykset voivat koskea esim. tiedon esittämistä, loogisia arvoituksia ja pelejä sekä tietotekniikkaa ja yhteiskuntaa. Nykyisellään kilpailu on järjestetty www-pohjaisen sovelluksen avulla, joka on otettu Suomessa käyttöön vuonna 2010. Majava-kilpailu löytyy osoitteesta www.majava-kilpailu.fi.

Projektiin liittyen vastasimme vuoden 2013 Majava-kilpailun valmistelusta ja järjestämisestä Suomessa. Valmisteluun sisältyi muun muassa kilpailukysymysten suomentaminen, niiden syöttäminen kilpailujärjestelmään sekä järjestelmän toimivuuden testaus.

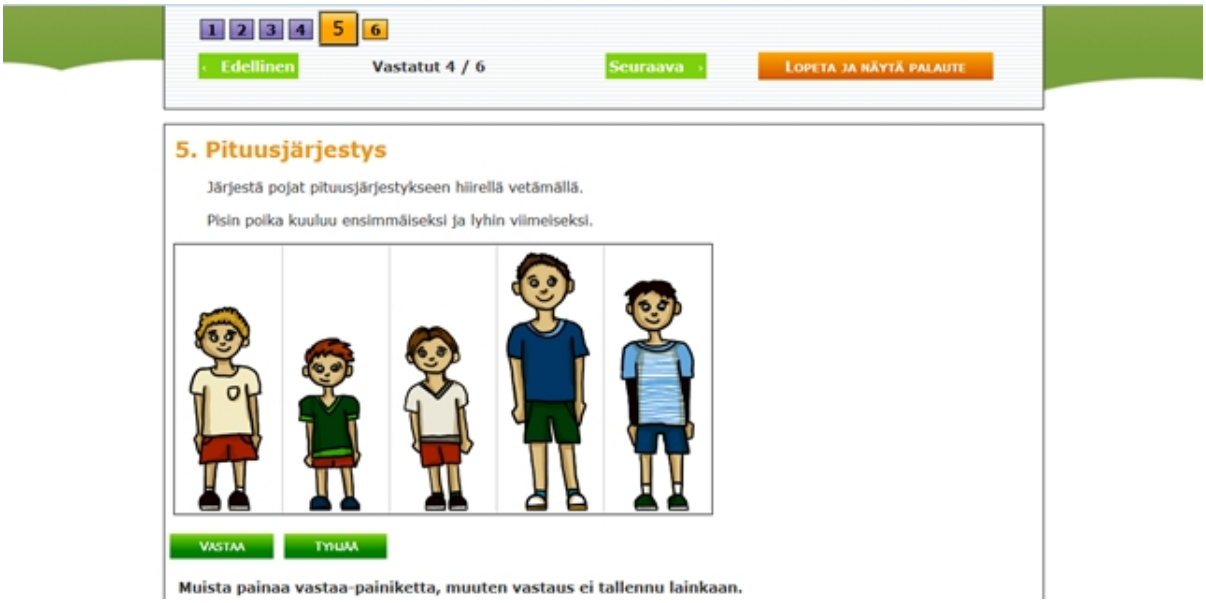
Vasta kilpailuvaiheen jälkeen pääsimme varsinaisesti työstämään järjestelmän pohjana olevaa Majava-sovellusta. Saimme slovenialaisilta yhteistyökumppaneilta

uuden koodipohjan, jonka lokalisoimme ja jatkokehittelimme edelleen lisäten uutta toiminnallisuutta ja korjaten vanhoja ohjelmistovirheitä. Päivitimme myös sivuston ulkoasua ja uusimme kuvia.

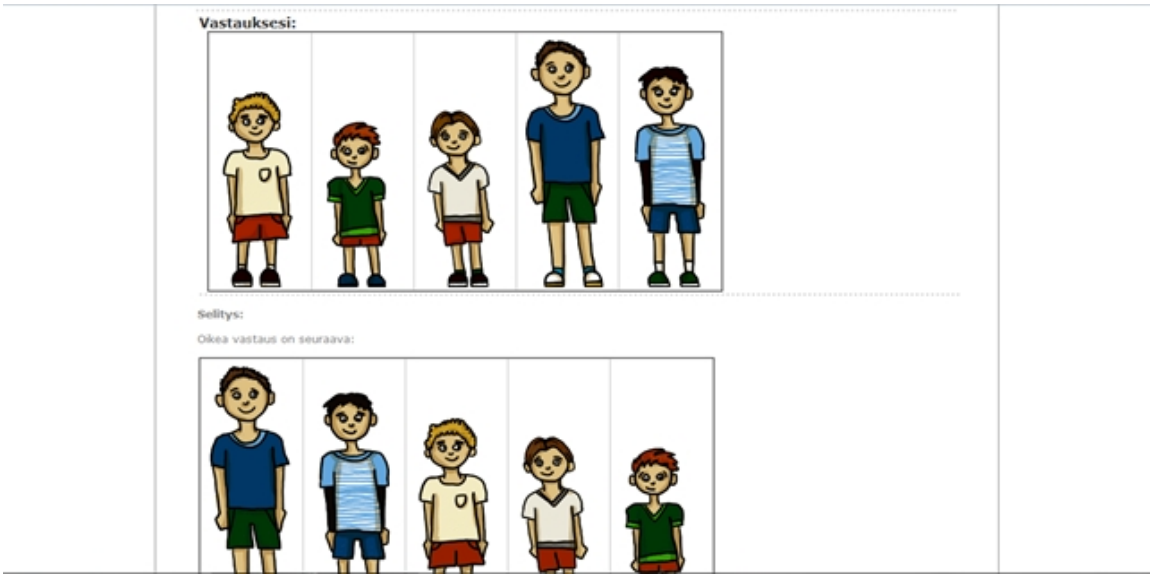
Tärkeimpänä yksittäisenä uudistuksena koodasimme sovellukseen tuen vuorovaikutteisille tehtäville. Aiemmin kilpailukysymykset olivat sen tyyppisiä, että annetuista vastausvaihtoehdoista piti valita yksi oikea vastaus. Sen sijaan uusissa vuorovaikutteisissa tehtävissä kilpailija voi esimerkiksi hiirellä siirrellä tehtävään liittyviä kuvia oikeille paikoilleen ja muodostaa näin vastauksensa. Toteuttamamme rajapinta mahdollistaa hyvin erityyppisten tehtävien laatimisen sekä annetun vastauksen tulkinnan.

Käyttöliittymä

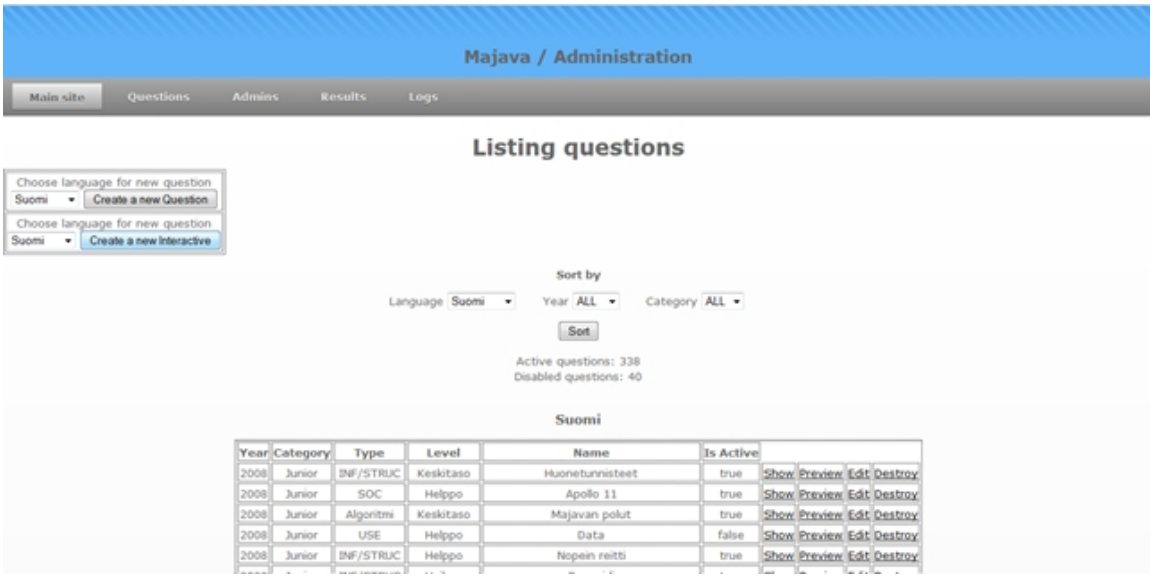
Kuva 2: Vuodenvalintasivun ulkoasu on yhtenäistetty etusivun ulkoasun kanssa.



Kuva 3: Esimerkki vuorovaikutteisesta tehtävästä: Pelkkien vastausvaihtoehtojen sijasta kuvan hahmojen paikkaa voi vaihtaa hiirellä vetämällä, ja tieto tallentuu tietokantaan yhteensopivassa muodossa.



Kuva 4: Annettu vastaus näytetään yhteenvetosivulla sellaisena, kuin käyttäjä siihen vastasi. Sisäisesti ohjelma osaa alustaa tehtävän lopullisen asennon, koska tehtävä alustetaan käyttäjän antaman vastauksen serialisoidun merkkijonon perusteella.



Kuva 5: Vuorovaikutteinen tehtävä lisätään hallintapuolen “Create a new interactive” -painikkeesta. Myös vanhaa rajapintaa käyttävät tehtävät toimivat järjestelmässä ilman muutoksia.

Kuva 6: Vuorovaikutteisen tehtävän koodit kirjoitetaan niille varattuihin kenttiin. Kun kuuntelijafunktiot määritellään erikseen, tehtävän esittäminen vain luku -tilassa on mahdollista. Tätä tarvitaan esimerkiksi tehtävien yhteenvetosivulla.

Ryhmä ja projektinjohto

Ryhmän jäsenet rooleineen

Päälliköt:

- Jouni Kähkönen: Johto, laadunvarmistus, palvelin- ja koodipäivitykset
- Antti Kiiskinen: Johto, business intelligence, palvelinpäivitykset

Ryhmäläiset:

- Toni Helenius: Vuorovaikutteistuen toteutus, koodaus
- Elias Roihuvuo: Dokumentointi, esimerkkitehtävät, koodaus
- Tuomas Räsänen: Layout-vastaava, esimerkkitehtävät, koodaus
- Antti Reunamo: Lokalisointivastaava, esimerkkitehtävät, koodaus

Asiakas:

- Timo Poranen

Projektin organisointi

Projektin yhtenä tärkeimmistä tavoitteista projektityöntekijän näkökulmasta oli oppia työskentelemään monenlaisten haasteiden parissa. Vastoin yleisiä oletuksia haasteet eivät välttämättä ole ainoastaan teknisluonteisia, vaan projektin etenemisessä saattaa välillä aiheuttaa ongelmia myös ihmisten välinen viestintä. Tämän johdosta projektin johtamisessa olennaista oli se, että pystyimme luomaan riittävästi ryhmän keskuudessa tehokasta työskentelyhenkeä ja keskinäistä luottamusta ryhmän jäsenten keskuudessa tehdyn työn tärkeydestä. Projektipäällikkönä koimme tärkeäksi nähdä minkälaista tietoa kannattaa jakaa projektilaisten keskuudessa aiheuttamatta sekaannusta ryhmäläisten keskuudessa toteutettavien toimintojen suhteen.

Myös tehtävien oikea jakaminen projektilaisten kesken oli olennaisessa osassa projektin onnistumisen kannalta. Oli pystyttävä mittaamaan projektityöntekijöiden valmiustasoa tehtävien suorittamiseksi tietyllä aikataululla, sekä organisoitava tehtäviä tämän mukaisesti. Lisäksi ongelmatilanteissa projektityöntekijöitä oli pystyttävä tarvittaessa motivoimaan ja avustamaan tehtävien loppuun saattamiseksi. Tähän käytimme muun muassa työpaja-lähestymistapaa projektimme aikana.

Tiedolla johtaminen on toimintatapa, jossa erilaisissa tietolähteissä olevaa informaatiota keräämällä, jalostamalla ja hyödyntämällä luodaan yhteinen ymmärrys siitä, mitä asioita liiketoiminnassa tulee johtaa sekä luodaan kyvykkyyttä muuttaa tämä ymmärrys paremmaksi tulokseksi ja kilpailukyvyksi. Tämä ajatusmalli voidaan viedä jopa projektiryhmätasolle.

Ohjelmistoprojekteja on erilaisia ja niiden kanssa käytettävien erilaisten työkalujen kanssa tulee pystyä työskentelemään. Projektin pitkäaikaisluonteisuus vaatii pitkäjänteisyyttä ajattelun ja motivoitumisen suhteen koko projektin aikana. Tämän tavoittamiseksi näimme kurssin antaneen hyviä toimintamalleja.

Metodit ja työkalut

Projektissa edettiin iteratiivisesti kevyemmän Scrum-mallin mukaan, jossa ryhmä tapasi viikoittain kasvotusten. Muuten yhteyttä pidettiin sähköpostitse ja IRC:n välityksellä. Dokumentit projektin kulusta ja julkaisuista kirjoitettiin Google Docsilla ja viimeisteltiin Microsoft Officella. Ohjelmakoodia ylläpidettiin Subversion-versionhallintajärjestelmässä ja projektin etenemistä seurattiin Redminellä. Majava-kilpailua ajetaan Ruby on Rails -sovelluskehyksessä, joka oli asennettuna jokaisen ryhmäläisen kotikoneen Ubuntu-näennäiskoneeseen.

Projektin vaiheet ja kehitystyö

Päiväys	**Aihe**	**Toteutunut päiväys**
17.9.–1.10.2012	Sprintti 1	17.9.–1.10.2012
1–15.10.2012	Sprintti 2	1–15.10.2012
5.10.2012	Ympäristööntutustumiskatselmointi	5.10.2012
28.9.2012	Esitutkimus	1.10.2012
15–29.10.2012	Sprintti 3	15–29.10.2012
29.10.–12.11.2012	Sprintti 4	29.10.–12.11.2012
1.11–7.11.2012	Henkilökohtainen raportti I	1.11–7.11.2012
12–16.11.2012	Majava-kilpailu	12–16.11.2012
12–26.11.2012	Sprintti 5	12–26.11.2012
12.10.2012	Projektisuunnitelmapalaveri	23.11.2012
23.11.2012	Asennuspalaveri	23.11.2012
26.11.–10.12.2012	Sprintti 6	26.11.–10.12.2012
28.11.2012	Väliesitys	28.11.2012
30.11.2012	Katselmointi 1	30.11.2012
10–24.12.2012	Sprintti 7	10-24.12.2012
2.1.2013–16.1.2013	Henkilökohtainen raportti II	2.1.2013–16.1.2013
16.12.2012	Katselmointi 2	28.1.2013
24.12.2012–7.1.2013	Sprintti 8	24.12.2012–7.1.2013
7–21.1.2013	Sprintti 9	7–21.1.2013
21.1.–4.2.2013	Sprintti 10	21.1.–4.2.2013

31.1.2013	Katselmointi 3	22.3.2013
4–18.2.2013	Sprintti 11	4–18.2.2013
18.2.–4.3.2013	Sprintti 12	18.2.–4.3.2013
4–18.3.2013	Sprintti 13	4–18.3.2013
15.3.2013	Loppupalaveri	22.3.2013
Projektin jälkeen	Henkilökohtainen raportti III	Loppupalaverin jälkeen

Projektin toteutuneet päiväykset ovat hyväksytysti toteutuneet hieman myöhempänä ajankohtana, johtuen muun muassa projektin alussa pidetystä kilpailusta, joka asiakkaan ja ohjaajan näkökulma huomioon ottaen pidettiin tärkeimpänä prioriteettina projektin alussa. Lisäksi yhteistyö ulkomaalaisen yhteistyökumppanin kanssa aiheutti sen, että kehitystyötä ei voitu aloittaa ennen päivitetymmän saantia heiltä.

Kehitystyö

Kilpailun järjestämiseen liittyvä toiminta

Projektin alussa järjestimme Majava-tietotekniikkakilpailun onnistuneesti yli 2 000 kilpailijalle. Tähän sisältyi kilpailuun valmistautuminen, itse kilpailun etenemisen seuraaminen sekä kilpailun jälkeen muun muassa osallistujille tarkoitettujen kunniakirjojen päivittäminen.

Kilpailuun valmistautumiseen liittyi kilpailukysymysten suomentaminen sekä syöttö järjestelmään ja kopiointi eri ikäryhmille. Lisäksi joihinkin tehtäviin liittyviä kuvia täytyi muokata, jotta tehtävät soveltuisivat suomalaiseen ympäristöön. Juuri ennen kilpailua testasimme vielä kilpailun toimintaa uusien kysymysten kanssa.

Kilpailun aikana valvoimme järjestelmän toimintaa. Olimme valmiudessa mikäli ongelmia syntyisi, mutta kilpailu sujui kuitenkin ongelmitta. Laadimme myöhemmin tehtävistämonisteen, joka julkaistiin laitoksen raporttina.

Sovelluksen jatkokehitys

Kilpailun jälkeen jatkokehitimme olemassa olevaa Majava-sovellusta lisäten siihen toimintoja ja korjaten vikoja. Saimme koodin slovenialaiselta yhteistyökumppanilta, ja se vaatikin aluksi lokalisoinnin parantamista. Siirsimme kaikki tekstit itse koodin sisältä erilliseen lokaalitiedostoon, joten jatkossa sovellus voidaan helpommin muuttaa eri kielillä käytettäväksi. Päivitimme myös projektin aikana sekä testi- että tuotantopalvelimien Rails-ohjelmistokehykset uusiin versioihin.

Korjasimme lukuisia ohjelmistovikoja ja teimme runsaasti pieniä muutoksia sovellukseen. Yhtenäistimme sivuston graafista ulkoasua ja lisäsimme statistiikan kävijätietojen seuraamiseen. Paransimme myös käytettävyyttä etenkin ylläpitopuolen näkymien osalta sekä poistimme tarpeettomia ylimääräisiä toimintoja. Testasimme sivuston toiminnallisuutta useilla eri selaimilla.

Sovellukseen liittyvän kehitystyön lisäksi laadimme projektisuunnitelman sekä muita analyysejä ja raportteja. Myös kehitysympäristöjen asennus kehittäjien omille koneille oli jonkin verran aikaa vaatinut prosessi.

Vuorovaikutteisten tehtävien tuki

Tärkeimpänä yksittäisenä toimintona kehitimme projektin aikana sovellukseen tuen vuorovaikutteisille tehtäville sekä niiden lisäämiselle ja muokkaamiselle. Toteutimme myös joitakin vuorovaikutteisia esimerkkitehtäviä. Ratkaisussamme vuorovaikutteiset tehtävät laaditaan JavaScript-ohjelmointikielellä jQuery-kirjastoa apuna käyttäen. Tehtävien laatijalla on vapaat kädet itse toteutuksen sekä vastauksen tulkinnan osalta; jokainen tehtävä voi olla koodiltaan erilainen. Tehtävien luonnissa on kuitenkin täysin erilliset rajapinnat tehtävän rakenteen luonnille, ulkoasulle sekä käyttöliittymätapahtumille.

Dokumentoimme vuorovaikutteisten tehtävien lisäämistoiminnon lisäksi myös joitakin käyttötapauskuvauksia. Tuotimme raportin jokaisesta pitämästämme palaverista ja kirjasimme ylös tietoja projektinhallintaohjelmiston avulla.

Analyysiä kehitysympäristöstä ja Majava-projektista

Rails on jatkuvasti kehittyvä ja modulaarinen ohjelmistoympäristö. Majavan koodin laadun johdosta sen ylläpidosta tulee oma haasteensa. Järjestelmä saatiin päivitettyä Rails 3.2 -versioon, joten haavoittuvuudet saatiin korjattua ainakin joksikin aikaa. Majavan aikaisemman version, Rails 3.0:n tuki loppui 2013 tammikuussa. Kilpailupalvelin toimii tällä hetkellä Ruby 1.8 -versiolla ja viimeinen Rails-versio, joka tätä tukee, on 3.2.

Majava-kilpailun tapaisilla järjestelmillä olisi hyvä olla jonkinlaista varmuutta toiminnasta. Käsin testaus antaa hyvin vähän takuuta toimivuudesta uusia ominaisuuksia luotaessa varsinkin rasituksen alla. Ruby on Railsin tapaiset sovelluskehykset on luotu testattavien järjestelmien toteuttamiseen ja sisältävät valmiiksi tähän tarvittavat työkalut.

Vuorovaikutteisten tehtävien toimivuutta ei tullut simuloidussa kilpailutilanteessa testattua. Vaikka olemme nähneet vuorovaikutteisten tehtävien toteutuksen toimivan moitteettomasti kehityspalvelimiemme, testipalvelimen ja kilpailupalvelimen harjoitusosiossa, niitä ei ole testattu kuitenkaan kilpailutilanteessa toimiviksi.

Kokemuksia projektista

Opin projektin aikana projektinjohtoa, sopivan haasteellisten tehtävien asettamista sopiville tekijöille sekä ohjelmistoprojektiryhmän yleistä organisointia. Koin tärkeäksi projektin aikana tuen antamisen ryhmäläisille silloin, kun jonkin toiminnallisuuden toteuttamisessa oli ongelmia. Lisäksi ohjelmointityöpajat olivat omiaan lisäämään ryhmän motivaatiota kehittää sovellusta eteenpäin.

Ennen olin tottunut käyttämään erinäisiä ohjelmistoprojekteissa tarvittavia työkaluja, mutta tämän kurssin myötä opin myös opettamaan toisia työkalujen käytössä sekä ohjeistamaan mistä saa lisätietoja tarvittavista komponenteista, jotta vaatimusten mukaiset toiminnot saadaan toteutettua.

Projektin johtaminen onnistui pääpiirteissään hyvin. Toisaalta tilanteessa, jossa projektia johtaa kaksi projektipäällikköä, olisi päälliköiden säännöllisempi vetovastuuvuorottelu voinut osoittautua hyödylliseksi.

Ryhmän työskentely oli pääosin kiitettävää. Ryhmäläisillä oli riittävä motivaatio aina kun sopivia tehtäviä saatiin asetettua sopiville henkilöille. Projektipäällikön näkökulmasta oli mielekästä seurata ryhmäläisten halukkuutta oppia uusia asioita sekä nähdä ryhmäläisten omaksuneen tärkeitä ryhmätyöskentelytaitoja sekä ohjelmointitekniikoita, joista on hyötyä ohjelmistoalalla työskentelyssä.

– *Jouni*

Projektin yhtenä tärkeimmistä tavoitteista projektityöntekijän näkökulmasta oli mielestäni oppia työskentelemään monenlaisten haasteiden parissa. Vastoin yleisiä oletuksia haasteet eivät välttämättä olleet pelkästään teknisluonteisia, vaan projektin etenemisessä saattoi välillä aiheuttaa myös ongelmia erilaiset ihmiskemiat. Tämän johdosta projektin johtamisessa olennaista oli mielestäni pystyä luomaan ryhmän keskuudessa tehokasta työskentelyhenkeä ja valaa keskinäistä luottamusta ryhmän jäsenten keskuudessa tehdyn työn tärkeydestä. Tämä osoittautui ajoittain haasteeksi ja näin jälkikäteen ajateltuna tiettyinä hetkinä olisi voinut projektinvetäjänä käyttää myös erilaisia hengen nostattamistaktiikoita. Toisaalta saimme projektin aikana meille asetetut tavoitteet täytetyiksi ja projektistamme yksikään jäsen ei jättänyt kurssia kesken. Lisäksi asiakas esitti tyytyväisyytensä projektin läpiviennin suhteen. Nämä seikat näen yhtenä selvänä merkkinä onnistumisestamme projektin kokonaiskuvaa ajatellen. Projektipäällikön näkökulmasta ryhmäläiset suoriutuivat mielestäni erittäin hyvin ja osasivat asennoitua oikealla mentaliteetilla työskenneltäessä pitkäkestoisempien projektien parissa.

Ohjelmistoprojekteja on erilaisia ja niiden kanssa käytettävien erilaisten työkalujen kanssa tulee pystyä työskentelemään. Projektin pitkäaikaisluonteisuus vaatii pitkäjänteistä ajattelua ja motivoitumista projektin suhteen. Tämän näen kurssin yhtenä tärkeimmistä opetuksista.

– *Antti K*

Projekti oli mielenkiintoinen vaikka lähtikin hitaasti alkuun. Pitkäaikaisempi projekti on harvinaisempi yliopistossa, joten varmasti hyödyllinen.

– *Toni*

Projekti oli hyvä ja hyödyllinen. Tärkeiden perustyökalujen sekä esimerkiksi jQueryn opetteleminen tuntui mielekkäältä. Sovelluksen kehittäminen ryhmätyöskentelynä oli uutta ja opettavaista.

– *Antti R*

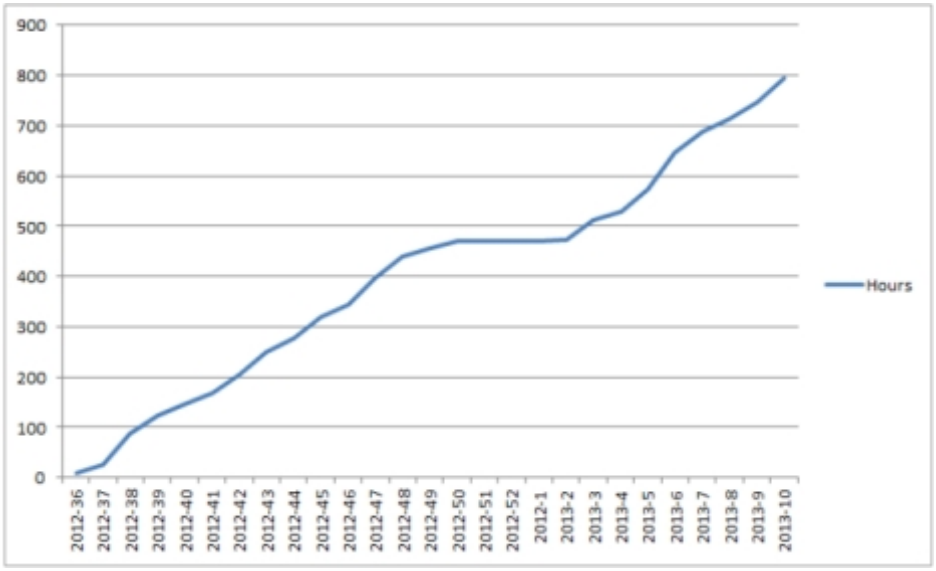
Projektin aikana sain jonkinlaisen kuvan siitä millaista on olla mukana ohjelmistoprojektissa. Projektin kuluessa ryhmätyötaitoni ja ehkä myös ohjelmointitaitoni kehittyivät. Lisäksi erilaiset työkalut tulivat tutuiksi. Arvelisin tämän kurssin olleen yksi hyödyllisimmistä joita olen yliopistossa suorittanut.

– *Tuomas*

Projektin aikana opin käyttämään tärkeitä projektinhallinnan työkaluja, kuten SVN. Projekti oli oivallista harjoitusta ryhmätyöskentelyn kannalta ja lisäksi oli innostavaa kehittää ohjelmistoa, joka on todellisessa käytössä.

– *Elias*

Tilastoja

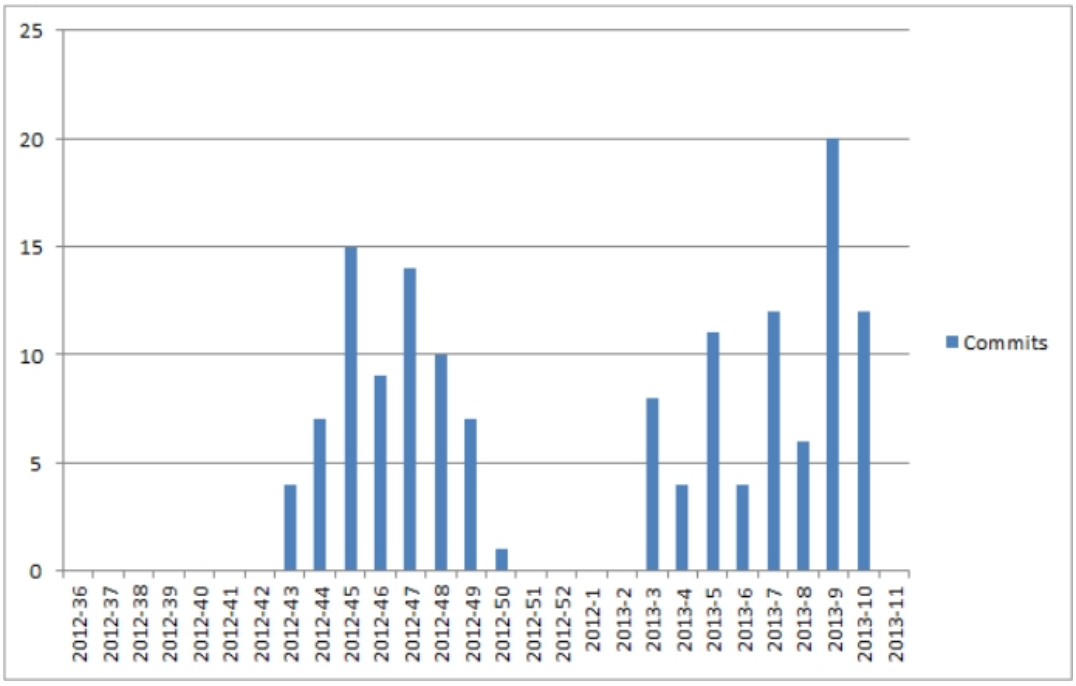


Kaavio 1: Tuntimäärät projektin aikana

Kaavio 1 havainnollistaa projektin eri vaiheet ja sen etenemisen kurssin aikana. Tuntimäärissä havainnollistuu projektin aloitus ja työkaluihin oppimiseen kulunut aikana viikoilla 36–42. Tämän jälkeen nähdään projektiin käytettyjen tuntimäärien kasvavan, kun projektilaiset alkoivat tehdä pieniä päivityksiä projektiin. Viikolla 46 tapahtunut Majava-kilpailu näkyy myös tuntitilastoissa lievänä tuntimäärien laskuna.

Projektisuunnitelmassa näkyy joululoma, jolloin lähinnä projektipäälliköt käyttivät muutamia tunteja projektin suunnittelemiseen ja ryhmäläisten väliseen

yhteydenpitoon. Joululoman jälkeen projektin kehitysvaihe käynnistyi viikolla 4. Tällöin myös kertyi projektin kannalta suhteessa eniten tunteja projektissa. Projektilaiset pääsivät keskittymään paremmin järjestelmän kehittämiseen ja päivittämiseen. Kehitystyötä tapahtui projektin loppuun saakka.



Kaavio 2: SVN-tallennukset projektin aikana

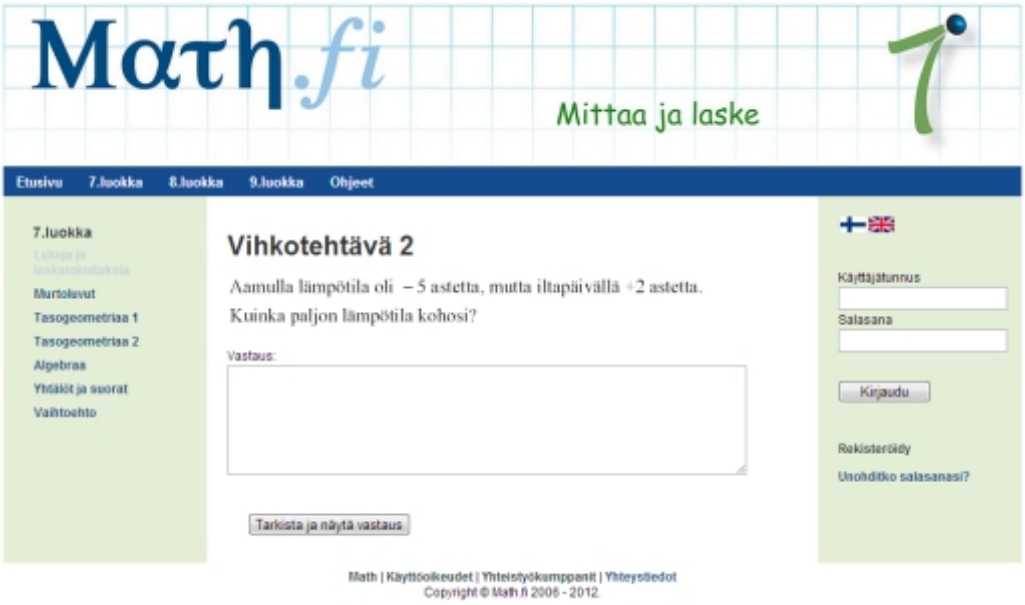
Kaavio heijastaa projektin eri vaiheiden laatua. Projektin alkuaikana keskityimme ohjelmointikielen opetteluun ja ohjelmistoympäristön asentamiseen jokaisen projektiryhmäläisen koneelle, jonka takia SVN-tallennuksia (engl. commits) ei tullut lainkaan. Viikolla 43 aloimme tallentaa kilpailun tehtäviä ja kuvatiedostoja SVN-arkistoon ja viikolla 46 pidettiin itse kilpailu.

Viikoilla 51–2 ei tullut SVN-tallennuksia joululoman vuoksi. Vuoden 2013 puolella nähdään SVN-tallennusten määrän vaihtelevan viikoittain suuremmaksi ja pienemmäksi. Tähän on todennäköisesti vaikuttanut se, että sprintin kesto on ollut aina 2 viikkoa, ja SVN-tallennukset ovat tapahtuneet aina sprintin jälkimmäisen viikon aikana. Lopussa SVN-tallennuksia tuli enemmän kuin koskaan muulloin projektin aikana, johtuen siitä, että viimeiset toiminnallisuudet piti nopeasti saada tehtyä.

Math.fi

Overview

Math.fi is a learning environment for a mathematical thinking. Main users are upper comphrehensive pupils and teachers. Idea is to take the learning away from a teacher leading teaching and make pupils more responsible of their own learning.



Organization and management

Managers:

Markus Ijäs

Hanne Korhonen

Markus Kumpulainen

Project team:

Paavo Happonen

Joonas Hartikainen

Jussi Kallava

Jesse Virtanen

We had weekly scrum meetings and open IRC-channel. MediaWiki was used for requirements engineering and Jira was for managing tasks and working hours.

Methods and tools

List all tools and methods that you used in the project and give comments about their usefulness.

JIRA for managing tasks and working hours. It was useful after project team learned how to use it.

MediaWiki was for some coding instructions and for requirements engineering.

Google Drive/Docs was very essential for writing documents and version control of documents.

Dropbox was easy way to share documents inside the project team.

Project team used xampp as a developing environment and NetBeans IDE for coding and developing.

Database: MySQL

Programming language: PHP, Javascript

Framework: CakePHP 1.3 and CakePHP 2.2

For communication project team used IRC-channel, face-to-face meetings and email.

Project phases and development model

Our development model was modified scrum with 3 week sprints. We held weekly meetings most of the time. There were separate occasions when we only met in IRC but other than that we always met face to face.

Project started 14.09.2012

Preliminary analysis meeting 27.09.2012

Project plan inspection 09.10.2012

First sprint 15.10.2012 - 05.11.2012

First review 05.11.2012

Second sprint 05.11.2012 - 26.11.2012

Second review 26.11.2012

Third sprint 26.11.2012 - 17.12.2012

Third review 17.12.2012

Fourth sprint 07.01.2013 - 28.01.2013

Fourth review 28.01.2013

Fifth sprint 28.01.2013 - 18.02.2013

Fifth review 18.02.2013

Sixth sprint 18.02 - 11.03.2013

Seventh sprint 15.03.2013 - 12.04.2013

Eight sprint 12.04.2013 - 26.04.2013

Final meeting and project ending 29.04.2013

Experiences

Inactivity among project members

We had some problems with inactivity but nothing serious. Typical reasons were othe work or school getting in the way. Next time we should try to get people more excited about the project to prevent inactivity.

Lack of time

We didn't have enough time to get all our requirements done. This was a bit unexpected because of the ground work we had to do before we could start doing new features. Next time we should keep the requirements as simple as possible and prepare to have delays.

Problems with technology

Technology caused us little problems because if a member of our team had a problem he would only bring it up on the next meeting. This resulted on delayed features.

Conflicts among the group

There were no conflicts and the group was open with suggestions.

Changing requirements

Requirements didn't change much, they only got defined better.

More work than expected

There was a lot more to be done than we expected. We had to first update CakePhp and the php version and that took a long time. Also we had to rethink the whole structure of drills and homeworks which resulted in delayed functionality.

Site crashing

The site was crashing a lot and the source of that was harder to find than expected. Our initial concern was drills which were badly coded but even though we removed them the site still crashed. What turned out was that the site had been infested with bots and we had to lock the public wiki site to fix the problem.

Statistics

Team size	Dev. model	Start date	End data	Days	Hours
3+4	Modified Scrum	12.09.2012	26.04.2013	226	1027,5

Table 1: General project information.

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	369,5	8,25	95,25	288,75	7	52,75	6,5	95,25	104,25	1027,5
%	36	1	9	28	1	5	1	9	10	100%
Usabi-lity										0
Total										**1027,5**

Table 2: Group effort by activity.

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
18	6	2	6	1	42

Table 3: Requirements and high-level design outcomes.

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
0	0	0	0	0	0

Table 4: Design outcomes.

Document	Pages	Versions
Preliminary analysis	8	1
Project Plan	22	4
Usability analysis	0	0
Requirements specification	5	98
Design plan	0	0
User interface document	0	0
Test plan	1	7

Test report	0	0
Usability test report	0	0
Final report	12	1
Project's story	5	1
Weekly reports	2	31

Table 5: Documents.

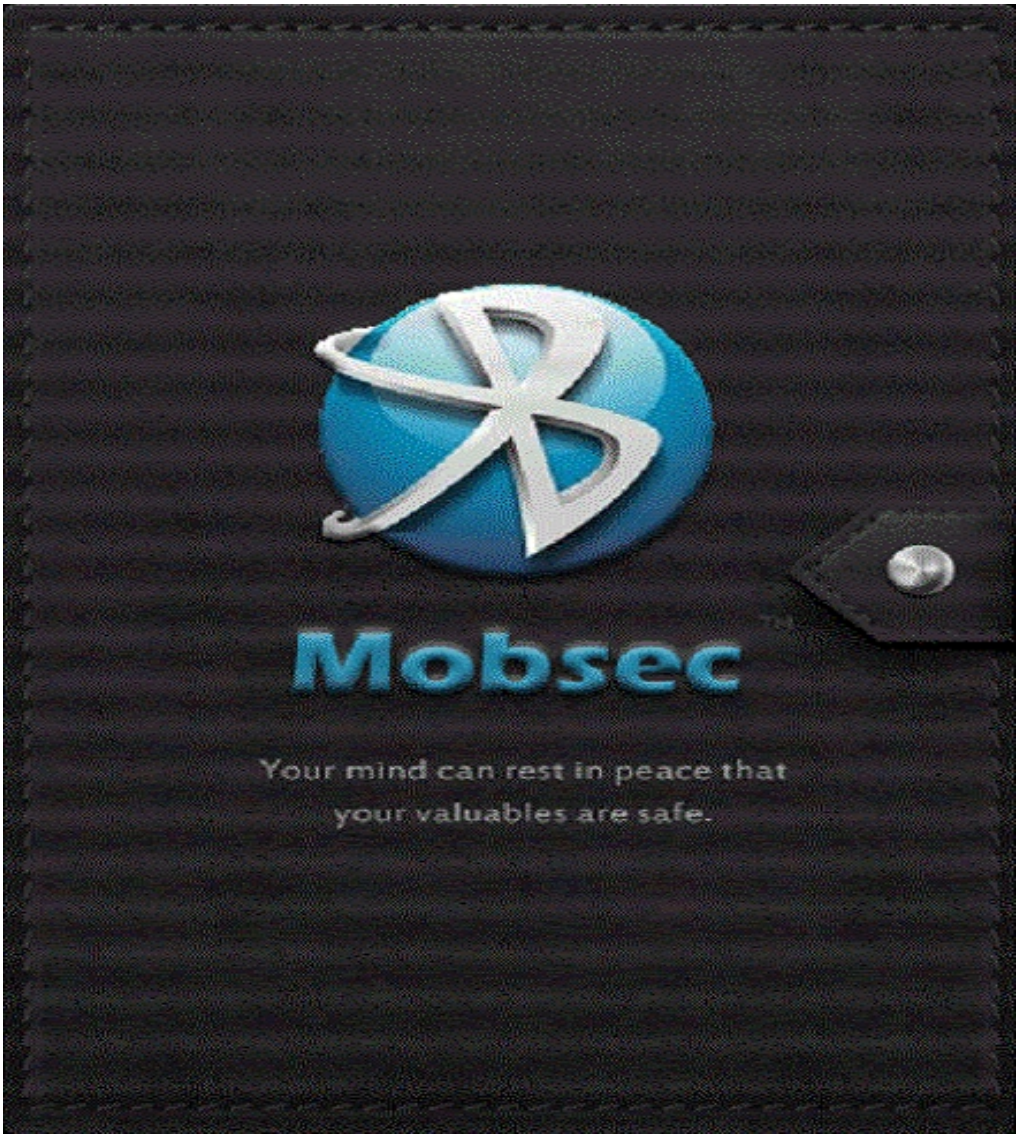
Language	PHP, Javascript
LOC	?
SLOC	?
Reused code	?
Reused and modified	?
Classes	39
Functions	95
Code revisions	928

Table 6: Codelines.

MobSec

Overview

The Mobile Security (MobSec) application is an security concept that monitors and protects all devices connected via Bluetooth, that when activated the application will monitor those devices and sound an audible alarm if any of the devices are removed from the owners vicinity, or when the user accidentally leaves a device behind.



Organization and management

Project Managers:

- Andrew Cox
- Xie Liwei
- Kotha Navaneetha

Team Members (developers):

- Kuisma Kuusniemi
- Ruibin Ye
- Zheng Qian
- Wang Chenlu

Methods and tools

MobSec was developed using Android SDK. Android phone were provided by the customer to use in the testing and other phases of the project. JUnit android testing framework is used for writing test cases.

Redmine is to be used for requirements management and timekeeping. The wiki is used for sharing documents, keeping track of the schedule, writing meeting minutes and for other information. https://redmine.sis.uta.fi/projects/mobsec/wiki.

SVN was used for version control.

Skype and email are used for communication.

The tools used for documents are Microsoft Word and Open Office.

Project phases and development model

We decided to run with the lean scrum but within a first couple of meetings we realized that we could not run lean scrum the reason been the face to face communication problems. Being a multicultural team and although English is the main communication in the meetings, it can be clearly seen that when individuals are asked to repeat the task or problem they were unable to. In a fulltime working environment time and money would be invested in language and communication training as well as team work. With the University environment it was not possible even coaching was limited.

After reviewing development models, it was decided that we would go with water fall model for management tasking.

The three phases that we concentrated are:

Requirements: In the initial stages of the development we met with the customer and gathered requirements. Based upon the agreed vision, we grouped and prioritized the requirements as high level Features. These features were further broken down to user stories, which are development level task. In the later stages of the development we met with the customer and scoped down the requirements with respect to time and effort that could be provided. In this phase some of the requirements were rejected as

agreed with the customer.

Implementation: We had a regular team meeting and initially we established the key strength of the developers according to their interest. In Every team meeting mangers had the prioritized list of user stories need to be developed and the developers has the freedom to choose the user story or task they are interested from the top of the list. User stories were easily tracked based on the prioritized list. This was main reason for meeting with the customer and rescoping requirements as ambition level was too high for the velocity we were achieving.

Hardening: In the later stages we transitioned to bug fixing and testing. From exploratory testing bugs were created and prioritized. Within the team meetings and workshops, developers focus would be to work on the high prioritize bugs. Specific bugs were also raised as for test asset creation. We were then able to scope development to bug fixing or test creation depending upon bug priority. Through hardening phase we also concentrated on the user documentation.

The project milestone was met on time as expected and can be seen in the table below:

Date	Scheduled Task
21.9.2012	First meeting with the project team
18.9.2012	First meeting with the client
27.9.2012	Preliminary analysis returned
28.9.2012	Preliminary analysis review
6.10.2012	Project plan returned
12.10.2012	Project plan review
7.11.2012	Personal report I
28.11.2012	Midterm presentation
16.1.2013	Personal report II
13.3.2013	Final presentation
18.3.2013	Final review

Experiences

The project was successful. We had a really good working team. Every team member was enthusiastic, committed and contributed to the project. One of the project managers has real time software management experience and that helped for the success of the project.

One of the risks met during the project is the communication problem. Being a multi culture team, sometimes verbal communication was also a problem. Skype, team meetings, retrospective and workshops helped managed this risk.

We think testing should go simultaneously as development and that's the area we are

going to concentrate better the next time.

Statistics

Team size	Dev. model	Start date	End data	Days	Hours
3+4	WaterFall Model	14.9.2012	18.3.2013	181	**1151.7**

Table 1: General project information.

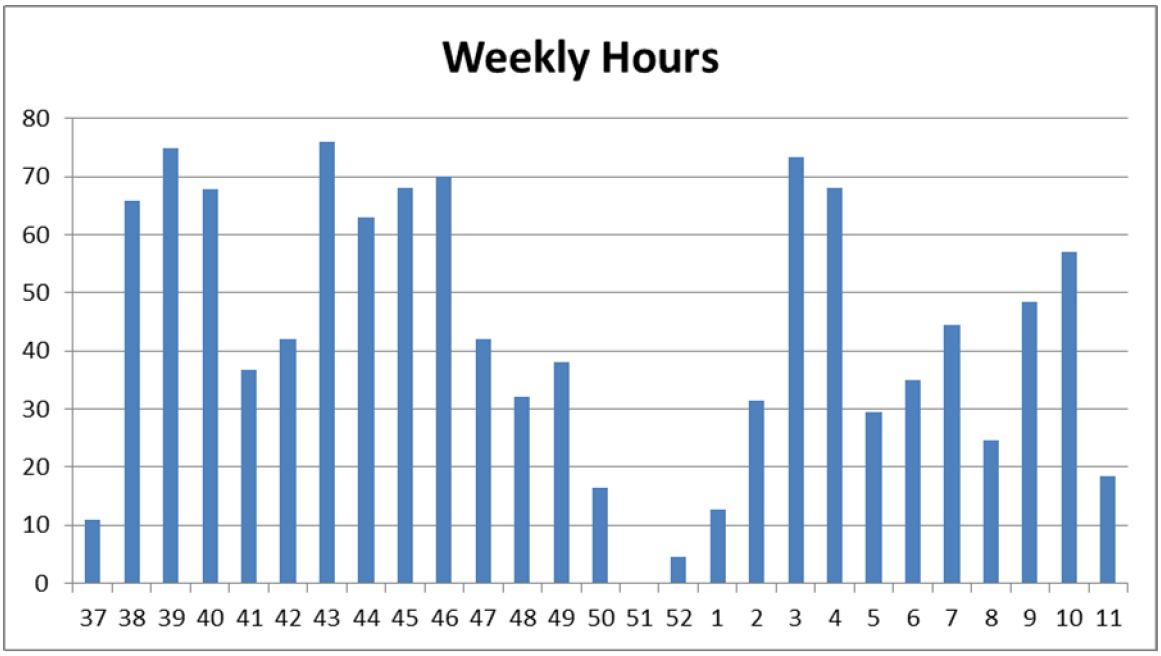


Chart1:Working hours per Week

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	529.80	14.50	5.50	230.80	40.50	2.50	70.00	219.50	38.60	1151.7
%	46	1.2	0.4	20	3.5	0.2	6	19	3.3	100%
Usabi-lity	-	-	-	-	-	-	-	-	-	-
Total	529.80	14.50	5.50	230.80	40.50	2.50	70.00	219.50	38.60	**1151.7**

Table 2: Group effort by activity.

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
15	-	-	4	-	-

Table 3: Requirements and high-level design outcomes.

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
-	-	-	-	-	-

Table 4: Design outcomes.

Document	Pages	Versions
Preliminary analysis	9	1
Project Plan	15	1.1
Usability analysis	-	-
Requirements specification	-	-
Design plan	-	-
User interface document	-	-
Test plan	-	-
Test report	-	-
Usability test report	-	-
Final report		
Project's story	5	1
Weekly reports	1	25
MobSec Bluetooth Concepting Discussion	9	1

Table 5: Documents.

Language	Android
LOC	1328
SLOC	-
Reused code	-
Reused and modified	-
Classes	14
Functions	71
Code revisions	211

Table 6: Codelines.

Pricing tool

Overview

The project's aim was to develop a tool for creating a Word-based opportunities. The customer company of the project is M-Files, which is specialized in developing of document management systems. Today majority of vendors create proposals inside the ERP system, which has its own limitations. Mostly these limitations are related to structural constrictions of the document.

Those proposals created by ERP systems tend to be system-specific and thus lack in those rich functionalities we are used to see in Word documents. The customer's vision for creating an opportunity is to provide sales personnel a traditional Word-oriented interaction model with necessary adjustments allowing retrieving of products without the need to open external supplementary documents.

Based on the customer's vision our team ended up to implement the tool as Word add-in. Although it might appear as obvious choice for underlying architecture there still remained many unknown factors and risks that had to be considered carefully. The biggest concern was the integration with Microsoft Dynamics, which was later rejected by customer.

The end product turned out to be very close to the goals our team had set. One of the major criteria was to make interaction with the tool as intuitive as possible. To do that our team had to find the proper approach to hide the complexity of offer making process. The GUI had an important role in this task by presenting the complex technical and logical structure in easily readable and user-friendly way. The screenshots below demonstrate the tool window (Fig 1) and the output of taken action (Fig 2).

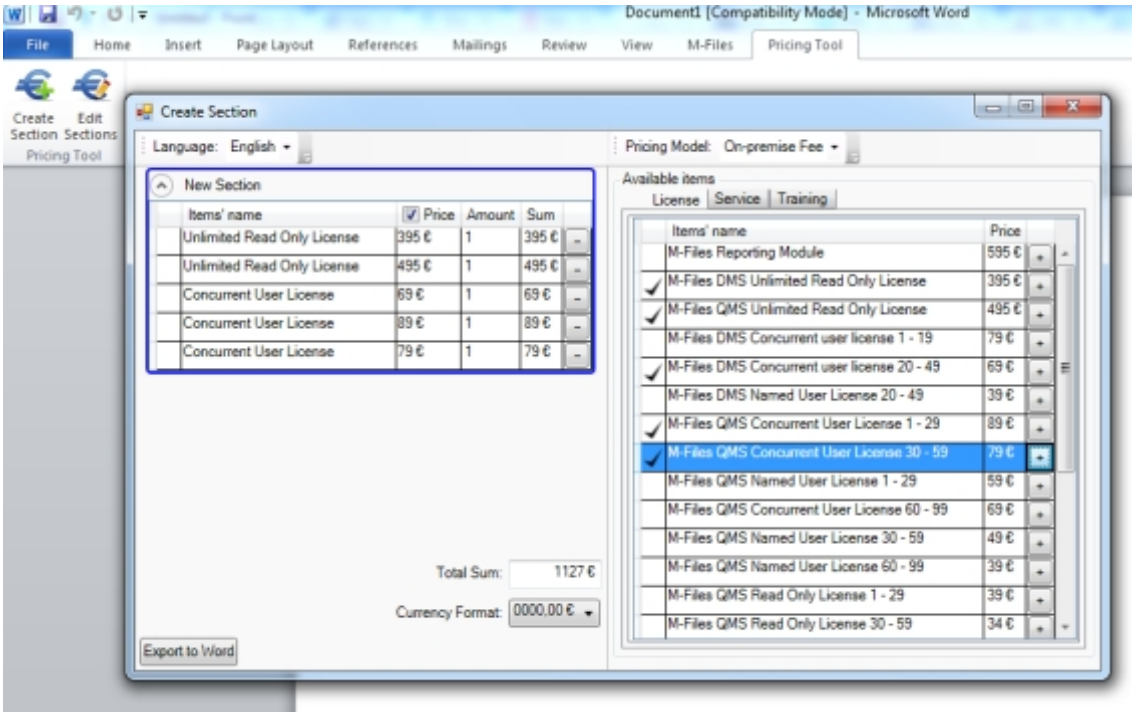


Fig 1.

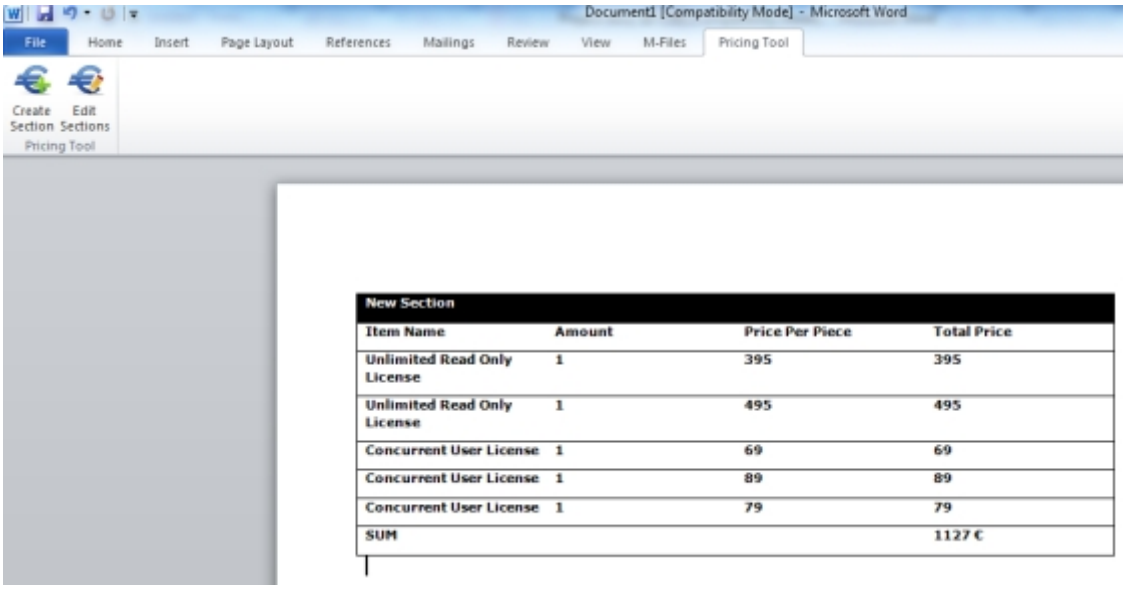


Fig 2.

Organization and management

Our project group consisted of 4 managers and 5 developers. One of the managers had previous sales experience in the Customer Company which allowed us to understand and asses better both technical and user requirements. In the following is a list of group members and roles.

Project managers:
Miia Ketolainen, Teemu Keskinen, Mirjan Merrukko and Elvis Okemou.

Developers:
Iulia Adomnita, Ville Murtonen, Reza Ahmadi, Elina Leino and Krzysztof Kachniarz.

Many of the tasks were performed collectively. In addition team members were assigned an area of responsibility. From managers Miia guided and monitored the coding process. Teemu's area of responsibility was creation and improvement of database along with customer relations. Mirjan monitored the overall progress of the project and prepared a variety of documentation. Elvis maintained the requirement specification document and governed the GUI team.

From developers Iulia worked on development and implementation of GUI's code and appearance. Ville contributed in setting up the developing environment and coding. Reza worked on code and ensured the integrity of application architecture. Elina conducted the usability analyses and application tests. Krzystof worked on database structure and sample data preparation.

Most of the decisions and task assignments were made in the weekly meetings. In the beginning of the project an agenda was created for each meeting. In practice this approach turned out to be impractical as each meeting usually raised a great amount of new issues. The issues covered in the weekly meetings were written to minutes.

Methods and tools

One of the customers' wishes was that the team makes as extensive use of Microsoft technologies as possible. This affected the selection of tools and approaches that were made during the project. Generally speaking all the tools selected for the project had to be compatible with Microsoft technologies. In the following is the list of utilized tools:

- Visual Studio 2012
- Visual Studio 2010
- C# programming language
- M-Files 9.0.3372.6
- Microsoft Word 2010
- SVN, TortoiseSVN –client
- Skype
- WPF
- .NET Framework 4.0

Visual Studio was used to build the frame of the application. All developers were familiar with C# making it natural choice as the programming language. M-Files served as the document warehouse for documentation produced during the course. In addition to a basic document storage M-Files was also used for hour reporting. To some extent M-Files was used for task assignments too.

M-Files also provided an intrinsic support for building a database. This helped the whole team a lot as team members could easily access and evaluate the database from their M-Files client application. To avoid possible Word version incompatibilities the Word 2010 was chosen as version used for development.

SVN was used for version control. Skype proved to be handy especially in the beginning of the project when general structure and overall architecture was in an early stage. WPF's role was to facilitate the development of GUI.

Project phases and development model

The development process was divided into seven sprints but instead we ended up developing the tool according to the waterfall model. This was mainly because the client wanted us to design the GUI, dataflow, architecture and the database completely before writing a single line of code. Each sprint was designed to proceed from preceding phase. In this respect the developing model differed from traditional Scrum framework. Other deviations from Scrum concerned the meeting policies. Because of busy schedule of group members the number of onsite meetings had to be reduced which in turn increased the importance of remote collaboration. The original Scrum-based schedule is presented in the table 1 below. Instead of this plan we started the implementation in December. Fortunately the GUI prototypes were already created by coding so we were able to use the code already written. The implementation was ready in the beginning of March, after which the testing began.

Phase	Start and end dates	Length	Description
Sprint 1	5.10.2012 - 26.10.2012	3 weeks	Design. First proposal for GUI, architecture and data model is presented to the client. The SVN will be up and running in the end of the first sprint.
Sprint 2	26.10.2011 - 15.11.2012	3 weeks	Design and implementation. GUI and architecture design will be ready during the second sprint. Implementation begins.
Sprint 3	16.11.2012 - 6.12.2012	3 weeks	Implementation and testing.
Sprint 4	07.12.2012 - 10.1.2013	5 weeks	Implementation and testing
Sprint 5	11.1.2013 - 31.1.2013	3 weeks	Implementation and testing
Sprint 6	1.2.2013 - 21.2.2013	3 weeks	Implementation and testing
Sprint 7	22.02.2013 – 15.3.2013	3 weeks	Testing and bug fixes

Table 1. The original schedule of the project.

Experiences

Generally the experiences reported by team members were mainly positive. There were also critical views. Despite the importance of achieving success it is equally importance to receive a constructive criticism. The following is an extract of feedback given by group members.

Positive feedback:
One of the managers considered the experience of going through an actual project as invaluable. The course should strive to maintain this arrangement and provide the chance for students to get involved in real projects (not necessarily with companies).

One of the developers was so satisfied that called this course “a lifesaver”, as this provided some real world experience which can be displayed on the developer's CV.

One other member was happy to be able to learn about project arrangements, various tools and tasks while getting the necessary amount of credits.

Criticism:
One of the members was extremely negative on the usefulness of the course and suggested that it should be replaced with innovation project which is “much better”.

Two managers suggested that there should be a maximum of two managers per project. A limit on the number of participants or smaller groups was suggested to accommodate this. Also having some weeks in advance to prepare for the projects would be beneficial for managers.

Statistics

The content of this chapter mostly consist of collection of different graphs and tables reflecting the content and progress of the project using graphical notation.

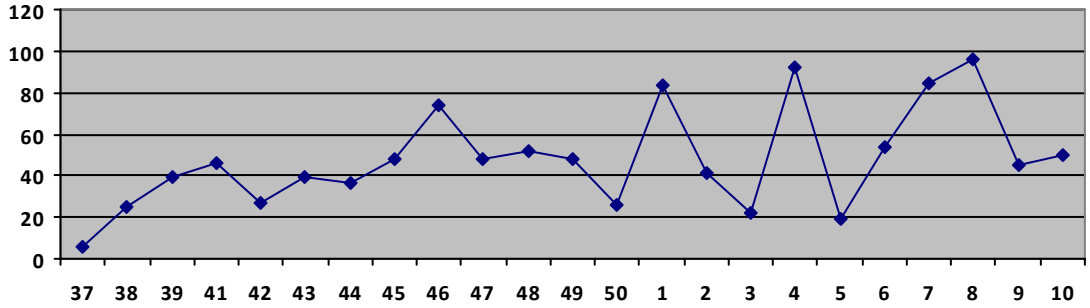


Fig 3.

Team size	Dev. model	Start date	End date	Days	Hours
4 + 5	Scrum	05.09.2012	15.03.2013	197	1162,5

Table 2: General project information.

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	398,25	38,75	76,5	241,5	75,5	3,25	24,75	223,8	158,5	1241
%	32,09	3,12	6,16	19,46	6,1	0,26	1,99	18,03	12,77	100%
Total										**1241**

Table 3: Group effort by activity.

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
17	2	3	9	1	10

Table 4: Requirements and high-level design outcomes.

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
	1	0	2	0	1

Table 5: Design outcomes.

Document	Pages	Versions
Preliminary analysis	7	1
Project Plan	17	1
Usability analysis	6	1
Requirements specification	2	1
Design plan	3	1
User interface document	0	0
Test plan	5	1
Test report	4	1
Usability test report	0	0
Final report	19	1
Project's story	7	1
Weekly reports	24	

Table 6: Documents.

Language	C#
LOC	2245
Reused code	0
Reused and modified	448
Classes	16
Functions	107
Code revisions	205

Table 7: Codelines.

Meta review tool

Overview

Meta review tool is a peer review program for TTT (Tietotekniikan taidot[1], also T3) course assistants and teachers. Tailored software was in need as Google Docs didn't scale up that well with many simultaneous editors.

In TTT course, there are tasks that students must complete. Every task must be reviewed by another student and these reviews are examined by the assistants. If the review is incomplete, student must supplement it.

With Meta review tool, it is possible to give an annotation regarding the incomplete or bad review. It is also possible to track whether this re-review gets fulfilled in an appropriate manner and on time.

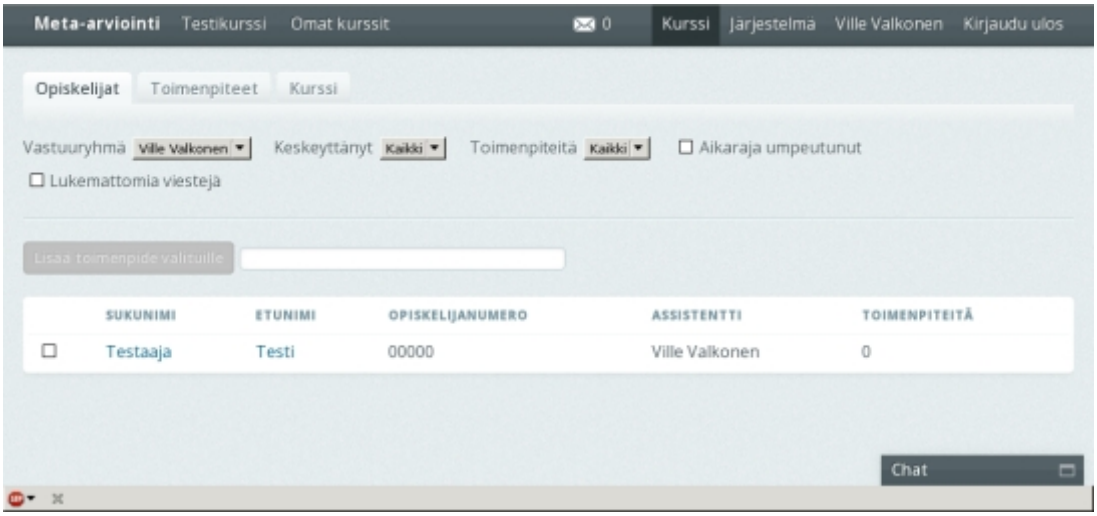


Figure 1. Screencapture of the Meta review tool

Organization and management

There were three project managers, three developers and one UI/UX designer. At the very start of the course, one of our project managers had to quit due the busy schedule at work.

[1] Free translation: *information technology skills*

Projet management

Risto Salo

- 4th year CS student
- Studies information systems as a major
- Works a part-time web developer at Descom Oy (front and back end developer)
- Gained experience on e-commerce development through work at Java
- Has gained skills of PHP, databases and Java during studies and freetime
- Been an assistant in TTT course for 1,5 years and also lectured the course once

Ville Valkonen

- 5 - 6th year CS student, counting style varies
- Software engineering as a major
- On freetime writes C and Python
- Past work experience of bit protocols with Java
- IT Engineer/Junior admin at Qualcomm Atheros

Developers

Janne Kallunki

- 3rd year CS student
- Works a full-time front-end and UI developer at Leonidas
- Likes mostly programming with HTML5/CSS3/JavaScript
- Handles also graphical designing
- Back-end experience of Javasta and PHP, also familiar with databases

Mika Tuunanen

- Nth year CS student
- Gained experience through studying and by own interest
- Knows PHP, [My|Postgre]SQL

Joni Hämäläinen

- 3rd year CS student, had studied CS for an year in the university of Jyväskylä before the switch
- Gained experience through studying and by own interest
- Knows Java, PHP and databases

Ella Kaugin

- 3rd year sociology student but CS as a minor
- Have studied math for two years in university of Helsinki
- UI and UX skills

We settled one face-to-face meeting a week and used IRC and emails for rest of the communication. In IRC, *#meta-arviointi@IRCnet* channel was created for the project communications. It was used to discuss about implementation specifications, ambiguous task descriptions and so on. Email was used merely for reporting.

We also discovered that sprint pool works better for us. In sprint pool, every developer can choose somewhat freely what tasks they will complete. Tasks were handled by Github's issues tracker.

Information regarding the meetings, more accurate specification (in Finnish), UI mock ups, statistics and meeting memos can be found from:

https://github.com/Meta-arviointi/meta-arviointi/wiki

Methods and tools

Project was developed by using a PHP programming language and a CakePHP framework. The used tools and services is listed on below.

IDE:

NetBeans

Widely used IDE that has support for several languages. Was chosen mainly because everyone has used it.

Programming language:

PHP

Check the next section.

Framework:

CakePHP

The choice was between FuelPHP and CakePHP as Janne (one of our developers) had been using these before. CakePHP was chosen because of the better database support for our needs.

Database:

PostgreSQL

Future of the MySQL is unpredictable since Oracle took over it. Postgresql values security, data integrity and reliability, all important features of the well designed

database system. Also, Postgresql has many built-in functions that can ease the development process.

SCM:

Git

Efficient distributed version control tool that has good support in Netbeans.

SCM hosting service:

Github

Superior hosting company who offers a free repository storage, a simple built-in issue/bug/requirements management, and interactive graphs and plots for project's efficiency monitoring.

Unit testing:

PHPUnit

Few of our developers had previous experience of this.

Source code analyzing:

Sonar

Ville had previous work experience of using this for improving source code quality.

Lines of code:

CLOC

Simple and clear, performs the needed calculations.

Public repository of the project:

https://github.com/Meta-arviointi

Issue and bug tracking: https://github.com/Meta-arviointi/meta-arviointi/issues/milestones

We didn't want to use any separate system for issue and bug tracking. Also, Github's issues were enough simple so that we were able to modify those to fill our needs perfectly. It was also possible to make milestones which were linked in to stable branch versioning.

Documentation

Google Docs

Google docs was chosen because it enables simultaneous editing, easy sharing and versioning of document files.

Project phases and development model

We used modified Scrum for the project management. We settled only one meeting per week (since this is an university course versus work life) and during the second period we started to use a sprint pool. With the sprint pool team members were able to choose freely what tasks they will complete. Tasks were chosen from the issues list that was/is available via Github (https://github.com/Meta-arviointi/meta-arviointi/issues?state=open). There were total five sprints during the course timespan. That makes it roughly a month for each sprint.

PM = Project management, All = PM + other team members

Happening	Invited	Where	When
Short project presentation	PM	B1029	19.9.2012, 12:25
Weekly meeting / Workshop of tools to be used	All	B3136	25.9.2012, 16-19
Preliminary analysis	All	B3136	26.9.2012, 16-17
Weekly meeting	All	B3136	02.10.2012, 16-18
UI/UX Workshop	Ella, Janne, Risto	B1029	08.10.2012, 14:30-16:45
Start of the 1st sprint	All	B1065	10.10.2012, 16-18
Database workshop	Ville, Massimo, Joni, Mika	Linna 3023	10.10.2012, 14:30-16:00
Weekly meeting	All	B1065	10.10.2012, 16:00-17:00
Review of the project plan	All	B1029	12.10.2012, 8-10
Weekly meeting	All	B0019	24.10.2012, 16-18

Weekly meeting	All	B2077	30.10.2012, 16-18
Review of the 1st sprint / **Start of the 2nd sprint**	All	B1065	06.11.2012, 16-18
Weekly meeting	All	Linna 3017	13.11.2012, 16-18
Weekly meeting	All	B1065	20.11.2012, 16-18
Recreational activities	Voluntary	Höyry	20.11.2012, after the weekly meeting
End of the 2nd sprint / **Start of the 3rd sprint**	All	B2077	27.11.2012, 16-18
Project presentation	All	B1097	28.11.2012, 13:40-14:00
Weekly meeting	All	B2077	04.12.2012, 16-18
Weekly meeting	All	B2077	11.12.2012, 16-18
End of the 3rd spinrt / **Start of the 4th sprint**	All	B2077	2.1.2013, 16-18 (proposed)
UI/UX related meeting	Ella, Risto	B1065	8.1.2013, 14:45-16
Weekly meeting	All	B1065	8.1.2013, 16-18
Weekly meeting	All	B1065	16.1.2013, 16-18
Hackathlon	All devs who do programming and PM	Office of Leonidas	21.1.2013, 17->
Review of the 4th sprint / **Start of the 5th sprint**	All	B1065	23.1.2013, 16-18
Hackathlon	All devs who do	Office of	28.1.2013, 17->

	programming and PM	Leonidas	
Weekly meeting	All	B1065	30.1.2013, 16-18
Weekly meeting	All	B1065	6.2.2013, 16-18
Weekly meeting	All	B1084	13.2.2013, 16-18
End of the 5th sprint / Start of the finalizing sprint	All	B1084	20.2.2013, 16-18
Weekly meeting	All	B1065	27.2.2013, 16-18
Final project presentation	All	B1097	13.3.2013, 13:40-14:00
Weekly meeting	All	B1065	13.5.2013, 16-17

Document name / document	Latest version (date)	Started	Pages
Sprint-pool - list of tasks (Sprintpool)	v.1 (16.1.2013)	6.11.2012	3
Project plan (Projektisuunnitelma)	v.0.7.4 (9.5.2013)	3.10.2012	24
Preliminary analysis (Preliminary analysis)	v.1 (6.10.2012)	22.9.2012	7
Document of filterable tables in UI (Filtteröitävät taulukot)	v.1 (13.3.2013)	28.2.2013	2
Observations of usability test analysis (Käytettävyystesti - havainnot)	v.1 (2.4.2013)	5.3.2013	4
User story (MA-kayttajatarinat)	v.0.4.1 (9.5.2013)	21.9.2012	3

Sprint backlogs (MA-sprint-backlogs)	v.1 (6.11.2012)	3.10.2012	3
Short project presentation (Short project presentation)	v.1 (19.9.2012)	18.9.2012	3
Midterm presentation (Meta review presentation #2)	v.1 (28.11.2012)	27.11.2012	8
Final project presentation (Meta review tool - Final presentation)	v.1 (10.3.2013)	10.3.2013	11
Email templates (Meta-arviointi - Maili-templatet)	v.1 (4.3.2013)	30.1.2013	3
Usability testing cases (Kaytettavyystestaus)	v.1 (9.5.2013)	3/2013	6
Usability testing testing plan (testitaulukko_kayttoliittyma)	v.1 (9.5.2013)	1/2013	9 (tables)
UI mock ups (several images in directory *Rautalankamallit*)	v.1 (9.5.2013)	9/2012	23 (images)
Final report (Meta-arviointityökalu - Loppuraportti)	v.0.7.1 (10.5.2013)	6.5.2013	28
Working hour accounting (Meta-tuntikirjanpito)	v.1 (10.5.2013)	9/2012	8 (tables)

* There is also meeting memos on every weekly meeting (includes reviews and workshops), totaling 26. These documents can be found under *Palaverimuistiot* on: https://github.com/Meta-arviointi/meta-arviointi/wiki

and detailed list of the tasks closed during the sprints:

https://github.com/Meta-arviointi/meta-arviointi/issues/milestones?state=closed. Only 2 and 3 sprints are shown since we made a switch to the sprint pool methodology around 3th sprint. Sprint pool milestone and can be seen on https://github.com/Meta-arviointi/meta-arviointi/issues?milestone=6

Experiences

The course taught important functionality and methodology regarding the work life. Scrum, version control systems, release model and so on, are few of the concepts that some people will learn at the very first time during the course.

Foreseen risks

Here we discuss about the two most critical and somewhat obvious risks that could happen. Hereby, this is not a full list of risks.

Since there were new tools, frameworks and concepts, we recognized this could exhaust some of the team members. Also, the course may take vast amount of time so there are possibility that someone could drop the course due lack of time.

The latter risk mentioned above, lack of time, concreted shortly after the course started. Massimo, one of the project leaders, had a rush in the work and couldn't find needed time for the course. Therefore he resigned. Although this had a little affect for the project, we managed to keep wheels running. On a positive side, this could have been one of the developers and that have had been more crucial. The former risk, lack of motivation, happened occasionally during the course. It is hard to come with a working rewarding system if motivation ceases. ECTS depending on hours, works for somebody but not for all. Nevertheless, we achieved a lot. Moreover, we think it is completely normal to have regression or suffer from procrastination during the course as many of the students have full-time jobs, families and other courses going on concurrently.

Overall we think our team was greatly successful and proceed better that what our initial expectations were.

Unforeseen

There were no unforeseen risks during the course period.

Statistics

Team size	Dev. model	Start date	End data	Days	Hours
2+4	Modified scrum	21.09.2012	22.04.2013	214	1118

Table 1: General project information.

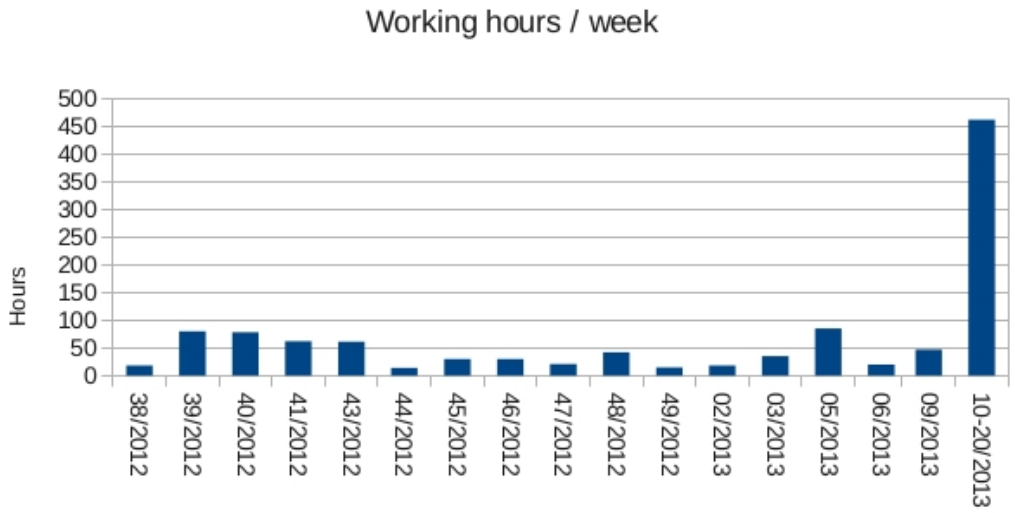


Figure 1. Working hours per week

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	416	15	79	379	77	41	1	60	53	1118
%	37,51	1,38	7,28	32,44	7,10	3,78	0,09	5,53	4,88	100%
Total	407	15	79	352	77	41	1	60	53	**1118**

Table 2: Group effort by activity.

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
48	-	48	21	1	14

Table 3: Requirements and high-level design outcomes.

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
-	-	-	-	-	-

Table 4: Design outcomes.

Document	Pages	Versions

Preliminary analysis	7	4
Project Plan	28	16
Usability analysis	5	2
Requirements specification	-	-
Design plan	-	-
User interface document	6	1
Test plan	-	-
Test report	-	-
Usability test report	5	1
Final report	28	10
Project's story	12	6
Weekly reports	21	1

Table 5: Documents.

Language	PHP, CSS, Javascript, SQL, CoffeeScript, XML, Python, Bourne, make
LOC	259 072
SLOC	160 167
Reused code	-
Reused and modified	-
Classes	-
Functions	-
Code revisions	414

Table 6: Codelines.

Smart Lightning

Overview

Smart Lightning is a software system which can help people to control the lights by their acts to get a vivid effect in the presentation. The project idea is all about how to develop applications that use the underlying light controlling system and some sensors in order to help people to get thing done easily, smartly. This is the idea from our project's client, TAUCHI Research group in University of Tampere. After having some discussions with our client and spending time on understanding the project, our group had a brainstorming session about finding out some applications based on the idea. We came up with the application named smart presentation room. That is by using light controlling system provided from our client and Kinect motion sensor; our application would help the presenter to be able to control the lights in the room himself. In detail, the presenter will use his own gestures to control the light brightness in the room, make the light focus on him and automatically direct the light beam follow him when he is moving around.

Organization and management

Person	Experience	Team	Job
Bo Huang	Project Manager	Kinect Team	Manage, Integration, Debug, Testing, Documentation
Zhang Zhang	Project Manager,	Kinect Team	Manage, Integration, Debug, Testing, Documentation
Trong Nghia Nguyen	Project Manager, Scrum Master	Low-level Server Team	Manage, Integration, Documentation, Coding(Connection with LLS)
Pawel Paradowski	Developer	Kinect Team	Coding(Develop gesture identification), Integration
Renfei Zhou	Developer	Kinect Team	Coding(Develop gesture identification), Integration, Testing
Thrushna Nalam	Developer	Low-level Server Team	Coding(Connection with LLS), Integration
Tianyi Hu	Developer	Low-level Server Team	Coding(Connection with LLS), Integration, Debug, Testing
Ao Li	Developer	Kinect Team	Coding(Develop gesture identification), Documentation, Testing

Methods and tools

Communication on Internet:
Wiki and Redmine:
Function: To communicate with documents of the project and requirements management, and the working hour can be checked on Redmine.
Level of Useful: High.

Skype
Function: Online communication among the team members.
Level of Useful: Low. (Because of the bad Internet quality)

Email
Function: Keep communication in the team by email, especially for weekly report.
Level of Useful: High.

SVN
Function: Upload and download the document of code.
Level of Useful: Medium. (USB is more popular)

Developing Tools:
(1) Kinect sensors for developers doing research at home (at least 1 device)
(2) Low-level light controlling system
(3) Kinect tool for detecting skeleton data
Project phases and development model
Modified SCRUM is used for this project, which is an iterative and incremental agile

software development framework for managing software projects and product or application development. Scrum focuses on project management institutions where it is difficult to plan ahead. Mechanisms of empirical process control, where feedback loops that constitute the core management technique are used as opposed to traditional command-and-control management. Its approach to planning and managing projects is by bringing decision-making authority to the level of operation properties and certainties.

Project Phases	Date Start	Date End
Define requirements	14/09/2012	28/09/2012
Finishing knowledge study(C#, Kinect, etc.)	10/2012	11/2012
Design 1st gesture	11/2012	12/2012
Design 2nd gesture	11/2012	12/2012
Integration	12/2012	12/2012
Connection with LLS	01/2013	01/2013
Alpha release	01/2013	01/2013
Debugging	20/01/2013	01/02/2013
Beta release	02/2013	02/2013
Debugging	01/03/2013	11/03/2013
Final version	12/03/2013	13/03/2013

Experiences

Expected Risks	**Risk Mitigation**	**Impact**	**Certainty**
Team member leaves the project	None Can only provide peer pressure and motivation.	HIGH	MEDIUM
Team member cannot commit sufficient time as originally indicated	Monitor, discuss and motivate. Encourage bidirectional trust to highlight early and take actions	HIGH	MEDIUM

Bad management practices	Peer review, retrospectives must cover development and management	HIGH	LOW
Resources unavailable in time	Active highlight and communication of risks	HIGH	LOW
Improper and unclear task division between project managers.	Clear definition of tasks, and active use of retrospective techniques to improve.	MEDIUM	LOW
Technology is new and could slow progress to all team members	Peer discussion, personal learning, Good Communications to help problems	MEDIUM	LOW

Experience:
Arrange more time on Self-learning.
Most of our team members have no experience of using C#. So they need more time to learn related knowledge.
Managers also need to learn SCRUM.
Solving problems.
We should also arrange time for working together to resolve some common problems, maybe about one time every two weeks.
Completed documentation:
We get some documents from our user; it is about the basic framework. We divided our developers into two small groups so that they can learn necessary knowledge as quickly as they can.
Statistics

Team size	Dev. model	Start date	End data	Days	Hours
3+5	SCRUM	14/09/2012	15/03/2013	182	1266

Table 1: General project information

Activities	Sep-12	Oct-12	Nov-12	Dec-12	Jan-13	Feb-13	Mar-13	Total
Planning and management	60.5	72.5	47.5	32	21	29	44	306.5
Code		20	74.5	102	87	169.5	20	473
Design		10	8.5	2	3			23.5
Studying	4	31.5	23	29	49	79.5	41.5	257.5
Other		3	8.5	4		13	79	107.5
Integration and testing			2		4	42	85	133
Review			2	5	7	17.5	7	38.5
Requirements specification		4			3			7
Repair							15.5	15.5
Total	64.5	141	166	174	174	350.5	292	1362

Table 2: Group effort by activity

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
4	3	4	NO	NO	NO

Table 3: Requirements and high-level design outcomes

Document	Pages	Versions
Preliminary analysis	7	1.0
Project Plan	13	1.2
Usability analysis	4	1.1
Requirements specification	2	1.0
Design plan	No	No
User interface document	No	No
Test plan	5	1.1
Test report	3	1.1
Usability test report	No	No
Final report	16	1.1
Project's story	5	1.1
Weekly reports	26	1.0

Table 4: Documents.

Language	C#, C++,
LOC	1470
SLOC	1470
Reused code	0
Reused and modified	0
Classes	48
Functions	Hard to Calculate
Code revisions	Beta

Table 5: Codelines

Virtual Patient

Overview

Virtual Patient is web browser application designed to help medical students learn medical reasoning. This application opens an open platform for the users to create, assign, and grade different tests in virtual environment.

Students can take tests assigned to them, or select open tests to complete at any time. Teachers can grade submitted tests and give feedback with them.

Organization and management

- Ahmed Dawood, MySQL database developer.
- Hao Hu, UI developer.
- Sundar Kunwar, Documentation Manager.
- Xiaolan Ma, Tester, UI development assistant..
- Santtu Mansikkamaa, Support Manager.
- Johannes Pitkänen, Tester.
- Mohammad Ejazudding Syed, Code Manager, Communications as well at the second half of the project.
- Chenyu Wei, UI developer.
- He Ye, Communications Manager, UI team lead.

Our project team used variety of different ways of management. We ended up using a method, where He Ye was lead of UI development team (Hao Hu and Chenyu Wei). Ye, Hu and Wei had coding meetings nearly daily, usually weekly for the most of the project. Ma also joined these meetings for coding support and testing reviews. Ejazuddin Syed was overseeing the code quality and conventions throughout the project, and guided the developers to improve.
This UI development team was in close contact with Ahmed Dawood for database integration work. Dawood created the database used in the project.
Other ways of management were through Redmine issues as a reminders of planned work, and reporting tool of daily progress.
Our main method was weekly meetings, where we discussed current status and planned for the upcoming week. Skype and e-mail were used during the week to discuss and plan work.
Santtu Mansikkamaa was appointed as product owner, and he created the priorities for implementation work with the client.

Methods and tools

TortoiseSVN: Easy to use graphical version control application that can be tied to development environment easily. It is always clear to see who and what files were changed, and see how the procedure goes. Using TortoiseSVN is convenient because of its GUI, and user doesn't need to remember all commands and syntax.

Jacob Nielsen's heuristics were used for UX evaluation. These are very useful in any application as always.

Task-oriented use case testing. Functionality testing. Relevant and useful for testing purposes.

Redmine was used as wiki, version control followup and work followup system. Very nicely created compact application that is very versatile for developers and managers alike. In management view it is very powerful tool in distributing work and following it, and creating reports. But only if everybody uses it effectively.

Google docs for distributed document editing. Very good especially when multiple people want to contribute or follow. Even has version control.

Editplus was used for HTML and PHP code. It is small, green, convenient and easy to use.

WAMP is a virtual website framework software, which can be used to testify and connect team mers' code. It can also contribute to a complete website.

Balsamiq mockups (www.balsamiq.com) is a tool that can be used to create mockups of UI's. The style of the mockups is close to hand drawn on a piece of paper. This tool was used by our designer throughout the project, but it was not very convincing. On one occasion one PM needed to create a design, but quickly moved to pen & paper approach, as the mockup designing speed was much faster, and quality almost similar.

LibreOffice was used as main document editing tool in addition to google docs. Even though LibreOffice is the demonspawn that will ruin your day, it is free, and better than OpenOffice in many ways.

Project phases and development model

We used modified Scrum in the project. We didn't see a need for very heavy Scrum processes, and the work was even more agile. It was a bit too light for some, although it made managing the project a bit lighter without the need of burn down charts and other similar tools. Once we used Skype to replace weekly meeting, but it was too messy and sound quality was awful and we had to continue meetings face to face.

We didn't have phases per say. We spent the time best we could to create as many features as possible. We moved to next feature when ever previous was finished. We had defined the minimum set of functions for the application, and also other features. When the minimum was fulfilled we started to implement other features on top of it.

This method created light and easy process for the team. It created us the ability to demonstrate our progress with self guided priorities in each review meeting. Client representative was quite often in our weekly meetings giving feedback and guiding to right direction.

Experiences aka horror stories

The following chapters introduce unforeseen risks and realized risks.

Unforeseen risks

Our main unforeseen risk that realized, was the risk of multiple personnel of same responsibility being unable to provide implementation. We had anticipated mitigation plan for leaving personnel, but we did not expect that all personnel responsible for implementation could be very difficult to work with. They were often absent from meetings, they did not provide what they committed in meetings and even didn't attend coding meetings even if they promised to join. This lead our project to slow down to a halt for testers.

Second worst was the unexpected difficulty to reach certain personnel via mail or Skype, opposite to their promises in meetings. This is overlapping with previous point, but very big issue in the project. We also had problems with communication in a sense that sometimes it was really hard to understand what another person was meaning.

Tertiary worst problem was our implementers inability to provide working code for all. This prevented testers to actually test the application during most of the project time.

It is very hard to say what we could have done better, and how this could be prevented in the future. Especially in student projects replacing personnel from the project is not viable option, unlike in business world. We did not have capable personnel to take their roles in full, unless we had replaced implementers with managers. Then managers would have had done all coding instead of managing the project itself.

For client point of view we could also have had provided better visibility of what is happening. It would have also helped project management. Even people promised to use Redmine issues for hour reporting, and work followup, it was impossible to actually have most of the project personnel use Redmine, even some managers declined to use it. We ended up having one system, namely Redmine, for wiki, documents, task- and hour tracking that was working only partially. But it was better than having three legged donkey, than pulling half beaten mule after it as well.

We weren't expecting to have problems with a server in the project. At the beginning only one of eight team members was supporting the idea of having a server to show our web app. Our client started asking about it at the end of October. Then we started to get it slowly. Then came christmas holidays, end of period test season, and then the only manager actually hoping to have the server had to go for paternity leave. After the leave we started to panic a bit, because the server issue wasn't getting any progress. Then we asked for different sort of server from a different provider, and when we sent the mails back and forth, we finally ended up in a hurry, few days before the deadline, still not getting through to database server or having working credentials for document server. The lesson learned here is this: if you aren't sure you want it, but it might be nice, and it doesn't cost very much, **get it early**!

Realized risks

1. Inexperience in project management

This risk could have been mitigated partially, if we had had any training in the beginning of the project. Some previous experiences and knowledge was passed down in the project, but there was not very much proactivity on that sector to gain more information.

2. Unfamiliar technologies

This was quite well mitigated with studies. It did take a lot of the implementation budget, but at least the learning responsibility from the course did fulfill.

3. Lack of communication

This risk realized towards managers. There was little to none from developers to managers, and often problems were realized after deadline was passed with halted work. This was problematic, as managers had hard time getting through to developers in the first place to ask about the status often to actually mitigate the risk itself. In a way the risk was realized by a separate risk.

Statistics

Team size	Dev. model	Start date	End data	Days	Hours
4+5	Modified scrum	14.9.2012	15.3.2013	126	1484

Table 1: General project information.

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	314,2	19,7	151,25	516,6	84,55	47,4	41,6	123,95	184,75	**1484**

Table 2: Group effort by activity.

Number of requirements	Pages	Use-cases	UI screens	Database diagrams	Database tables
35	1(wiki)	5	40	2	4?

Table 3: Requirements and high-level design outcomes

Pages	Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
4	1	1	0	0	1

Table 4: Design outcomes.

• Documen Documen	Pages	Versions
Preliminary analysis	8	3
Project Plan	19	8
Usability analysis	0	0
Requirements specification	1	9 (wiki edits)
Design plan	0	0
User interface document	43	5 or more.
Test plan	18	Google docs.
Test report	26	Google docs.
Usability test report	0	0
Final report	14	2
Project's story	6	2
Weekly reports	25	1
User's guide	3	Goole docs.

Table 5: Documents.

Language	PHP	HTML5	CSS	JS	MySQL
LOC	2619		86	n/a	n/a
Classes	46		4	n/a	n/a
Reused code			49	all	n/a
Revisions	68	68	68	n/a	n/a

Table 6: Codelines.

ViRPRO

Overview

Terms like virtual environments and gesture recognition are usually associated with modern video games, but the technology has promise in many other areas as well. Many TV productions today are filmed in a so-called virtual studio, where the environment is digitally generated by a computer. The VirPro project is an attempt to integrate a virtual environment with gesture recognition and lighting control in a way that has not yet been done in Finnish TV productions. The central idea is that the user could trigger changes in the virtual environment and/or studio lighting by using motion gestures instead of external devices like push buttons to accomplish the same effect.

VirPro is a piece of software that uses existing technology by TAUCHI as well as the team's own code to accomplish the task described above. The software includes its own virtual environment generated by JMonkeyEngine (a free open source 3D engine), but towards the end of the project the focus quickly shifted to implementing the needed functionality for the software to work in Yle's virtual studio in Tohloppi. The virtual environment in Yle's virtual studio is generated by a commercial engine called Vizrt.

The VirPro software uses Microsoft Kinect devices for gesture recognition, and it is capable of controlling lights that support the DMX512 protocol.

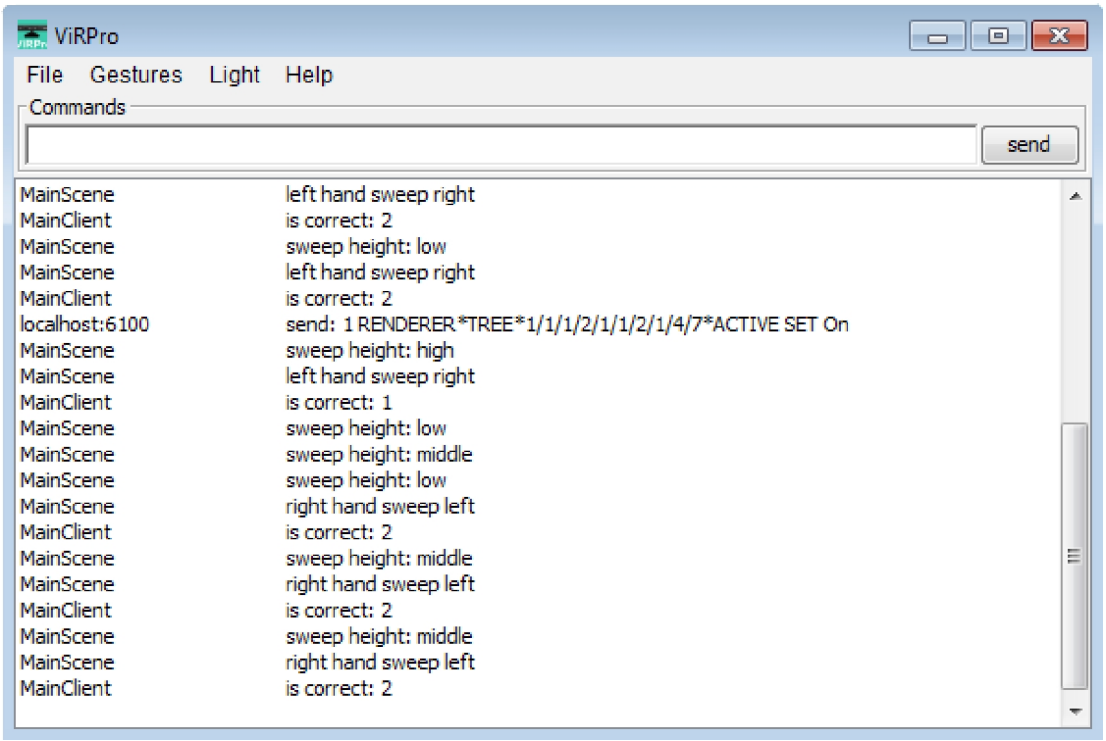


Figure 1: Main window view. Console log prominent.

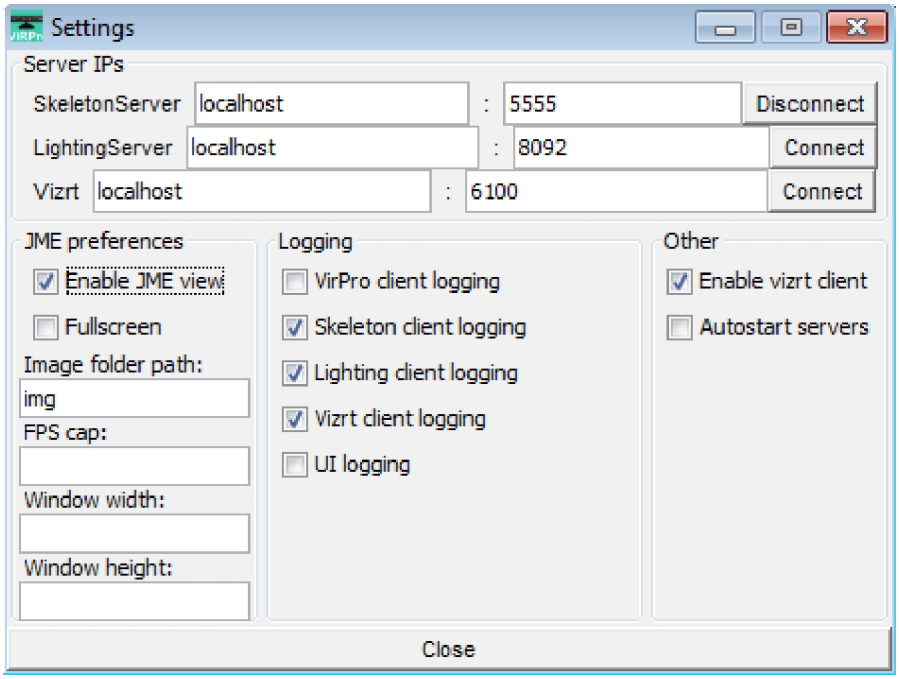


Figure 2: Settings view.

Organization and management

In the beginning the project team consisted of 7 members - 3 managers and 4 developers. However, quite early in the project one developer decided to call it quits as he lacked sufficient time and motivation to actively participate in the project. The project then continued with the remaining 6 members until the end.

The project team included the following personnel: Aku Häsänen (technical director), Mikko Myllylä (usability expert) and Timi Vienola (scrum master) as project managers and Kimmo Hämäläinen, Tuomas Laitinen and Antti Nykänen as developers. Almost all of the programming work was done by Antti and Kimmo. Tuomas was assigned to other tasks, like testing and helping the managers with the course documents.

The roles in the project were assigned on the basis of the members' own interests and competence in specific areas. This approach worked quite well, although at times some members were burdened more than others. However, the team generally had a high level of motivation, and the members had no problem investing the long hours needed to accomplish the tasks at hand.

Figure 3: The VirPro team and Yle's representatives; Ilmari Huttu-Hiltunen and Riku Karvakuono.

Methods and tools

Devices:

- Microsoft Kinects
- DMX512-protocol controlled RGB lights, moving and stationary

None of the team members had any previous experience working with Kinects, but that didn't turn out to be a problem. The team gained some good experience in developing software for the device, and encountered no significant obstacles in getting the device to function as intended. The DMX512 protocol also turned out to be quite understandable, and no problems relevant to our team's goals were observed.

Project management tools:

- Google Drive (working hour reporting and document writing)
- Redmine (programming tasks and requirements management)
- Wiki (document and general information archive)
- Doodle (scheduling meetings)

All of the tools used for project management worked quite well. We had no problems worth mentioning with any of the tools and we were successful in using them in an appropriate manner.

Development tools:

- Eclipse IDE (development environment)

- NetBeans IDE (development environment)
- JMonkeyEngine 3.0 (3D engine)
- TortoiseSVN (version control)
- Java 6 (programming language)

Most of the development work was done using the Eclipse IDE, as both Antti and Kimmo were more familiar with it. However, NetBeans was also used for some amount of work. Both performed reasonably well, and developing with different IDEs didn't pose a problem - technical or otherwise. We needed an open source 3D engine (preferably in Java) for our demos, and we decided to choose JMonkeyEngine 3.0. The engine turned out to be a good choice as it performed quite well in the rudimentary tasks we wanted to accomplish. Many of the members had previously used SVN for version control, and TortoiseSVN proved to be an easy to use GUI for that.

Communication tools:

- IRC (daily communication)
- E-mail (reminders)

Most of the communication outside of meetings was via IRC. This worked quite well, since almost all of the members had previous experience in using it, or were using it already for communication not related to this project.

Project phases and development model

Our project used a slightly modified version of the Scrum development model. We had (generally) two-week development sprints, each of which ended in a sprint review with the client. We held weekly face-to-face meetings in addition to discussing things in IRC whenever necessary.

Our project's sprints and milestones with dates are described in the table below.

Sprint	**Milestone**	**Date**
Sprint 0		
	Short project presentation	19.9.2012
	Preliminary analysis	27.9.2012
Sprint I	Development version 0.1	5.10.2012 - 19.10.2012
	Project plan	10.10.2012
	Project plan inspection	19.10.2012
	Sprint I review	19.10.2012
Sprint II	Development version 0.2	19.10.2012 - 2.11.2012
	Sprint II review	2.11.2012
Sprint III	Development version 0.3	2.11.2012 - 16.11.2012
	Personal Report I	1.11.2012 - 7.11.2012
	Sprint III review	16.11.2012
Sprint IV	Development version 0.4	16.11.2012 - 30.11.2012

	Midterm presentation	28.11.2012
	Sprint IV review	30.11.2012
Sprint V	Development version 0.5	30.11.2012 - 14.12.2012
	Sprint V review	14.12.2012
Sprint VI	Development version 0.6	14.12.2012 - 11.1.2013
	Sprint VI review	11.1.2013
Sprint VII	Development version 0.7	11.1.2013 - 25.1.2013
	Sprint VII review	25.1.2013
Sprint VIII	Development version 0.8	25.1.2013 - 8.2.2013
	Sprint VIII review	8.2.2013
Test sprint	Development version 1.0	8.2.2013 - 8.3.2013
	Test plan	15.2.2013
Wrap up		8.3.2013 - 15.3.2013
	Test report	10.3.2013
	Final report	12.3.2013
	Project story	12.3.2013
	Final presentation	13.3.2013
	Final meeting	15.3.2013
	Personal report III	15.3.2013-

Table 1: Project sprints and milestones.

Experiences

All in all the project turned out to be a success. From a difficult starting point, we started to build a piece of software that would perform a number of tasks in the direction of the quite ambiguous assignment we were given. Fortunately, as the project progressed our assignment became clearer and the agile development model ensured that a minimal amount of work was done in vain. The project's goals shifted somewhat towards the end, but the team nevertheless managed to make the necessary adjustments to satisfy the client's goals.

The project turned out to be quite challenging for both the developers and the managers - mainly because of the unclear requirements. Much of the early work was based on the team's own ideas and basic building blocks that could be safely assumed to be useful in the future, regardless of the more specific future goals.

In addition to the problems already mentioned, we also lost one developer and another developer lacked the necessary programming skills and interest to participate in the programming work. These problems were resolved by making the necessary changes to the project schedule and trying to keep up a high level of motivation and team spirit.

Statistics

Team size	Dev. model	Start date	End date	Days	Hours
3+4	Scrum	17.9.2012	15.3.2013	180	1076,5

Table 1: General project information.

Activity	Planning and management	Req. specification.	De-sign	Code	Integration and testing	Reviews	Repair	Study	Other	Total
Hours	344	8,5	11	253,5	79,5	65	4,5	134,5	176	1076,5
%	32	1	1	24	7	6	0,5	12,5	16	100

Table 2: Group effort by activity.

Figure 4: Working hours by week.

Number of requirements	UI screens
37	7

Table 3: Requirements and high-level design outcomes.

Overview diagrams	Class diagrams	Sequence diagrams	State diagrams	Other diagrams
4	0	0	0	0

Table 4: Design outcomes.

Document	Pages	Versions
Preliminary analysis	7	1
Project Plan	20	4

Test plan	5	2
Test report	6	1
Final report	21	1
Project's story	8	1
Weekly reports	25	-
Memos	28	-

Table 5: Documents.

Language	Java
LOC	8850
SLOC-P	6285
SLOC-L	4772
Code revisions	125

Table 6: Codelines.