Muksis project

Testing plan

Richard Domander
Tuomas Mäenpää
Teemu Nisu
Tommi Teistelä

Version: 0.3
Public
November 25, 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, Testing plan
Pages: 11
Sourece file: `testing_plan-0.3.tex`

Abstract: Description of the testing strategy and methods used by the Muksis software project. Includes test cases and information about sample data.
Keywords: Software project, black frame detection, DVB, MPEG, search in MPEG TS, MPlayer.

Version history

Version	Date	Changes	Authors
0.1	7.11.2008	First version	TN
0.2	10.11.2008	Testing enviroment	TM
0.3	19.11.2008	Reporting chapter and corrections	TN

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.

Working software is the most important goal of the project. To achieve this, extensive testing is required. This document describes the testing strategy and methods used to verify the functionality of the software.

The resulting software consists of multiple components, which are added to MPlayer. Each component has specific requirements for testing. Special occasions as well as general functionality of the component should be taken into account. Test cases are used for testing the functionalities. The cases are listed in chapter 5. The software works correctly when it produces satisfactory results from all cases. The results will be reported in testing report.

Because changes in MPlayer's source code are required, one major goal for testing is to confirm that the original functionalities of MPlayer remain intact. Beta testing will be used in attempt to eliminate any unforeseen side effects.

Terms used in this document are listed in chapter 2. Chapter 3 is about testing environments. Chapter 4 briefly describes the methods used for testing. Test cases are listed in chapter 5 and the sample material for testing in chapter 6.

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 Testing environments

The implemented features were tested with the computers in the project room C225.3 by project members and with the Leffakone enviroment by the client. Some other enviroments were also used.

The environment used in the room C225.3:

- Four computers with AMD Athlon 64 Processor 3500+, 2 GB of RAM and NVIDIA Geforce 7300 graphics cards.
- Three of the computers had Fedora core 8, kernel 2.6.25.14-69.fc8 and one had Microsoft Windows XP Professional version: 5.1.2600, Service Pack: 2.0.
- Four 19" Samsung SyncMaster 940B monitors.
- MPlayer version 1.0rc1 and the latest version from MPlayer SVN.

The Leffakone enviroment:

- Ahtlon XP 1800+ with 256 MB RAM and 120 GB HDD with the graphics Matrox G400.
- Debian 3.0 Woody, with an home-made patch to XFree86.
- Matrox G400 graphics card, analog TV card Hauppauge WinTV Go and Hauppauge WinTV Nova T digital TV card.
- Regular TV, connected to the VGA connector.
- MPlayer version 1.0rc1.

Other enviroments:

- Ubuntu 8.04 running under VMware Player.
- Ubuntu 7.10 running under Athlon 1Ghz, 512 MB RAM.

4 Testing methods

This chapter is about the methods used for testing the application. Since the nature of the project differs quite a lot from a typical software project, some methods must be adapted to fit the needs of the project.

4.1 Module testing

The purpose of module testing is to verify the functionality of each component developed. Since the components are basically just enchancements to MPlayer, the components cannot be tested without being integrated to MPlayer first. This is why module testing must be done in parallel with integration testing. Test cases for testing the components are described in chapter 5.

4.2 Integration testing

The components developed during the project are quite individual and don't integrate to each other. They must however be integrated to MPlayer. The purpose of integration testing is to verify that MPlayer works properly with the new features.

4.3 Beta testing

The group will deliver a test version of the software to Matthieu Weber for beta testing it with his "Leffakone" which is practically the target system for the project. At this point the testing will expand beyond the sample recordings described in chapter 6. Any issues that haven't emerged with the sample files will hopefully come up during the beta phase.

5 Test cases

Test cases are used to confirm that software works and fullfills the requirements. Test cases can be applied at all phases of testing and the tests will be done with both MPlayer-1.0rc1 and the latest SVN version of MPlayer.

5.1 Black frame detection filter

5.1.1 Detecting commercial breaks and writing EDL files

Description: The black frame detection filter must be able to correctly detect commercial breaks from a video stream and write their starting and ending points to an EDL file. Approximate times of commercial breaks in samples are listed in chapter 6.

Samples for testing: 6.1.1 __ 6.1.2 __

Result:

1. Advertisements were marked correctly. __ __

5.1.2 Handling PTS reset

Description: The EDL files created by the black frame detection filter must be correct even if there is a PTS reset in a recording.

Samples for testing: No testing material available at the time of writing this document.

Result:

1. Advertisements were marked correctly despite the PTS reset. __ __

5.2 MPEG TS seek

5.2.1 Accuracy

Description: To fulfill the requirements, MPEG TS seek must end up within one second (± 0.5s) of it's target unless the target is unreachable (beyond EOF for example).

Samples for testing: 6.1.1 __ and 6.1.2 __ for skipping commercials with EDL files. 6.2.1 __, 6.2.3 __ and 6.2.5 __ for testing general functionality by jumping back and forth with the arrow keys.

Result:

1. MPEG TS seek is accurate enough. __ __

5.2.2 Stress test

Description: Even multiple seeks within a short period of time must not cause MPlayer to fail. This can be tested with any of the samples by jumping backwards and forwards repeatedly.

Samples for testing: 6.1.1 __ 6.1.2 __ 6.2.1 __ 6.2.3 __

Result:

1. Repetitive seeking does not cause crashing or freezing. __ __

5.2.3 Special occasions

Description: MPEG TS seek must be able to handle the following occasions:

- Jump to the beginning when trying to seek backwards over the beginning of the recording.
- Jump to the end when trying to seek over the end of the recording.
- Seek accurately over a PTS reset.
- Seek accurately over starting and ending points of an EDL record.

Samples for testing: 6.1.3 __ for seeking over PTS reset. 6.1.1 __ and 6.1.2 __ for seeking around EDL skips. 6.1.1 __ and 6.2.3 __ for testing the other occasions.

Result:

1.	Beginning of file	__	__
2.	End of file	__	__
3.	PTS reset	__	__
4.	EDL skip	__	__

5.3 DVB subtitles

Description: Subtitles should be visible on all samples listed below.

Samples for testing: 6.2.1 __ 6.2.2 __ 6.2.3 __ 6.2.4 __ 6.2.5 __

Result:

1.	General functionality	__	__
2.	Visual quality	__	__
3.	Timing	__	__

6 Sample recordings

Features are tested with several sample files recorded from the Finnish digital television. Samples described in section 6.1 are recorded from MTV3 channel and they are meant for testing black frame detection and MPEG TS seek. Samples for testing DVB subtitles are recorded from YLE channels and they are listed in section 6.2.

6.1 Commercial skipping

Both black frame detection filter and accurate MPEG TS seek are required for successful commercial skipping. The black frame detection filter must be able to mark the starting and ending points of a commercial break correctly into an EDL file. Accurate seek is needed for the actual skip.

6.1.1 House.mpeg

A recording from MTV3. Includes multiple commercial breaks. Approximate times (PTS values) of the commercial breaks are listed below.

Commercials:

1. Start: 8372 End: 8409
2. Start: 8595 End: 8869
3. Start: 9726 End: 9965
4. Start: 10857 End: 11085
5. Start: 11815 End: 12009

6.1.2 Rakkauden_anatomia.mpeg

A recording from MTV3. Includes multiple commercial breaks. The recording stops during commercials so the last skip should jump to the end of file. Approximate times (PTS values) of the commercial breaks are listed below.

Commercials:

1. Start: 59270 End: 59599

2. Start: 60252 End: 60564
3. Start: 61749 End: 62059
4. Start: 62695 End: 62725
5. Start: 62889 End: 63149

6.1.3 jumppa.mpeg

This file does not include any commercials but has a PTS reset, which the seek function must be able to handle.

6.2 DVB subtitles

6.2.1 FST-fin.mpeg

A recording from YLE FST channel with Finnish subtitles.

6.2.2 FST-swe.mpeg

A recording from YLE FST channel with Swedish subtitles.

6.2.3 dvb-TV1-subtitles.mpeg

A recording from YLE TV1 channel with Finnish subtitles.

6.2.4 dvb-TV2-subtitles.mpeg

A recording from YLE TV2 channel with Finnish subtitles.

6.2.5 medico_de_familia.mpeg

A recording from YLE Teema channel with Finnish subtitles.

7 Reporting

The testing report will consist of the test cases presented in chapter 5 and error reports. The results of test cases shall be reported by marking an 'X' on the left-hand line when succeeded and on the right-hand line in a case of failure. Sample files used for testing can be marked by putting an X next to the reference of a sample in the section that says "Samples for testing:".

Errors should be reported in detail to a report template like below. If the recording the error occurred with is not described in this document, delivering the file to the project team might help them to solve the problem.

Test case: ___ Sample file: _______________
Error description:
____
____
____
____
____
Tested by: ________________
Fixed: __.__.200_, _________