User Requirements Document
OpenLogbook

OpenLogbook devteam
developer@openlogbook.org

15th May 2003

Contents

Version	Date	Author	Description
1.0	28.10.2002	Mikko	First release version
1.1	10.11.2002	Mikko	Removed chapter about abbreviations
1.2	21.11.2002	Mikko	Changed priority for use case 201 to 307
1.3	27.11.2002	Mikko	Added a use case (206) for modifying existing hot spots
1.4	2.1.2003	Mikko	Removed use cases for activity view
1.5	3.2.2003	Mikko	Returned the use case for activity view plus added a new use case for text comment logging (203)

Table 1: Changelog

More detailed history log can be found from WebCVS of the project [4].

1 Introduction

1.1 Introduction to OpenLogbook concept

An important aspect of performing a physics experiment that is taking place over an extended period of time is logging it, that is, making accurate documentation of the data itself, and the circumstances under which they were recorded. Certain kinds of experiments take hours of preparation and idle monitoring of the experiment in the laboratory, and when something important happens everything can be over in a matter of seconds. At that moment one frantically tries to write down relevant data, usually on a notebook or something similar. This means that lots of data is lost because of inadequate logging capabilities. Also some kinds of data, such as video and audio are not logged at all.

OpenLogbook is a software for reviewing different kinds of experiments. With this application a scientist can go through an experiment and review all data related to it, e.g. video streams, audio streams and log files describing e.g. equipment parameters. It also enables the scientist to search through the data for a specific event using logged parameters, as the amount of data produced from one experiment can be very large.

1.2 About this document

The purpose of this document is to define what OpenLogbook system is doing from user's point of view. This document is considered as an agreement about the essence of the future OpenLogbook software and describes what it should be like and what it should do.

At the end of project, the success of the project and it's product is validated by the requirements specified in this requirement documentation. Test plan is used to verify that existing functions meet the requirements and are working as required. These tests are built based on the requirements defined in this document.

This document can be updated during the project based on available project resources, changes in customer needs or in the order of importance of requirements.

1.3 References

Existing OpenLogbook concept documentation was used for gathering requirements. These documents can be found from project's homepage [4].

2 System overview

OpenLogbook is a java-application which is capable of handling video, audio and text files, and which provides an easy-to-use interface to them. OpenLogbook itself will not be used to record all data from an experiment. The program does not care how the video and audio streams are produced, as long as they are in a format that it can handle. After recording the data, the video streams may then be edited using a video-editing program. When the user is finished editing the video he will save it in a format that is understood by OpenLogbook. OpenLogbook is used to catalog, review and search through the data afterwards. OpenLogbook per se provides two different views to the data, the timeline view and the activity view.

The software can also be used to record hot spots, i.e. time stamps of important events. These time stamps can be used to instantly jump into a specific point in time with all data synchronised to that point in time.

OpenLogbook uses XML to describe the contents of an experiment. This ensures flexibility and compatibility with other possible products that could use experiment data produced by OpenLogbook.

2.1 Use case overview

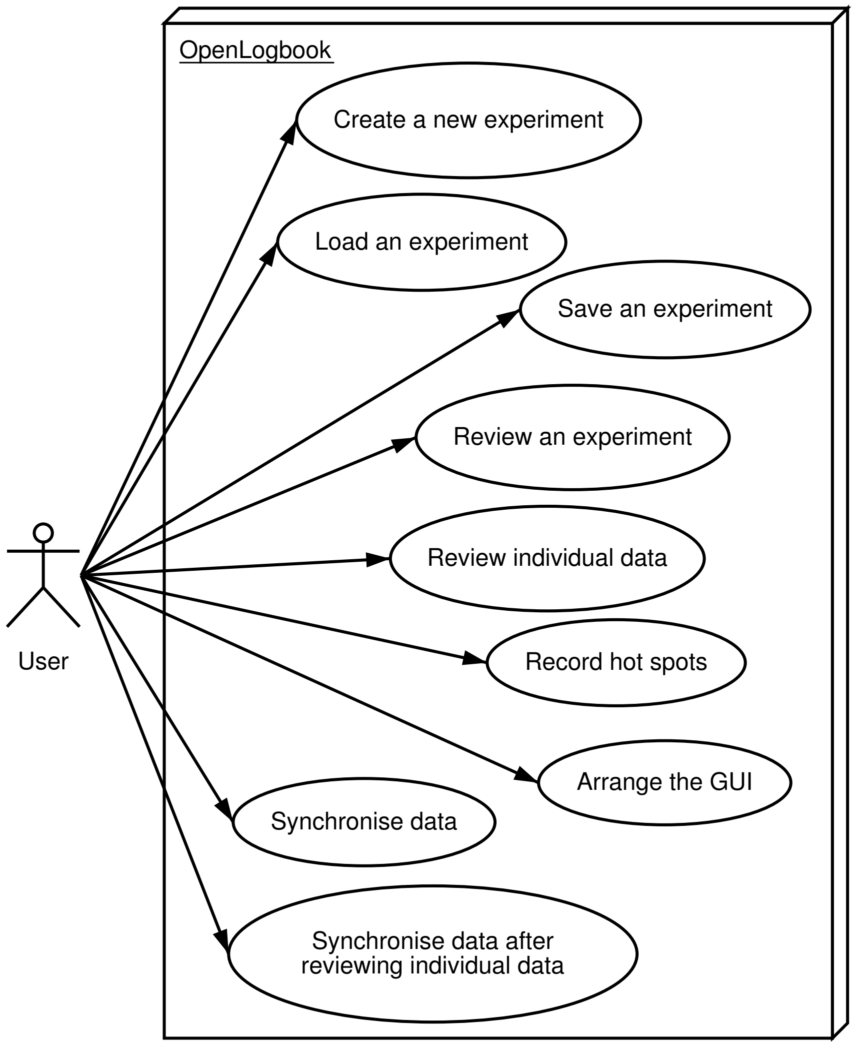

Figure 1: Use cases of mandatory requirements

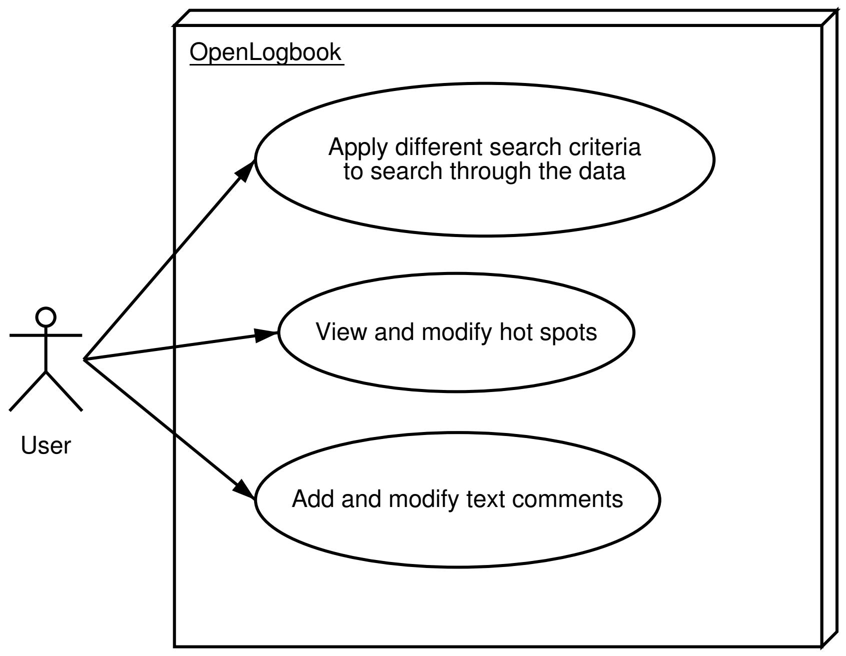
OpenLogbook
Apply different search criteria to search through the data
View and modify hot spots
Add and modify text comments
User

Figure 2: Use cases of useful requirements

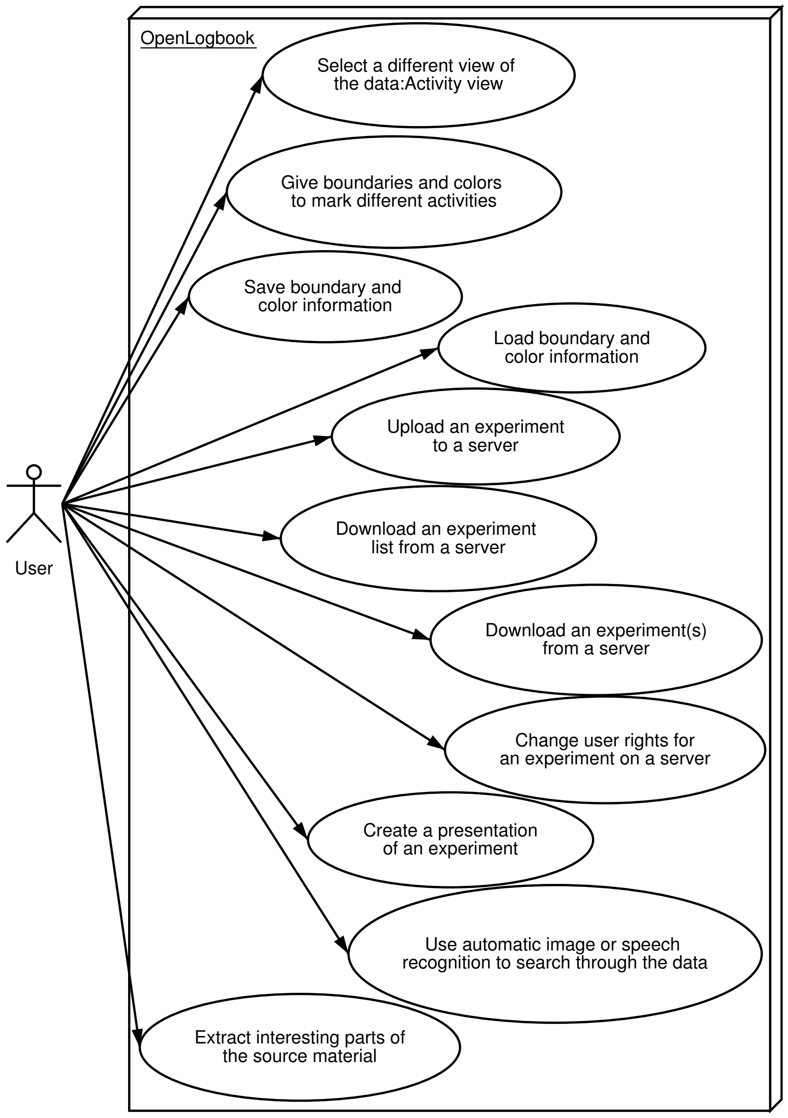
OpenLogbook
Select a different view of the data:Activity view
Give boundaries and colors to mark different activities
Save boundary and color information
Load boundary and color information
Upload an experiment to a server
Download an experiment list from a server
Download an experiment(s) from a server
Change user rights for an experiment on a server
Create a presentation of an experiment
Use automatic image or speech recognition to search through the data
Extract interesting parts of the source material
User

Figure 3: Use cases of supplementary requirements

3 Business goals

There has been discussion at the MIC (Mikroelektronik centre) in Denmark about improving the ways of logging nanoscale experiments because of reasons mentioned in section 1.1. Lots of data is lost because of inadequate logging capabilities, and a new way of logging and reviewing data is needed. OpenLogbook is part of the OpenLab-concept [3].

4 User groups

4.1 Scientists

This is the most obvious user group of OpenLogbook. They use the OpenLogbook to review and analyse experiments. They can also share their experiments with other scientists all over the world by uploading them to a central server. Initially the amount of users is rather small, as the software is first tested at MIC.

4.2 Teachers and students

OpenLogbook can be used to visualise physics experiments. Instead of having to use slide shows and trying to explain in words what happened in an experiment, a lecturer could simply show the audience the experiment itself, using material such as video clips of a real experiment. Number of users can be large if e.g. all students attending a lecture are counted.

5 Functional requirements

Following requirements were gathered from the intended end-users at MIC last summer. Scientists were interviewed and actual laboratory experiments were observed to identify the needs and different possible use cases of the software. These requirements were iterated and shown to many people to have a consensus on what the software should do.

Requirements are categorised with three different priorities:

- Mandatory
  These requirements have to be implemented.
- Useful
  These requirements are not required for the software to work, but would be useful.
- Supplementary
  These requirements are implemented if there is sufficient time. They are taken into account in the design of the software, so that they can be implemented without architectural changes.

Every requirement has its own code, which consists of three numbers. First one is the category, 1 for mandatory requirements, 2 for useful requirements and 3 for supplementary requirements. Next two numbers specify the number of the requirement.

5.1 Use cases

5.1.1 Create a new experiment

Code: 101
Summary: A user creates a new experiment consisting of different types of data.
Actors: User
Preconditions: Basic sequence: User selects the action from the menu and selects the data files that are included in the experiment. A new experiment is initialised.
Exceptions:
Post conditions: User can either choose to save the new experiment, create synchronisation information for it or discard it.

5.1.2 Load an experiment

Code: 102
Summary: User loads an existing experiment dataset.
Actors: User
Preconditions: An experiment data set exists.
Basic sequence: User selects the action from the menu. He then selects the experiment datafile from a file selector window. All data items are shown to the user in the basic view, i.e. the timeline view.
Exceptions: Experiment file is corrupted. Insufficient disk space for untarring the experiment data set.
Post conditions: User has all data items displayed, and he can start reviewing them simply by pressing the start button from the timeline window.

5.1.3 Save an experiment

Code: 103
Summary: User saves an experiment dataset
Actors: User
Preconditions: User has modified the experiment.
Basic sequence: User selects the action from the menu.
Exceptions: User doesn't have write rights to the target directory. There is not enough disk space left on disk.
Post conditions: The experiment is stored.

5.1.4 Review an experiment

Code: 104
Summary: User reviews an experiment by playing the associated data in a synchronised manner.
Actors: User
Preconditions: User has opened an existing experiment.
Basic sequence: User first opens an experiment. He then starts reviewing the experiment. The timeline shows all hot spots, so that a user can easily select a hot spot and all data that is visible is instantly moved to that point in time. User can use the controllers provided by the timeline window to start, stop and play the experiment.
Exceptions: Computer is not fast enough to play all data items. Parts of data have been

corrupted.
Post conditions:

5.1.5 Review individual data

Code: 105
Summary: User chooses a single data item and reviews that. Every data item has its own time slider and buttons for play, stop and pause.
Actors: User
Preconditions: User has opened an experiment
Basic sequence: User presses button in timeline view to free data from synchronization and uses the individual controllers which every data item has to play and review a single data item.
Exceptions:
Post conditions: Data synchronisation is lost.

5.1.6 Record hot spots

Code: 106
Summary: When recording an experiment, a user creates hot spots consisting of a time stamp and additional comments about the hotspot.
Actors: User
Preconditions: User has opened a new experiment.
Basic sequence: User starts the hot spot recorder from a menu. Whenever he wants to record a hot spot, he presses space. The system records the time and opens a text box for inserting additional comments related to the hot spot.
Exceptions:
Post conditions: User saves the experiment.

5.1.7 Arrange the GUI in timeline view

Code: 107
Summary: User arranges data items in the GUI. Each data item that is shown has its own window, its size depending on the data. Every window has a time bar and control buttons, e.g. play, stop, pause. Each window can be individually moved around and hidden from the working area. Each window has also two buttons in its upper right corner for quickly hiding or removing it.
Actors: User
Preconditions: User has opened an existing experiment.
Basic sequence: User can move, hide or remove individual windows.
Exceptions:
Post conditions:

5.1.8 Synchronise data

Code: 108
Summary: User synchronises the data for simultaneous playing.
Actors: User
Preconditions: User has opened existing experiment or created a new one.
Basic sequence: User selects each data item and synchronises them related to the experiment timeline.

Exceptions:
Post conditions: Experiment data is synchronised.

5.1.9 Synchronise data after reviewing individual data

Code: 109
Summary: User loads previously created synchronization data.
Actors: User
Preconditions: Previously created synchronisation data exists and at least one data file is out of this synchronisation.
Basic sequence: User presses button in timeline view that reloads the synchronization.
Exceptions: Synchronisation data is corrupted.
Post conditions: Experiment data is resynchronised.

5.1.10 Apply different search criteria to search through the data

Code: 201
Summary: User searches through the data using different criteria based e.g. on the experiment parameters.
Actors: User
Preconditions: User has opened an existing experiment.
Basic sequence: User selects the action. He gives desired parameter values when prompted.
Exceptions: Parameters are not in the given range.
Post conditions: User is shown data which matches his search criteria.

5.1.11 View and modify hot spots

Code: 202
Summary: User views recorded hot spots and modifies them.
Actors: User
Preconditions: User has opened an existing experiment, which has recorded hot spots.
Basic sequence: User first selects the action from the menu. He is presented a list of recorded hot spots, and by clicking a hot spot he can see the comments related to it. User can also change existing comments.
Exceptions:
Post conditions: User can see all hot spots and comments to them.

5.1.12 Add and modify text comments

Code: 203
Summary: User adds, modifies or deletes text comments to the project.
Actors: User
Preconditions: User has created a new project.
Basic sequence: User first selects the action from the menu. A window is opened with existing comments visible. User adds new comments by inserting text into a text field and pressing enter. User can modify an existing comment by clicking on it and modifying the text. User can delete comments by clicking on them and pressing a delete-button. User saves the log file.
Exceptions:
Post conditions: User has an XML file of all the comments.

5.1.13 Select a different view of the data: activity view

Code: 301
Summary: User chooses another view to the data. Basic view is the timeline view. User can also choose to review the experiment in an activity view, where the experiment is divided into color bars marking different activities.
Actors: User
Preconditions: User has opened an existing experiment.
Basic sequence: User selects another view from the menu. The system changes into the other view, displaying data in a new window. In activity view user can see different state of activies from color bars and make conclusions of interesting parts of experiment.
Exceptions:
Post conditions: Data is now displayed in a new kind of window.

5.1.14 Give boundaries and colors to mark different activities in activity view

Code: 302
Summary: User can give number boundaries to mark up alteration in numerical log files.
Actors: User
Preconditions: User has opened an existing experiment in activity view.
Basic sequence: User selects the action. User gives numerical boundaries and colors that correspond to them.
Exceptions: User gives boundaries or colors which cannot be applied, e.g. are out of range.
Post conditions: User has created boundary and color information for activity view.

5.1.15 Save boundary and color information

Code: 303
Summary: User saves boundary and color information.
Actors: User
Preconditions: User has created boundary and corresponding color information in the activity view.
Basic sequence: User selects the action to save boundary and color information.
Exceptions:
Post conditions: Boundary and corresponding color information have been saved.

5.1.16 Load boundary and color information

Code: 304
Summary: User loads boundary and color information.
Actors: User
Preconditions: User has opened an existing experiment in activity view and a file including boundary and color information exists.
Basic sequence: User selects the action to load boundary and color information.
Exceptions:
Post conditions: Boundary and corresponding color information have been loaded and can bee seen on activity view.

5.1.17 Upload an experiment to a server

Code: 305
Summary: User uploads an experiment to a central server after saving it into a file
Actors: User, server
Preconditions: A server that is running OpenLogbook server software is used.
Basic sequence: User first saves an experiment and then uploads it to a server by selecting the action from the menu, and giving additional information when prompted.
Exceptions: Server is not running. User doesn't have sufficient user rights to upload an experiment. Server is out of disk space.
Post conditions: A copy of the experiment exists on the server.

5.1.18 Download an experiment list from a server

Code: 306
Summary: User downloads a list of all experiments that exist on a server. This list has information about the experiments, e.g. name, author, access requirements, size etc.
Actors: User, server
Preconditions: A server is running
Basic sequence: User first connects to the server by selecting the action from the menu. Then he downloads the list by selecting the corresponding action.
Exceptions:
Post conditions:

5.1.19 Download an experiment(s) from a server

Code: 307
Summary: User downloads an experiment data file from a server for reviewal.
Actors: User, server
Preconditions: A server is running, and the user has sufficient access rights to the experiment in question.
Basic sequence: User first connects to the server by selecting the action from the menu. Then he downloads the experiment by choosing the corresponding action, first giving his username and password to verify his rights to download the experiment.
Exceptions: User doesn't have sufficient access rights to the experiment. User doesn't have sufficient disk space.
Post conditions: User has a local copy of the experiment on his hard disk.

5.1.20 Change user rights for an experiment on a server

Code: 308
Summary: User changes user rights of an experiment existing on a server.
Actors: User, server
Preconditions: An experiment exists on a server.
Basic sequence: User logs into a server, and selects the appropriate action.
Exceptions: User doesn't have rights to change access rights.
Post conditions: The experiment has new user rights.

5.1.21 Create a presentation of an experiment

Code: 309
Summary: User creates a multimedia presentation of an experiment
Actors: User
Preconditions: An experiment exists.
Basic sequence: User opens an experiment. He selects parts of the data to be included in the experiment.
Exceptions:
Post conditions: User has a multimedia presentation of the experiment.

5.1.22 Use automatic image or speech recognition to search through the data

Code: 310
Summary: User can search through the data using an existing image or a speech pattern.
Actors: User
Preconditions: User has a suitable image or a speech pattern available.
Basic sequence: User selects the action. He then selects the file from a menu, and starts the search.
Exceptions: The supplied image or speech pattern cannot be used.
Post conditions: User is shown the data which matches the given image or speech pattern

5.1.23 Extract interesting parts of the source material

Code: 311
Summary: User can select which parts of the material he wants to save and which to discard.
Actors: User
Preconditions:
Basic sequence: User selects the action. User goes through the data and marks the parts that user wants to save. When user is done he applies the changes.
Exceptions:
Post conditions: User has a modified experiment.

6 Non-functional requirements

6.1 Usability requirements

OpenLogbook is mainly intented for users with no extensive background in computer science. This presents big requirements for the simplicity of the user interface. If users want to benefit from the software, it must be easy to learn and use. Usefulness of the software depends heavily from the user interface.

User interface must be usable without need to first read through user manual. Heuristic analysis is used to evaluate the usability of the GUI. Detected weaknesses should be fixed before the final release of the software.

6.2 Performance requirements

Computers running OpenLogbook software must be fast enough to play multiple video streams with audio involved which makes some restrictions for underlying hardware. These hardware requirements are defined by Java Media Framework and OpenLogbook software can't do anything to correct this weakness.

Other parts of the software should be fast enough for comfortable usage. This means that no additional delays occurs while using menus or accessing other operations. This is evaluated by the client, but a good rule of thumb is that the user shouldn't have to wait for a response from the system for more than three seconds. This applies to response times from commands to the GUI, e.g. opening new windows. It doesn't apply to response times of functions that can take a long time, e.g. applying a search function on a large data set. In these cases the user should be presented with an estimation of the presumed completion time, e.g. using a progress bar.

6.3 Portability requirements

Because OpengLogbook is going to be Java based program it should take advantage of its properties and should therefore work at least in Windows and Linux environments.

6.4 Maintainability/quality requirements

OpenLogbook software will be developed further after the project as open source software. Because of the nature of this future development, special emphasis should be placed on the overall quality of the software. All source code has to be commented well and enough documentation has to exist about the architecture to enable further development. Using methods described in the Quality Manual [6] will ensure this.

6.5 Reliability requirements

System must be reliable during recording of the experiment hot spots so the information is not lost. Failures during playback are more acceptable but not desirable.

There are no requirements for hardware reliability meaning that duplicate systems are not needed. Responsibility of the backups is left to users or administrators. The same applies to OpenLogbook servers because they are not vital for usage of the OpenLogbook clients and therefore unavoidable downtime is acceptable.

7 Constraints

There is no special constraints besides those mentioned here. Recommendations for the project can be found from the Quality manual.

7.1 Software constraints

OpenLogbook is required to run on Windows and Linux operating systems. The programming is done using Java programming language which is designed to be independent from the underlying operating system. Development is done using Java version 1.4 meaning that the user must have Java 1.4 compatible Java Runtime Edition (JRE), too.

OpenLogbook uses JMF 2.1.1. to handle and present streaming media content. Therefore the JMF installation is required on client computer.

Resulting from the operating system independence of Java, OpenLogbook should run on any operating system supposed the system has Java 1.4 and JMF. It may even run on different types of hardware if the same conditions are met.

7.2 Hardware constraints

OpenLogbook runs on PC hardware. Due to the multimedia content the PC must have a sound card. It must also have enough CPU power to decode and play several video streams in real time. Normal Java environment should be enough for using OpenLogbook software. However, playing video files through Java Media Framework (JMF) requires the following minimum hardware:

- 200 MHz Pentium, 160 MHz PowerPC, or 166 MHz UltraSparc.
- At least 64 MB RAM
- Soundcard

There might be enormous needs for storage space depending on the experiment types stored. OpenLogbook software itself doesn't need over 10Mb of disk space, but for storing experiments at least 2-10Gb is recommended for each experiment.

8 Main domain concepts

- **Data** Data is recorded during experiments. It can include e.g. audio, video and text data.
- **Experiment** Experiment is a series of consecutive events performed in order to do research on a certain scientific theory/process.
- **Hot spot** Hot spots are important, user-defined events in an experiment. A hot spot consists of a time stamp of the event and additional comments.
- **Logging** Logging means gathering important information of an experiment. This data could be parameter values, microscope images, charts etc.
- **Open source software** Open source software basically means software which anyone can use and modify at will. A more complete definition is [7].
- **Views to data** These are different kinds of representations of the same data. Two have been specified so far; timeline view and activity view.

9 Acronyms and abbreviations

All related acronyms are explained in a separate terminology document [5]

References

[1] Course homepage
`http://www.soberit.hut.fi/T-76.115/`
Cited: 23.10.2002.

[2] Java Media Framework
`http://java.sun.com/products/java-media/jmf/index.html/`
Cited: 23.10.2002.

[3] OpenLab-concept
`http://wikihip.cern.ch/twiki/bin/view/Openlab/OpenLab`
Cited: 23.10.2002.

[4] OpenLogbook project webpages for the course T-76.115
`http://motha.tky.hut.fi/openlogbook/`
Cited: 28.10.2002.

[5] OpenLogbook Terminology document
`http://motha.tky.hut.fi/openlogbook/delivery/terminology_doc.pdf`
Cited: 26.11.2002.

[6] OpenLogbook: Quality Manual
`http://motha.tky.hut.fi/openlogbook/delivery/quality_manual.pdf`
Cited: 26.11.2002.

[7] Open Source Software definition by OSI
`http://www.opensource.org/docs/definition_plain.php`
Cited: 20.11.2002