Muksis project

Project plan

Richard Domander
Tuomas Mäenpää
Teemu Nisu
Tommi Teistelä

Version: 1.0
Public
27th October 2008

University of Jyväskylä

Department of Mathematical Information Technology

Jyväskylä

Approver	Date	Signature	Name verification
Project manager	__.__.2008		
Customer	__.__.2008		
Supervisor	__.__.2008		

Document information

Authors:

- Richard Domander (RD) dimadoma@jyu.fi 050-3482668
- Tuomas Mäenpää (TM) tutamaen@jyu.fi 040-7600465
- Teemu Nisu (TN) tejonisu@jyu.fi 040-8349310
- Tommi Teistelä (TT) totateis@jyu.fi 045-6528709

Document title: Muksis project, Project plan
Pages: 23
Sourece file: project_plan-1.0.tex

Abstract: This is the project plan document of software project Muksis. The document introduces readers to the area of this project by explaining terms and giving background information, as well as describing goals, available resources and management methods. In the later part of the plan information about tasks, schedule and risks are given.
Keywords: Software project, black frame detection, DVB, MPEG, search in MPEG TS, MPlayer.

Version history

Version	Date	Changes	Authors
0.1	19.9.2008	First version. Introduction and contact information started.	TM
0.2	28.9.2008	Introduction written. Terms, background, project goals and resources started.	TM, RD, TT
0.3	29.9.2008	Resources written.	RD
0.4	3.10.2008	Management methods started. Tasks, workload, division of labour, project goals and risks written.	TM, TT, TN
0.5	8.10.2008	Management methods written. Schedule started.	TM
0.6	10.10.2008	Corrections made for chapters.	TM
0.7	15.10.2008	Corrections for Risks chapter.	TN
0.8	22.10.2008	Corrections for project plan found during the document inspection.	TM
1.0	27.10.2008	Minor updates. Document finalized.	TM

Project information

Project Muksis designs and implements new features, like black frame detection, support for DVB subtitles, and search in MPEG TS, to an open source media player application MPlayer for Matthieu Weber. There is also an option to improve similar existing features to meet the requirements.

Authors:

- Richard Domander (RD) dimadoma@jyu.fi 050-3482668
- Tuomas Mäenpää (TM) tutamaen@jyu.fi 040-7600465
- Teemu Nisu (TN) tejonisu@jyu.fi 040-8349310
- Tommi Teistelä (TT) totateis@jyu.fi 045-6528709

Customer:

- Matthieu Weber mweber@mit.jyu.fi 014-2603056

Supervisors:

- Ville Isomöttönen vilisom@cc.jyu.fi 014-2604976

Contact information:

- Mailing lists: muksis@korppi.jyu.fi
- Archives: https://korppi.jyu.fi/kotka/servlet/list-archive/muksis/
- Room: AgC 225.3 / 014-2604971

Contents

1 Introduction

Muksis is a student software project in the Department of Mathematical Information Technology at the University of Jyväskylä. During the fall 2008 and the beginning of the year 2009 the project designs and implements new features to an open source media player application MPlayer. The main new features are black frame detection, support for DVB subtitles, and seek function to skip commercials based on black frame locations. The project team has an option to improve similar existing features possibly found in original MPlayer code or patches written by project client. The software is made for Matthieu Weber who is a senior assistant in the Department of Mathematical Information Technology at the University of Jyväskylä.

MPlayer is a free open source media player distributed under the GNU General Public License. It is a command line application with different optional graphical user interfaces. The program supports all major operating systems, including Linux and other Unix-like systems, Microsoft Windows, and Mac OS X. MPlayer has support for a variety of media formats. In addition to its generous range of supported formats MPlayer can also save all streamed content to a file.

The project will be performed in a team of four students: Richard Domander, Tuomas Mäenpää, Teemu Nisu and Tommi Teistelä. All members except Tuomas Mäenpää have a good knowledge of programming language C. Tommi Teistelä is also familiar with the MPlayer. Tuomas Mäenpää has mostly done web based software development before the project.

This document is an introduction to the work flow of the project Muksis. In Chapter 2 the main terms used in the document are explained. Chapter 3 expounds background information and chapter 4 outlines the main goals. Chapter 5 analyzes available resources, chapter 6 explains the process model and schedule, and chapter 7 describes project management methods,introduces tasks, workload, division of labour, etc. The risks involved in the project are finally described in chapter 8.

2 Terms

Following terms appear in this document:

Agile software development	a software development process. Agile methods emphasize real-time communication and working software as the primary measure of progress using iterative development.
Software project	course at The Department of Mathematical Information Technology.
C	a general-purpose programming language.
Demuxing	means the same as demultiplexing, the opposite of multiplexing.
DVB	Digital Video Broadcasting, a set of open standards for digital television. Defines various details about the physical and data link layer-level transmission of data, refers to existing MPEG standards for the actual format specifications where possible. The data stream itself is an MPEG-2 Transport Stream with some DVB-specific constraints and may contain multiple channels.
IDE	integrated development environment is an application that provides tools for software development.
Iterative development	technique of developing and delivering incremental components of business functionality. A single iteration results in one or more bite-sized but complete packages of project work that can perform some tangible business function. Multiple iterations recurse to create a fully integrated product.
MPEG	The Motion Picture Experts Group, a working group of ISO/IEC. Also a common name for certain standards created by them.

MPEG-2	The MPEG standard specifying video, audio and related format specifications, primarily used for DVDs and digital television broadcasting.
MPEG TS	is a MPEG transport stream. It is a communication protocol for audio, video and data. It's goal is to allow multiplexing of digital video and audio and to synchronize the output.
Multiplexing	a process where multiple digital streams are combined into one stream over a shared medium.
patch	is a small piece of software designed to fix problems with or update a computer program. This includes fixing bugs, replacing or adding features and improving the usability or performance.
OSS	open source software, a computer software, which source code is made available under a copyright license or arrangement. This permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form.
Subversion	is a free version control system. It is used to maintain current and historical versions of files such as source code.
SVN	is an abbreviation for Subversion.

3 Background

The MPlayer project was started in 2000 by a group of programmers from Hungary who were unsatisfied with the video players available for Linux at the time. They were soon joined by others and the open source project has been (slowly) advancing towards a 1.0 release since then. The player is written in the C language and is noted for its command line interface and support for a large number of different media formats and video output methods. A media encoding application called MEncoder is built on the same codebase and can encode video and audio into many different formats. [1]

Finnish television broadcasting migrated to the DVB - Digital Video Broadcasting standard in 2007. The current MPlayer code does not have support for its "soft" subtitling (subtitles which are separate from the video images) feature, which is used by some Finnish broadcasters, notably the national broadcasting company YLE.

Patches that add the primary features was previously created by Matthieu Weber, but the continued development of MPlayer has left these incompatible with the MPlayer source code. The features can thus be implemented either from scratch or by adapting the previous patches' code to work in the current MPlayer source.

It is a known fact that MPlayer's documentation and coding is not done in the best way. This adds more challenge for the team to understand the structure and functions of the application. Research in general is one major ensemble that will take most of the time in the first iterations of the project.

There are several versions of the MPlayer code available. The team decided to use the stable v1.0rc2 version of the source code instead of the SVN version, since the code might change too much during the project causing severe impacts to the project schedule.

4 Project goals

4.1 Software

The project's primary goal is to produce a patch for the latest stable version of MPlayer (MPlayer-1.0rc2) that adds various features:

- Support for DVB subtitling, at least for the type used by YLE. Other similar formats may be considered additional goals if time can be spared to implement them.
- A black frame detection filter that can be used to identify potential commercial breaks in a TV broadcast by counting the number of frames considered black (actual zero luminance is uncommon in MPEG video due to the ITU-R Recommendation BT.601, which effectively limits 8-bit luminance values to a range of [16,235] for compatibility with legacy equipment). The MPEG timestamps of the detected commercial breaks will be written to a file.
- Improved accuracy for seeking in a stored MPEG Transport Stream, the media container format used for digital television broadcasting. This will require changes to the MPlayer MPEG TS demuxing code. With some additional code and the output of the black frame detection filter, this can be used to make MPlayer automatically (and accurately) skip commercial breaks in a TV recording.

Some of the features may be of interest to the MPlayer community; creating and submitting a patch that adds these to the current SVN version of MPlayer could be considered an additional goal.

4.2 Documentation

Besides the software itself, the project will produce a number of documents related to its development. All documents will be written in English, except the ones created for the related course, which will be in Finnish. After the project these will be collected in a folder for reference in the university's future software projects. These include (in chronological order):

- **GPL agreement** contract on licensing the source code and the software under General Public License.
- **Project plan**, which introduces the project, its background and goals, and describes its overall management; the schedule, the tasks, how they are divided and what risks the project may face.
- **Application report**, which explains the overall structure of the MPlayer software, or at least the parts related to the project's goals.
- **Test plan & report** shortly outlines the testing strategy, the test cases, and report on the test results.
- **Project report** will be written at the end of the project to analyze and summarize the project progress.
- **Project presentation reports** (esittelyraportit) for the related course.
- **Meeting agendas, minutes** and related materials.
- **Project's e-mail list archive** in printed form.
- **Source code** a print out of the code written by project team.

The software patches and the documentation are also written in a CD, which is delivered to the university and to the client. One additional CD is filed in the department's archive.

4.3 Learning goals

The project's participants will gain useful experience in teamwork and working with an open source project and its codebase and (hopefully) some communication skills. There are various aspects in managing a software project and every team members should have a chance to learn about them.

Richard wants to refresh his knowledge of C, and learn to understand and work with a large existing (and in this case, challenging) code base. He is also interested to see a process in action in a real project. Richard believes that he will learn to understands why processes and the tasks related to them are so useful.

Teemu is interested in open source software, and wants to learn more about participating in an OSS project. He is also looking forward trying out agile software development methods in practice and getting experience of software projects in general. He is quite familiar with object oriented programming, and is excited about confronting older programming paradigms involved in the project.

Tommi is interested in open source development and wants to learn more about working with code originally written by others.

Tuomas has interests in project management and wants to learn about the management methods. He is also looking forward trying out agile software development methods and to understand programming language C better.

5 Resources

This chapter describes the available resources for project Muksis.

5.1 Recource constraints

One ECTS credit (op, opintopiste) equals about 27 hours of work [2], so a student can expect to work between 270 and 400 hours for the project (10-15 op for the course). The team members don't have many other courses simultaneously, so it won't be a challenge for anyone to allocate the required time. If the allocated time is not enough the features will be at least partially implemented, because of the iterative process used in this project.

5.2 The team

The team consists of four mathematical information technology majors. The supervisors tried to form as heterogeneous a team as possible from the course applicants. Still, all members meet the requirements set for the course, and this fall all selected were quite advanced in their studies.

5.3 Physical resources

The project team has been given the room AgC225.3 inside the project space AgC223.1. In the room there are four computers (1/member), office supplies and white boards. The team can also use the project space, but a reservation has to be made beforehand. The printer in project space and MIT department's copier can be used for project documents. Also a small "library" of technical books is available, but most of the technical documents needed is provided by the client, Matthieu Weber.

The team can request further software installations and resources (for example, Trac) from MIT's PC support. Team members are not allowed to install programs. The team chose Linux and Code::Blocks IDE, but kept one Windows machine just in case. The team can also get a projector and a digital audio recorder for the meetings.

5.4 Supervisors

The supervisor in charge is Ville Isomöttönen from MIT. He supervises all aspects of the project and especially its planning, process, management, and general state. He also acts as a sort of mediator between the client and the team. There currently is no technical supervisor for the project, but everyone in the team has some experience with C. Linux OS and Code::Blocks IDE are familiar too for most of the members, so they can instruct those less versed. Weber has specific technical knowledge about the subject, so the team will cope even if no technical supervisor is found, although lack of one presents unique risks.

6 Process and schedule

This chapter explains the process and schedule of project Muksis.

6.1 Process model

The project is carried out using an agile process model that uses multiple sequential two-week iterations. The beginning phase is called 0-iteration, in which the topic of the project, workspace and personnel are introduced. 0-iteration acts as a start-up and orientation phase with no software implementation. After this, iterations are performed in two-week cycles.

Each iteration aims to build further the product made in previous iteration. This ensures that the project is progressing steadily, and participants get to monitor the results as soon as possible. Short iteration intervals also help reacting faster to the changes and better predict the output.

6.2 Schedule

Well planned schedule is essential for a software project. Because of the agile process model, a timetable is planned for every iteration. At first only an overview for each iteration is written in the start-up phase. The final schedule for the next iteration is planned at the end of each two-week cycle.

6.2.1 Iteration schedules

The release plan is presented in Figure 6.1 and explains the major goals for each iteration. The first day of new iteration is the next day after the official meeting between iterations. The project encompasses a 2-week 0-iteration and five iterations, which is 12 weeks in total.

6.2.2 Iteration steps

All implementations are decided iteration-by-iteration. In the meeting at the end of the iteration project team reviews the priorities of the tasks and requirements with the customer and decides which tasks the team does next. After the meeting, in the internal meeting of project team, the persons in charge are nominated for the tasks and time usage estimated. Previous schedules can be used to more precisely evaluate this. During the iteration the software is produced with a typical design-implement-test sequence.

Iteration	Duration	Interval	Most important tasks
0	2 weeks	17.9.2008 - 1.10.2008	Introduction to the subject, research and beginning of the project plan.
1	2 weeks	2.10.2008 - 15.10.2008	Black frame detection's implementation, completing project plan, research of seek function and dividing it's implementation to tasks.
2	2 weeks	16.10.2008 - 29.10.2008	Seek function's implementation and starting to implement DVB subtitles, writing architecture plan. DVB subtitle implementation and testing.
3	2 weeks	30.10.2008 - 12.11.2008	More testing and writing test plan & report and application report.
4	2 weeks	13.11.2008 - 26.11.2008	Writing project report. Decide whether or not to send the patch to MPlayer.
5	2 weeks	27.11.2008 - 10.12.2008	Ending project.

Table 6.1: Release plan of the iterations.

7 Management methods

This chapter describes management methods of project Muksis.

7.1 Meetings

An official meeting is held between every iteration with all stakeholders of project Muksis. Before the meeting the project team plans and prepares the material, which includes a proposal on the next iteration's requirements and explains how working time was distributed in the previous iteration. In the beginning of a meeting the previous iteration is reviewed, then the goals for the future iteration are decided. The goals mean the parts of software to be implemented and the documents to be written during the next two weeks. The chairman of the meeting brings and sets up the needed equipment to the meeting room and opens the session. The secretary prepares the agenda of the meeting and takes notes during the session. After the meeting he writes a report called minutes, which reviews the main items discussed and any decisions that were made. All material is to be delivered two work days before the meeting, the agenda one day before and the minutes at maximum three days after the meeting.

7.2 Communication

Communication is held up with e-mail list `muksis@korppi.jyu.fi`. Unofficial meetings are held at least once per week if there are no official meetings. Project supervisors have also delivered the contact information to every participant. Most of the team's internal communication is maintained with informal meetings at the team's office. An e-mail list `muksismafia.group@korppi.jyu.fi` is also established and used for internal communication. The project manager has the responsibility to act as a mediator between the team and the rest of the participants and to monitor communication's sufficiency.

7.3 Task management

The example of the task proposal the team presents for the meeting is given in Table 7.1. After meetings, the team plans together how the selected part of the software will be implemented by dividing the work to tasks using the office room's white boards. These tasks might already be presented in the meeting, when there is only a need to write them down to the white board. The description of the task is written to one column in the white board and the estimated time for accomplishing the task to another. The names of team members who will participate the task are also written to the board. The percentage estimation of task progress is written to the last column during the iteration.

7.4 Time logging

Time usage of every iteration is written into an Excel document. All work hours of the team members are written to the Excel table by sorting them using task identification. The time used in the peripheral course is also written down to the same Excel table. In the beginning of the iteration, the project team evaluates the estimated time needed to accomplish each task. This means that the team must have internal meetings during this phase to divide tasks and to estimate the time requirement. In the end of the iteration the time log shows how well the plan held during the iteration, and this helps to compose even more realistic approximations during the project.

7.5 Responsibilities

To make the project easier to accomplish all work is managed with a set of responsibilities. A team member is named for each responsibility. This is presented in Table 7.2. The Peripheral course's lectures, group works and presentations are handled as a separate responsibility belonging to all team members.

A responsibility means that a team member is selected to be a person in charge for the area of responsibility. The person in charge monitors the progression of the area but doesn't necessarily complete the selected tasks by himself. The responsibilities will be taken into account when the tasks are assigned to the team members.

7.6 Documentation

Documents are typeset with LaTeX 2_ε typesetting software using department's document templates and named with format `document_name-version.pdf`. All documents are published in PDF format and stored in the public project space created on the network of the University of Jyväskylä, and are available for all. At the end of the project these files will be burned on CD-R media and placed into the project folder. In addition, meeting agendas and minutes are stored in the project management application named Trac, where they are easily editable and accessible to all participants. All complete plans and reports are signed by the customer, the project manager and the supervisor.

7.7 Version management

The team uses Subversion for version management. A branch is created for every main feature to be sure code tweaking doesn't affect other members implementing different features. Every official document has a version number is named as `document_name-a.b`, where a is the major number of version with only values of 0 or 1. b is the minor number of version and is an integer starting from 1. For example after version number 0.9 comes 0.10 not 1.0. The number 1.0 indicates that the document is complete.

Task	Richard	Teemu	Tommi	Tuomas	Summary (h)
Project management	**9**	**9**	**9**	**17**	**44**
Informing	2	2	2	2	8
Project plan	2	2	2	6	12
Project planning	1	1	1	4	7
Schedule managing	1	1	1	2	5
SVN	2	2	2	2	8
Trac & WWW	1	1	1	1	4
Meetings	**4**	**4**	**4**	**7**	**19**
Agenda & preparation	1	1	1	1	4
Meetings	3	3	3	3	12
Minutes	0	0	3	0	3
Research	**9**	**9**	**9**	**14**	**41**
Functions of old MPlayer	2	2	2	2	8
MPlayer code & specs	5	5	5	5	20
Running TV records with MPlayer	0	0	0	5	5
Research patch	2	2	2	2	8
Peripheral course	**6**	**6**	**6**	**6**	**24**
Lectures	5	5	5	5	20
Group work	1	1	1	1	4
Implementation	**17**	**17**	**17**	**4**	**55**
Changing existing bf code	2	2	15	2	21
Trying Matthieu's bf code	15	15	2	2	34
Total hours	**45**	**45**	**45**	**48**	**183**

Table 7.1: Task division and time estimation for iteration 1.

Area of responsibility	Description	Person in charge
Project management	Monitoring the project. Planning and scheduling tasks. Keeping communication up with the stakeholders.	Tuomas
GPL agreement	Preparing the General Public License.	Richard
Project plan	Writing most of the project plan. Writing corrections.	Tuomas
Application report	Writing the application report.	Richard
Test plan & report	Planning the test cases and the report with the help of other members.	Teemu
Project report	Writing the project report with the help of other members.	Tuomas
Project presentation reports	Planning the presentations with the help of other members.	Tommi
Meeting agendas, minutes	Printing corrected agendas and minutes into the project folders.	Tuomas
Email list archive	Printing and placing Email list into project folders.	Tommi
Source code	Printing source code made during the project into the project folders.	Tommi
Research	Monitoring that every member understands researched topics.	Tommi
WWW	Adding documents and files to network drive and updating the main HTML-page.	Richard
DVB	Understanding how DVB subtitles work and integrating the results to the final product.	Richard
BFD	Understanding how black frame detection works and integrating the results to the final product.	Teemu
MPEG TS	Understanding how seek works and integrating the results to the final product.	Tommi
documenting code	Documenting the code written during the project.	Teemu
uniforming code	Fixing the code to be mostly uniform.	Richard
finalising code	Formatting the code so it is similar as in the MPlayer	Tommi
testing	Running video files and testing the features.	Tuomas
Trac	Formatting the wiki pages so that they are mostly uniform.	Tuomas
Project folder	Preparing and delivering the project folders.	Tuomas
SVN	Setting up the Subversion. Ensuring the files are committed.	Tuomas

Table 7.2: Areas of responsibilities.

8 Risks

Many things can prevent a software project from succeeding. This section is about evaluating risks related to this project and considering how to avoid them. Risks are defined in Table 8.1. The probabilities and effects of each risk are evaluated on a scale of low-intermediate-high.

Risk	**Probability**	**Effect**
Client's availability	Low	High
Supervision	Intermediate	Intermediate
Inability to participate	Low	High
Human relationships	Low	High
Inexperience	High	Intermediate
Programming skills	Low	High
Scheduling	Intermediate	Intermediate
Communication	Intermediate	High
Language	Intermediate	Low
MPlayer development	Low	Intermediate
Applications	High	Intermediate

Table 8.1: Risks of the project.

8.1 Client's availability

The resulting software will be mostly for personal use, so the client has a great importance for the project. In this case, the client is also the participant with the most knowledge about the technical details of the subject. For these reasons the progress of the project may slow down if the team is not able to contact the client at some point.

Anticipating: Meeting times should be settled early enough.

Tracking: The team should be in contact with the client frequently. Tracking the response times helps the team to keep up with the situation.

Reacting: If the client is prevented from participating to the supervision of the

project, the team should try to proceed on their own on the basis of what has been agreed previously.

8.2 Supervision

The supervisor might not always be able to give instructions in short-term. Also the subject is very technical, and a technical supervisor has not been found.

Anticipating: Meeting times should be settled early enough. Each member is responsible for actively contacting the supervisor if the need arises.

Tracking: Keep contact with the supervisor by e-mail or telephone.

Reacting: Ask the project supervisor to find someone to brief the team if technical difficulties arise.

8.3 Inability to participate

Obstacles such as vacations, social matters or illness may turn up to prevent some team members from participating in the project.

Anticipating: Inform the other team members in advance about vacations. Share tasks equally within team members.

Tracking: Communication between team members is essential for everyone to know the current situation. If someone doesn't show up for an appointment, the others should contact him.

Reacting: The other members try to complete the tasks of the missing participant. Unexpected absence shall be reported immediately.

8.4 Human relationships

Effective teamwork is not possible if the group members don't get along with each other for some reason.

Anticipating: The group members should get to know each other in the beginning of the project. Share tasks equally within team members.

Tracking: Follow the situation and bring up the subject in a group meeting to supervisor if it looks like there is an issue.

Reacting: Try to resolve conflicts by talking. Ask supervisor to participate in the discussion.

8.5 Inexperience

The team members don't have much experience of software projects of this magnitude. Most of the team members are also unfamiliar with the subject.

Anticipating: Survey the factors causing uncertainty when sharing tasks. One should inform the others if he cannot carry out an assignment by himself.

Tracking: Have situation reviews frequently.

Reacting: Ask for help from other team members, the supervisor(s), and the client.

8.6 Programming skills

There may be insufficiencies in the team's programming skills. Dealing with MPlayer's source code may be challenging because the team is unfamiliar with the programming paradigms it uses.

Anticipating: Survey the team's programming skills in the beginning of the project.

Tracking: Keep watch on the progress of programming by having inspections and situation reviews.

Reacting: Ask for briefing from other team members or supervisors if problems arise. Use pair programming to share knowledge.

8.7 Scheduling

The project has only a few main requirements. Estimating the amount of time required to implement them might still be quite difficult. Underestimation is likely to lead to too strict schedule.

Anticipating: Estimate time requirements for each task individually. Use iterative process to make it easier to adopt changes in the schedule. Inform the client within an iteration if delays are likely to occur.

Tracking: Time consumption is monitored with the time management application and on the whiteboard. The amount of time actually used should be compared with the estimates at the end of an iteration.

Reacting: Negotiate with the client about rearranging the schedule and leaving objectives out if necessary.

8.8 Communication

Communication between the participants of the project may be insufficient. This can be a serious problem because the project involves a great amount of background research.

Anticipating: The participants should get to know each other so they would feel more comfortable while communicating. Adopt minimum practices for communication (see 7.2).

Tracking: The project manager follows the situation and makes sure that the group communicates sufficiently.

Reacting: In serious lack of communication, contact the supervisors for a solution.

8.9 Language

The native language of the team members is Finnish. The language used for documentation and communicating with the client is English. This may cause some misunderstandings and make writing the documents more challenging for the team.

Anticipating: Survey the team's skills in English language in the beginning of the project. The students should get familiar with the subjects that are about to be discussed and study the most common terms in advance.

Tracking: Constantly track if the foreign language causes issues such as misunderstandings or delays that might compromise the progression of the project.

Reacting: If misunderstandings occur, correct them as soon as possible. Take up the question with the supervisor and the client if the language seriously holds back the progress.

8.10 MPlayer development

Radical changes in MPlayer source code could make applying the patches to latest SVN version of MPlayer more complicated. This would make delivering the patch to MPlayer community significantly harder, since they only accept patches compatible with the latest version. There is also a chance that other developers implement similar functionalities to MPlayer, which would make this project obsolete.

Anticipating: Research the existing code to see what has been done already.

Tracking: Keep watch on the situation with MPlayer development by tracking changes in the SVN version and new releases.

Reacting: Negotiate with the client about project requirements if it looks like something has already been implemented by other developers.

8.11 Applications

A lot of software applications are needed during the project. Some of them may not be already installed on the computers in the project room. The team members may also be unfamiliar with some of the applications. When it comes to a computer software, there is always a risk of malfunction too.

Anticipating: Determine in the beginning what kind of software is required to complete the project.

Tracking: Make sure that all the required applications are installed and working properly.

Reacting: Request the technical support to install missing software and to resolve other software related problems. Remember to check that the software requested actually wokrs. Also inform the others via project mailing list to keep them up with the situation. Ask for help with unfamiliar applications from other team members and the supervisor.

9 Bibliography

[1] MPlayer documentation; Appendix D. History, referenced 10.10.2008
http://www.mplayerhq.hu/DOCS/HTML/en/history.html

[2] University of Jyväskylä; KIEPO termipankki, referenced 29.9.2008
http://www.jyu.fi/hum/laitokset/solki/tutkimus/projektit/kiepo/termipankki/opintopiste/?searchterm=opintopiste