Janne Pesonen

Optimization of the preprocessing steps of MEG measurement data in Meggie application

Master's Thesis in Information Technology

May 12, 2015

University of Jyväskylä

Department of Mathematical Information Technology

Author: Janne Pesonen

Contact information: `janne.o.pesonen@student.jyu.fi`

Supervisor: Tapani Ristaniemi

Title: Optimization of the preprocessing steps of MEG measurement data in Meggie application

Työn nimi: MEG-aineiston esikäsittelyvaiheiden optimointi Meggie-käyttöliittymässä

Project: Master's Thesis

Study line: Ohelmistotekniikka

Page count: 66+0

Abstract: This thesis examines the optimization of the preprocessing steps of a MEG (magnetoencephalography) measurement data preprocessing, analyzing, and vizualizing software Meggie, mostly by adding batch processing functionality. A comprehensive list of requirements helped with the development that covered the practical phase of the thesis. The development mostly consisted of refactoring the old code with minor user interface modifications and few new functionalities. The theoretical phase focuses on the architecture at the current state of the software, the memory management, and the requirements with some details about their implementation.

Keywords: batching, Canopy, computational neuroscience, ECG, EEG, EOG, GUI, MaxFilter, MEG, memory management, MNE-Python, MVC, PyQt4, refactoring, requirements engineering, software architecture

Suomenkielinen tiivistelmä: Tutkielma optimoi MEG (magnetoenkefalografia) mittausaineiston esikäsittelyyn, analysointiin ja visualisointiin kehitetyn Meggie-käyttöliittymän esikäsittelyvaiheita pääasiassa lisäämällä mahdollisuuden rinnakkaislaskentaan. Optimointi toimi tutkielman käytännön osuutena ja sen tukena oli kattava lista vaatimuksia. Optimoinnissa suurin osa työstä meni vanhan koodin refaktorointiin, mutta myös käyttöliittymäkomponentteihin tuli pieniä muutoksia ja itse ohjelmaan muutama uusi toiminnallisuus. Teoreettisessa osuudessa keskitytään Meggien nykyiseen tilaan tutkimalla sen arkkitehtuuria, minkä lisäksi tarkastellaan kehitysvaiheita vaatimuslistan pohjalta sekä tuodaan esiin muistinhallinnan ongelmia rinnakkaislaskennassa.

Avainsanat: Canopy, ECG, EEG, EOG, GUI, laskennallinen neurotiede, MaxFilter, MEG, MNE-Python, muistinhallinta, MVC, ohjelmistoarkkitehtuuri, PyQt4, refaktorointi, rinnakkaislaskenta, vaatimusmäärittely

Janne Pesonen

Glossary

Batch process	Process in which multiple files are processed without user intervention.
Computational neuroscience	An area of computer science that investigates the information originating from the human brain.
Cortex	The outermost layered structure of neural tissue in brain.
CSV	A file format including comma separated values. Widely supported by scientific applications.
Data averaging	The epoched data is averaged to preserve the oscillating elements of the brain while averaging out the irregular background noise.
Data epoching	Cutting time intervals from the measurement data.
ECG	Electrocardiogram; the recorded electrical activity of the heart.
EEG	Electroencephalography; a noninvasive method for recording electrical activity along the scalp.
Elekta	A global human care company.
Elekta Neuromag	A MEG device that measures MEG data from the brain. Basically a helmet that consists of hundreds of SQUIDs.
EMG	Electromyogram; the recorded electrical activity of the muscles.
EOG	Electro-oculogram; the recorded signal of the eye movements.
Evoked oscillation	Oscillation in the brain caused by a certain type of stimulus. Phase-locked in the signal as an experimental planning.
fMRI	Functional magnetic resonance imaging; investigates the brain activity by measuring the blood flow in the brain.
Induced oscillation	Oscillation in the brain that is not phase-locked in the signal.
MaxFilter	Software package that enables the preprocessing of MEG measurement data. Intended to be used with Elekta Neuromag products.
MEG	Magnetoencephalography; a noninvasive method for mapping

	brain activity.
MEG device	A helmet that consists of more than 300 SQUID sensors that are evenly located to cover all the areas of the cortex.
MNE	An academic software package for processing MEG and EEG (electroencephalography) data.
MVC	Model-view-controller architectural pattern.
Neuron	Processes and transmits electrical impulses to other neurons in the brain which generates the measurable magnetic field of the brain.
OS	Operating system.
Pass-band	A range of frequencies passed by a filter.
PCA	Principal component analysis; replaces the original variables by a smaller number of principal components in order to reduce the complexity of the data.
Preprocessing	Removal of disturbances from the measurement data.
PyQt4	A set of Python bindings for v4 and v5 of the Qt application framework from Digia.
Qt	A set of C++ libraries including platform independent abstractions for graphical user interfaces, networking, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, and user and application settings.
Raw	A measurement data file in FIF format.
SQUID	Superconducting Quantum Interference Device; a sensor that provides sufficient sensitivity for practical MEG work. The SQUID measures the cerebral magnetic field that is caused by brain activity.
SSP	Signal-space projection; Estimates an interference subspace and applies a linear projection operator to the MEG data to remove the physiological (EOG and ECG) interferences.
SSS	Signal-space separation; Removes the external artefacts by creating separate subspaces for the external and the internal sources

	to the sensor helmet. After the separation the external sources can be removed.
TFR	Time-frequency representation of a signal.
Terminal	A command-line interface that provides access to the operating system in Linux. Comparable to the Command Prompt in Windows.
UI	User interface; enables the user actions of a software.

List of Figures

Figure 1. A view of the main window after adding subjects to an existing experiment. 6
Figure 2. Creating an experiment and adding subject/subjects. 8
Figure 3. Preprocessing tab of the main window displaying the completed preprocessing steps. 10
Figure 4. Compute ECG projections dialog. 11
Figure 5. Epoching tab of the main window. 12
Figure 6. Epoch creation dialog. 13
Figure 7. Averaging tab of the main window. 13
Figure 8. RawFIFF.plot() function of MNE software package used on a measurement data before and after SSS calculation provided by MaxFilter. 18
Figure 9. Normalized data with the eigenvectors. 20
Figure 10. Process diagram displaying the preprocessing steps. 26
Figure 11. Process diagram displaying the epoching and averaging steps. 28
Figure 12. Sequence diagram of the use case ’Create a new experiment’. 29
Figure 13. Sequence diagram of the use case ’Add a subject’. 30
Figure 14. Sequence diagram of the use case ’EOG batch process’. 31
Figure 15. Activity diagram of Meggie. 34
Figure 16. Package diagram of Meggie. 36
Figure 17. Deployment diagram of Meggie. 39

Contents

1 INTRODUCTION 1
1.1 Research problem 1
1.2 Structure of the thesis 2

2 OVERVIEW 3
2.1 Background 3
2.2 History of Meggie 4
2.3 Overview and purpose of Meggie 4
2.4 Possibilities and Effects 5

3 THE USER INTERFACE OF MEGGIE 6
3.1 Creating an Experiment and Adding Subjects 7
3.2 Preprocessing Tab 7
3.2.1 ECG/EOG Dialog 8
3.3 Epoching Tab 9
3.3.1 Epoch Creation Dialog 9
3.4 Averaging Tab 9

4 MAGNETOENCEPHALOGRAPHY 14
4.1 MEG in Brief 14
4.2 Data Acquisition 15
4.3 Data Preprocessing 16
4.3.1 Signal-space Separation 17
4.3.2 Signal-space Projection 19
4.3.3 Principal Component Analysis 19
4.4 Data Epoching and Averaging 21
4.5 Spectral Analysis 21
4.6 MEG in Comparison to EEG and fMRI 22

5 METHODOLOGY 23
5.1 Theoretical Phase 23
5.2 Practical Phase 23

6 ARCHITECTURE 25
6.1 Logical View 25
6.2 Process View 32
6.3 Development View 35
6.4 Physical View 37
6.5 Scenarios 40
6.6 Architectural Changes 42

7 IMPLEMENTATION 43
7.1 Tools 43
7.1.1 MNE 43

7.1.2 MaxFilter 43
7.2 Test Data 44
7.3 Requirements 44
7.3.1 Batching 44
7.3.2 Preprocessing 47
7.3.3 Epoching and Averaging 48
7.3.4 Nonfunctional Requirements 48
7.4 Memory Management 49
7.4.1 Main Cases 49
7.4.2 Problems in Memory Management 50

8 CONCLUSIONS 52

BIBLIOGRAPHY 54

1 Introduction

Meggie is a software for preprocessing, analyzing, and vizualizing magnetoencephalography (MEG) measurement data. MEG is a method of measuring magnetic fields caused by electrical activity in the brain. By analyzing the data accumulated by the measurements, the type and location of the brain activity can be discerned.

Before the practical phase of this thesis, Meggie didn't include functionality for processing multiple subjects at the same time. Motivation for the thesis was to include this functionality to the preprocessing phase of Meggie, since that phase is somewhat mechanical compared to the later phases of MEG analysis. The problem was to elicit a comprehensive list of requirements to enable the needed functionalities.

Originally the application didn't have any kind of architectural design, which probably caused the biggest challenges to the development phase. It also lead to the theoretical phase of the thesis to evaluate the architecture of the application.

1.1 Research problem

The research problem leading to this thesis is how to implement the optimization of the preprocessing steps of MEG measurement data to an existing software application, Meggie, to speed up the MEG research and make the steps more intuitive. Also, reproducibility of the MEG research is essential. Like in all respectable software projects, the requirements elicitation is needed to ensure that the software works as it's supposed to. The final requirements list will be presented as well as short explanations for their implementation. The requirements are considered as the research questions to be answered (Hevner and Chatterjee 2010).

The main theoretical aspect focuses on the architecture of the software to possibly reveal some of the good or bad solutions during the implementation phase. Also, the memory management needs to be inspected since the files are large enough to exceed high amounts of memory, especially when batch processing several subjects at once.

1.2 Structure of the thesis

The main goal of the thesis is to enable batch processing to the preprocessing phase which is part of the optimization. For example, the batch processing allows the user to simultaneously remove interfering eye blinks from multiple MEG measurement data files with predefined parameters. The rest of the optimization focuses on making the existing preprocessing steps more intuitive and exact in the sense of producing relevant results. Theoretical phase of the thesis includes an evaluation of the architecture after the optimization phase, the memory management, the requirements for the optimization that can be stated as the research questions (Hevner and Chatterjee 2010), and how the requirements were met.

Chapter 2 gives an overview of Meggie. Chapter 3 goes through the basic functionalities of the application. Chapter 4 examines the methods Meggie uses to process MEG data, and provides general knowledge about MEG. Chapter 5 represents methods that were applied to the practical and theoretical phases. Chapter 6 introduces the architecture with pictures and their explanations. Chapter 7 focuses on the implementation phase, including the requirements, and the memory management problem.

2 Overview

This chapter gives an overview of the scope of Meggie and the reasons behind further development.

2.1 Background

Jyväskylä Centre for Interdisciplinary Brain Research was established in 2012 within the Faculty of Social Sciences. The Centre focuses on the longitudinal study and changes in a human brain caused by natural events, illnesses and treatments. The Centre holds a magnetoencephalography (MEG) device that measures the magnetic fields caused by electrical activity in brain. With the measurement data, the activities of a human brain can be researched.

At the moment, the researchers in the MEG field utilize a mix of closed and open source software, graphical user interfaces and command line scripts. However, their use is not very intuitive and present a steep learning curve for an uninitiated person, such as a student trying to learn the field. In addition, during the preprocessing and analyzing the researchers/students have to manually save and load multiple files.

The main goal was, therefore, to develop a graphical user interface prototype Meggie to integrate the functionality of the open source libraries and scripts. Where applicable, as in the case of MaxFilter (Elekta Neuromag, Stockholm, Sweden) (see 7.1.2), optional integration of closed source software was also taken into account. The prototype interface was designed to guide an inexperienced user, clearly denoting the most common sequences of methods and paths of analysis. At the same time it was designed to be flexible enough to allow the experienced user to perform advanced analysis without resorting to manual managing of parameters and files. (Aliranta and Pesonen 2013)

2.2 History of Meggie

In january 2013 Hoksotin project group was created during a software project course. It's a compulsory course in information technology for software engineering students in the University of Jyväskylä. The project was started off from the need to create an improved way of preprocessing, analyzing and visualizing magnetoencephalography (MEG) data (Aliranta and Pesonen 2013).

As it's expected of a software project, a lot of problems emerged during the project mostly because of the Hoksotin group members' lack of expertise in neuroscience and little experience in the tools used for the development. Therefore, Meggie needed further development right after the project ended in summer 2013. The further development process included three members of Hoksotin project group, of which mainly two were seriously involved, and lasted for three months. This development mostly focused on fixing the errors and adding some missing functionalities to the existing components. Even after the further development, a lot of work will be needed in correcting the old and adding the needed functionalities for researchers to be able to run a complete MEG analysis chain.

Including this thesis, there is also another Master's thesis involved in the further development of Meggie. The authors of these two theses were also members of Hoksotin project group and they applied to the software project course as student participants.

2.3 Overview and purpose of Meggie

Meggie is an application originally designed to aid MEG research at the University of Jyväskylä. The problem is currently that the research is executed in terminal utilizing several applications with manually writing the scripts to execute the needed steps. This makes it troublesome especially for the students when learning to do the research themselves. Also, a lot of extra work is required when writing the scripts and parameter values on command line.

Meggie improves the reproducibility of MEG research, facilitates the research by integrating the needed processes into one application, gives the user a visual approach to the research steps, passes the default parameters to the processes, and reduces the time consumed in the research.

Typical usage environment is a private computer in a researcher's office or laboratory. Supposedly, the operating system is Linux. A MEG research usually involves around tens of subjects which are preferably measured under similar conditions to create the MEG measurement data files to be processed. This ensures that all the measurement files can be processed using practically the same parameter values. During the analysis process, the user wants to save statistics, movies and pictures to support his/her research.

There exists several applications for MEG analysis, but some of them are commercial and offer functionalities that aren't essential for the research. Meggie is free, simple to use and designed to have enough options in the future for a comprehensive MEG research.

2.4 Possibilities and Effects

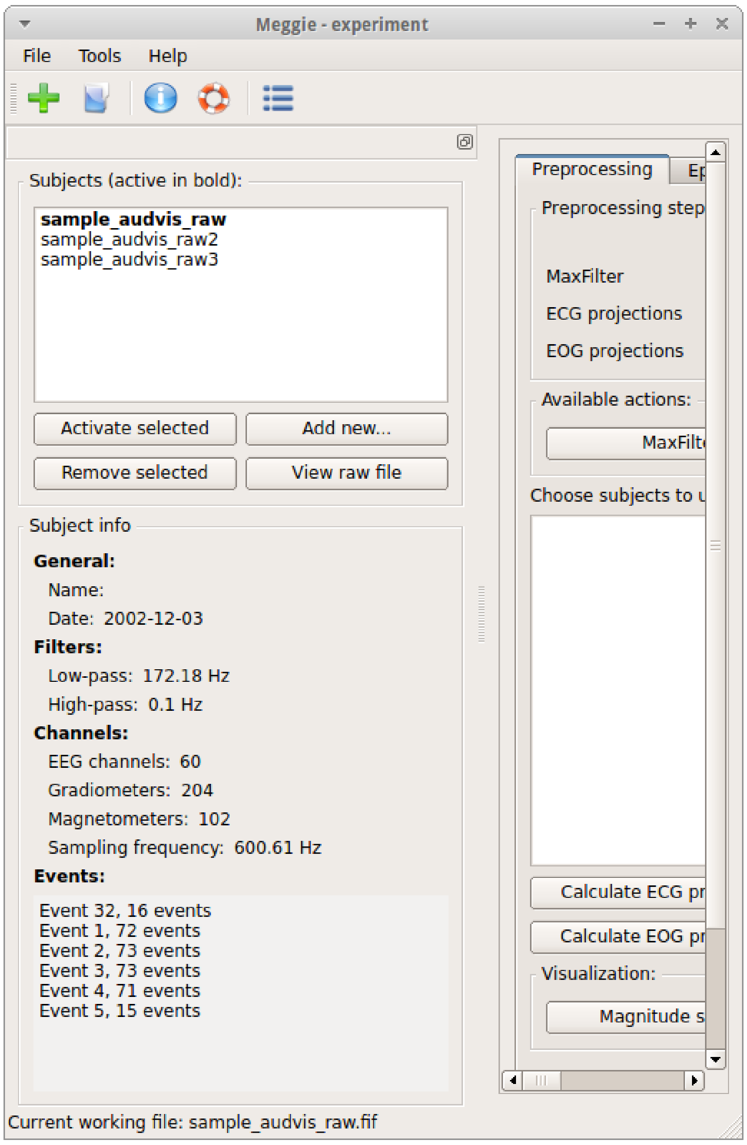
The overall goal of the system is to provide a visual and a simple way of conducting MEG research even for the amateurs with computer technology.

Meggie will have some general effect in computational neuroscience (Gewaltig and Cannon 2014) by providing means on integrating several important open source libraries and recommending an analysis pipeline for computing MEG measurement data. The development also adds an effort to testing those libraries and possibly finding weak solutions which gives us an opportunity to e-mail the developers these problems helping them to further develop their libraries.

Meggie has already been used in at least one course in the University of Jyväskylä and occasionally some students have been using it for their projects.

3 The User Interface of Meggie

The user interface of the application is based on a single persistent ’Meggie’ main window (fig. 1). Presently, the main window represents the preprocessing and analysis process of a single measurement file at a time. However, some of the steps are available for several measurement files at once, if there are two or more subjects in the representation, as shown in the figure 1. The representation is called an experiment. The various stages of the experiment are divided into tabs in the user interface. (Aliranta and Pesonen 2013)



sample_audvis_raw
sample_audvis_raw2
sample_audvis_raw3

Subject info

General:
Name:
Date: 2002-12-03
Filters:
Low-pass: 172.18 Hz
High-pass: 0.1 Hz
Channels:
EEG channels: 60
Gradiometers: 204
Magnetometers: 102
Sampling frequency: 600.61 Hz
Events:
Event 32, 16 events
Event 1, 72 events
Event 2, 73 events
Event 3, 73 events
Event 4, 71 events
Event 5, 15 events

Current working file: sample_audvis_raw.fif

Figure 1: A view of the main window after adding subjects to an existing experiment.

Initially, the main window doesn't have any active experiment or subjects. The tabs representing the stages are shown all the time, but the actions are enabled after an experiment is created and at least one subject is added to it and set as an active subject. The tabs allow various methods of preprocessing, analysing and visualization. (Aliranta and Pesonen 2013)

On the technical side, the application uses PyQt4 ("PyQt4 Reference Guide" 2014) as the user interface library, allowing the usage of most Qt widgets. Qt is composed of C++ libraries and development tools that provide platform independent abstractions to graphical user interfaces, regular expressions, networking etc. This in turn allows the application to integrate with the (Qt) theme of the desktop environment in use. (Aliranta and Pesonen 2013)

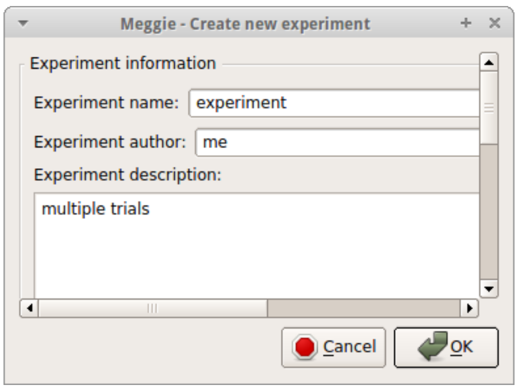
This chapter doesn't include the introduction of the source analysis tabs, because their functionalities were still under development when writing this thesis.

3.1 Creating an Experiment and Adding Subjects

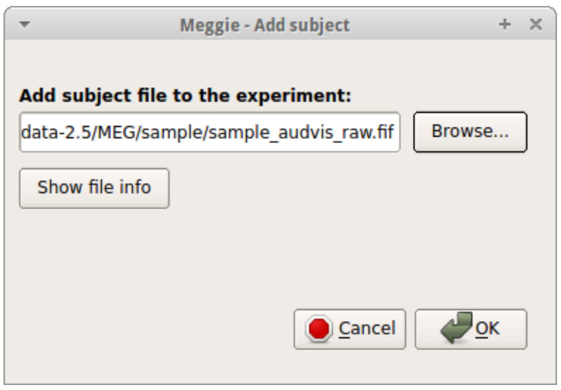
Originally, when the application was launched, the user was presented with an empty window with toolbar commands allowing the creation of a new experiment or loading a pre-existing one. Current view of Meggie after launching the application is shown in the figure 1, but without the subjects and subject info. Also, a single dialog handled the experiment creation and adding the measurement file to it. To enable multiple subjects, the dialog was then divided into two separate dialogs, the 'Create new experiment' dialog (fig. 2a) and the 'Add subject' dialog (fig. 2b), where it is possible to add subjects to the recently created experiment.

3.2 Preprocessing Tab

The 'Preprocessing' tab (fig. 3) includes options to apply MaxFilter (see 7.1.2) methods (see 4.3.1) to the measurement file to reduce external disturbances. The tab also holds MNE (see 7.1.1) methods to calculate SSP/PCA projections (see 4.3.2, 4.3.3) for artefacts caused by cardiac activity (ECG, electrocardiogram) and eye blinks (EOG, electro-oculogram), and view power spectral density (see (Gramfort et al. 2014) for ECG, EOG and power spectrum).



(a) Create new experiment dialog.


Meggie - Add subject
Add subject file to the experiment:
data-2.5/MEG/sample/sample_audvis_raw.fif
Browse...
Show file info
Cancel
OK

(b) Add subject dialog.

Figure 2: Creating an experiment and adding subject/subjects.

The methods already completed are indicated by the checkboxes in the area ’Preprocessing steps completed’.

3.2.1 ECG/EOG Dialog

ECG and EOG projections are calculated using almost all the same parameters. Therefore, only ’Compute ECG projections’ dialog (fig. 4) is presented here.

’Compute ECG projections’ dialog collects the user defined parameters to calculate the projection files that are applied to the measurement file to reduce the disturbances from the subject’s cardiac activity, or eye blinks with the ’Compute EOG projections’ dialog. Originally, the parameters setting and the computation was possible for a single measurement file only, but there was added a ’Batch processing’ area for several subjects as seen in the figure 4. ’Apply’ button applies the user defined parameters to the selected subject file on the list shown on the bottom left corner of the dialog. ’Apply to all’ button applies the parameters to all of the subject files on the list. After the ’Compute’ button is pressed and the ’Batch processing’ area is enabled the projections are calculated to all of the subject files on the list with the applied parameters. If the area is disabled, the computation is done for the currently active subject of the experiment.

3.3 Epoching Tab

’Epoching’ tab (fig. 5) has options for epoch creation, deletion, modification, exporting, importing and visualization. Currently, there are two epoch collections shown in the picture and the parameters of the selected Epochs2 collection are shown on the right side of the tab. For more information about epoching see 4.4.

3.3.1 Epoch Creation Dialog

’Epoch creation’ dialog (fig. 6) includes selections for the epoch collection name, length of the epochs with the start/end time selection, event IDs, event name and rejection parameters. Events added to the list can be saved into a file with Excel supported file format .csv.

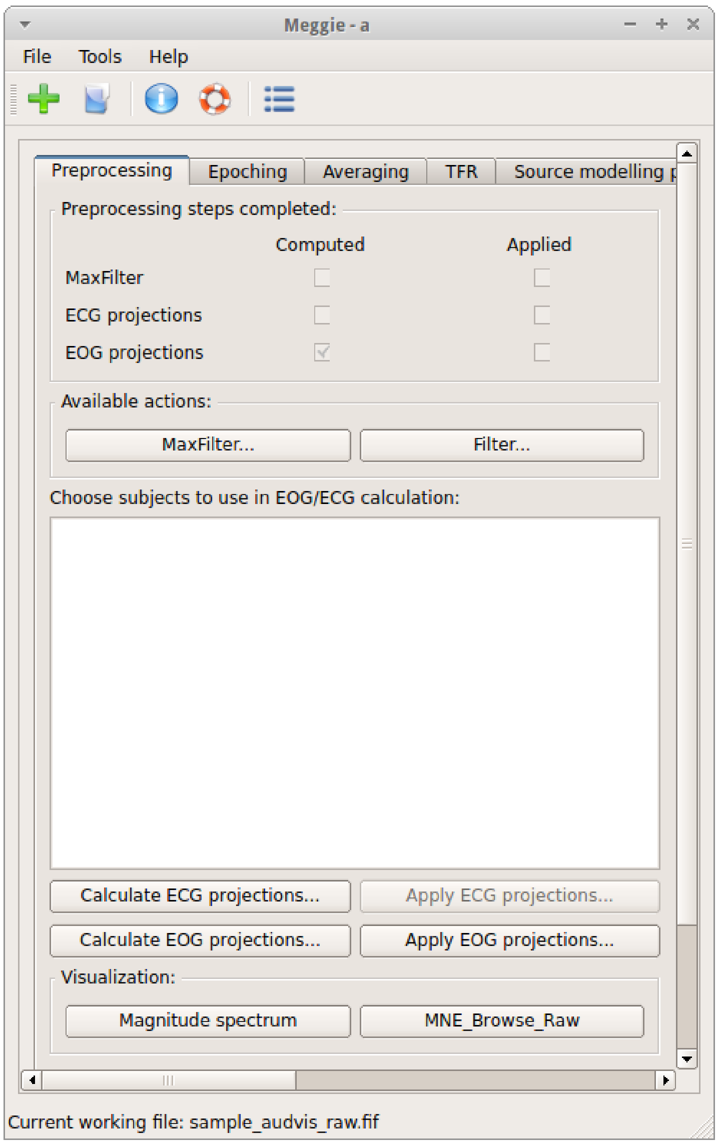
After ’Create epochs’ is clicked, the epoch collection is created for the chosen events with their lengths, rejections and chosen event IDs. The resulting epoch collection is added to the ’Epoch collections’ list on the ’Epoching’ tab in figure 5.

3.4 Averaging Tab

’Averaging’ tab (fig. 7) holds functionality for the methods used to average the created epoch collections on ’Epoching’ tab and to visualize the averaged epoch collection datasets. For more information about averaging see 4.4.

Pressing the ’Create evoked dataset’ button creates an averaged epoch collection file, later referred as evoked dataset/file in this thesis, for the selected epoch collection on the upper list on the left side of the tab. The averaging is done for all the events in the collection unless the user wants to average only specific events in it by making a selection on the list below the epoch collections list.

There can be multiple evoked datasets on the ’Evoked datasets’ list and the selected one can be visualized with the ’Visualize selected dataset’ button.



Preprocessing steps completed:

	Computed	Applied
MaxFilter	☐	☐
ECG projections	☐	☐
EOG projections	✓	☐

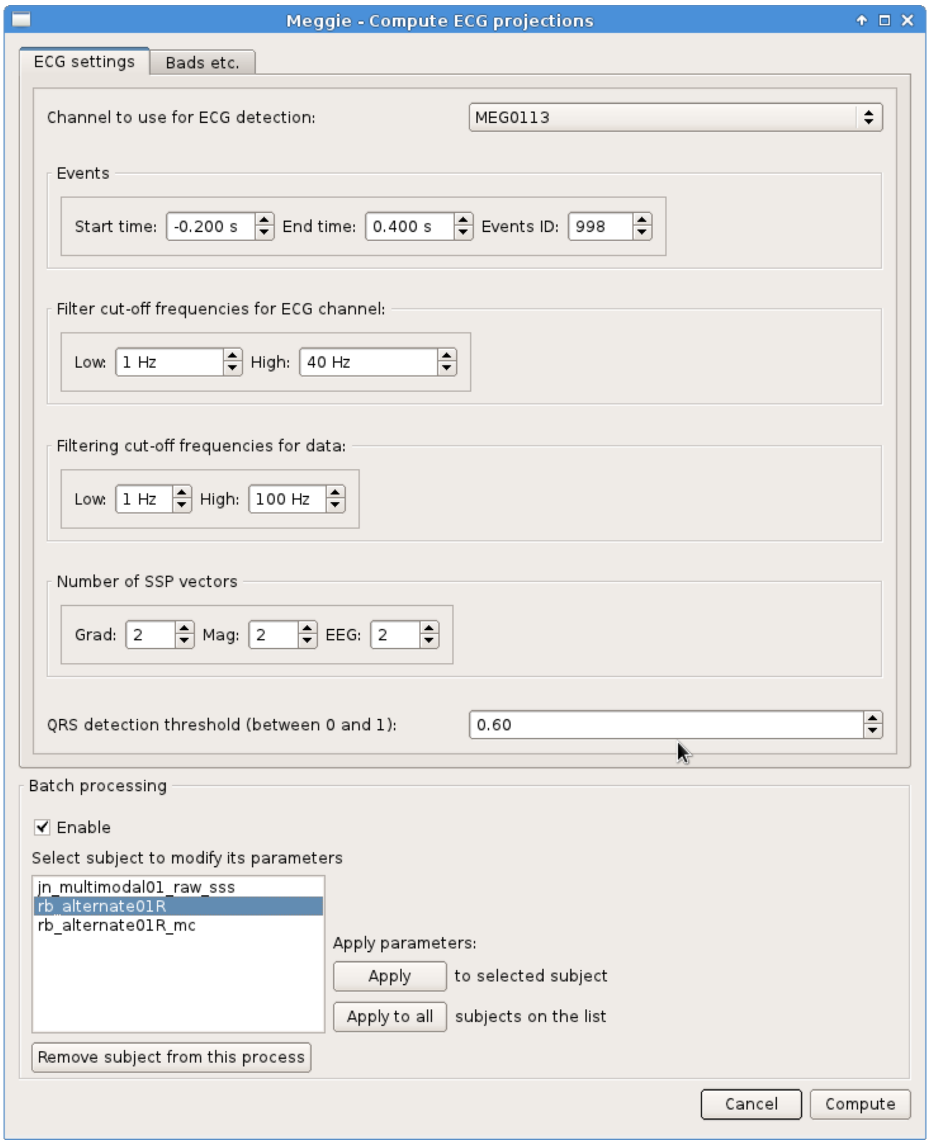
Available actions:

Choose subjects to use in EOG/ECG calculation:

Visualization:

Current working file: sample_audvis_raw.fif

Figure 3: Preprocessing tab of the main window displaying the completed preprocessing steps.



Meggie - Compute ECG projections

Channel to use for ECG detection: MEG0113

Events

Start time: -0.200 s End time: 0.400 s Events ID: 998

Filter cut-off frequencies for ECG channel:

Low: 1 Hz High: 40 Hz

Filtering cut-off frequencies for data:

Low: 1 Hz High: 100 Hz

Number of SSP vectors

Grad: 2 Mag: 2 EEG: 2

QRS detection threshold (between 0 and 1): 0.60

Batch processing

✓ Enable

Select subject to modify its parameters

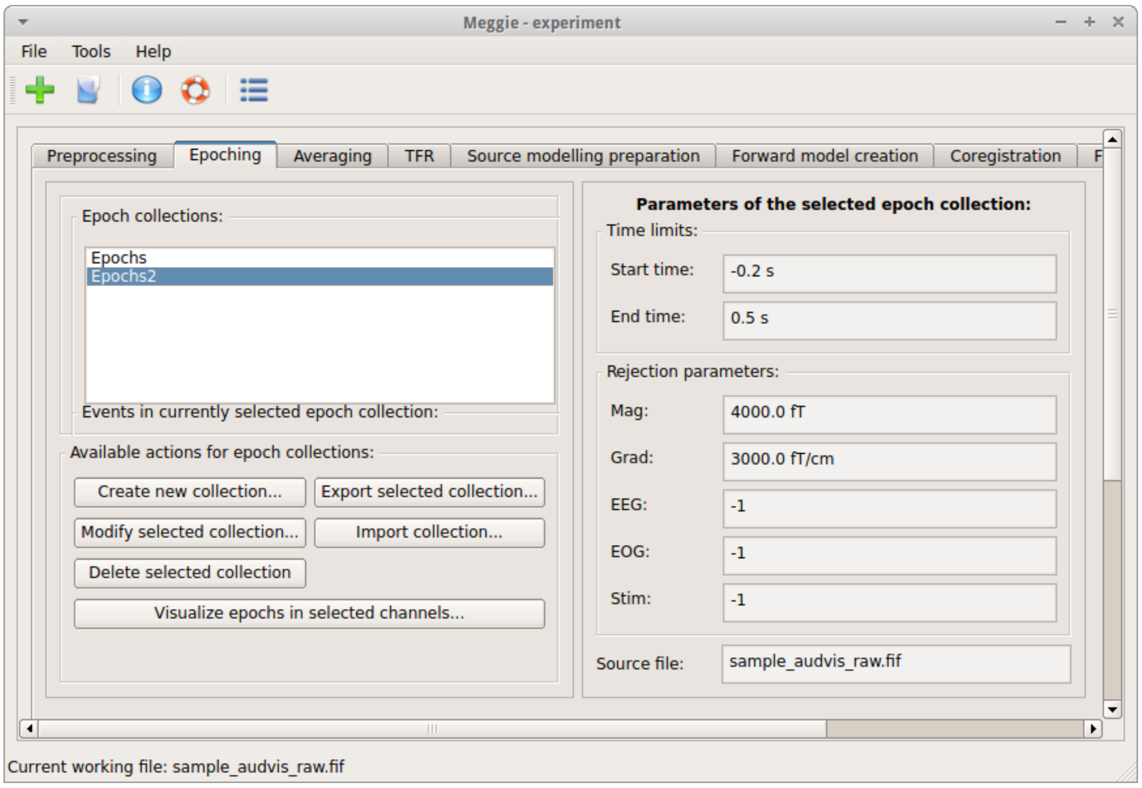
jn_multimodal01_raw_sss
rb_alternate01R
rb_alternate01R_mc

Apply parameters:

Apply to selected subject

Apply to all subjects on the list

Figure 4: Compute ECG projections dialog.


Meggie - experiment
File
Tools
Help
Preprocessing
Epoching
Averaging
TFR
Source modelling preparation
Forward model creation
Coregistration
F
Create new collection...
Export selected collection...
Modify selected collection...
Import collection...
Delete selected collection
Visualize epochs in selected channels...

Epoch collections:

Epochs
Epochs2

Events in currently selected epoch collection:

Available actions for epoch collections:

Parameters of the selected epoch collection:

Time limits:

Start time:	-0.2 s
End time:	0.5 s

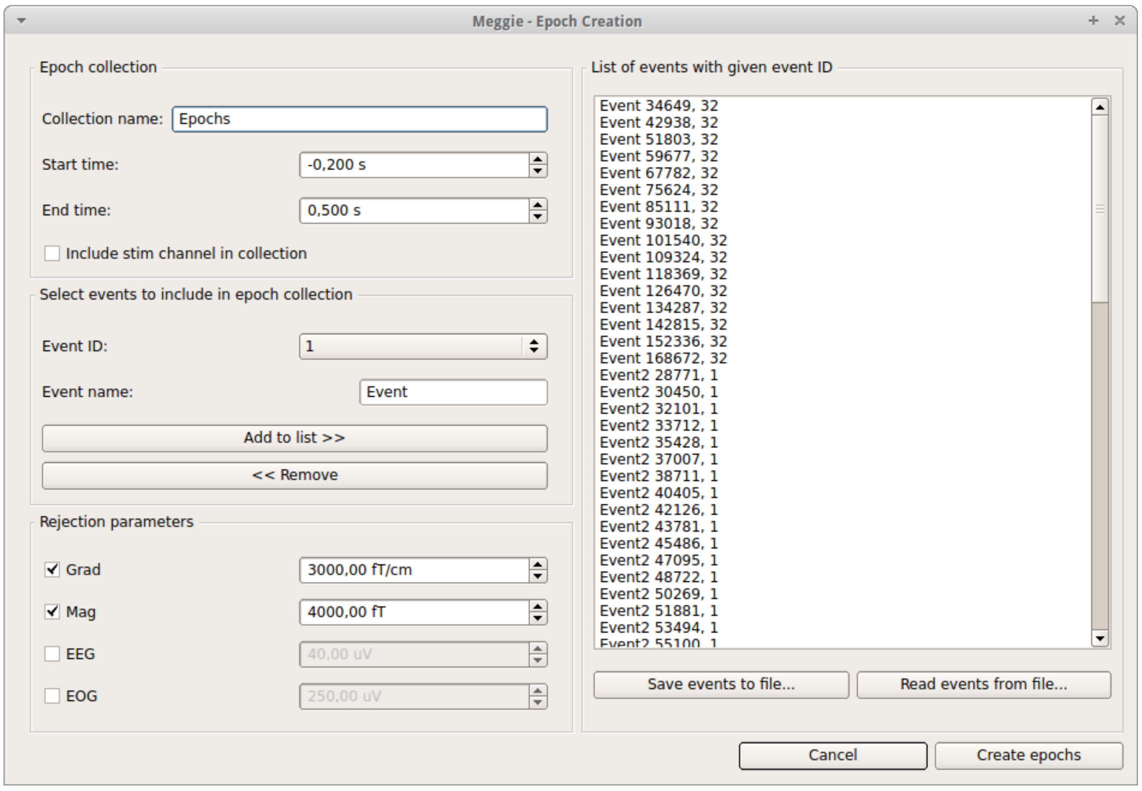
Rejection parameters:

Mag:	4000.0 fT
Grad:	3000.0 fT/cm
EEG:	-1
EOG:	-1
Stim:	-1

Source file: sample_audvis_raw.fif

Current working file: sample_audvis_raw.fif

Figure 5: Epoching tab of the main window.



Collection name: Epochs

Start time: -0,200 s

End time: 0,500 s

Include stim channel in collection [ ]

Event ID: 1

Event name: Event

Grad [✓] 3000,00 fT/cm

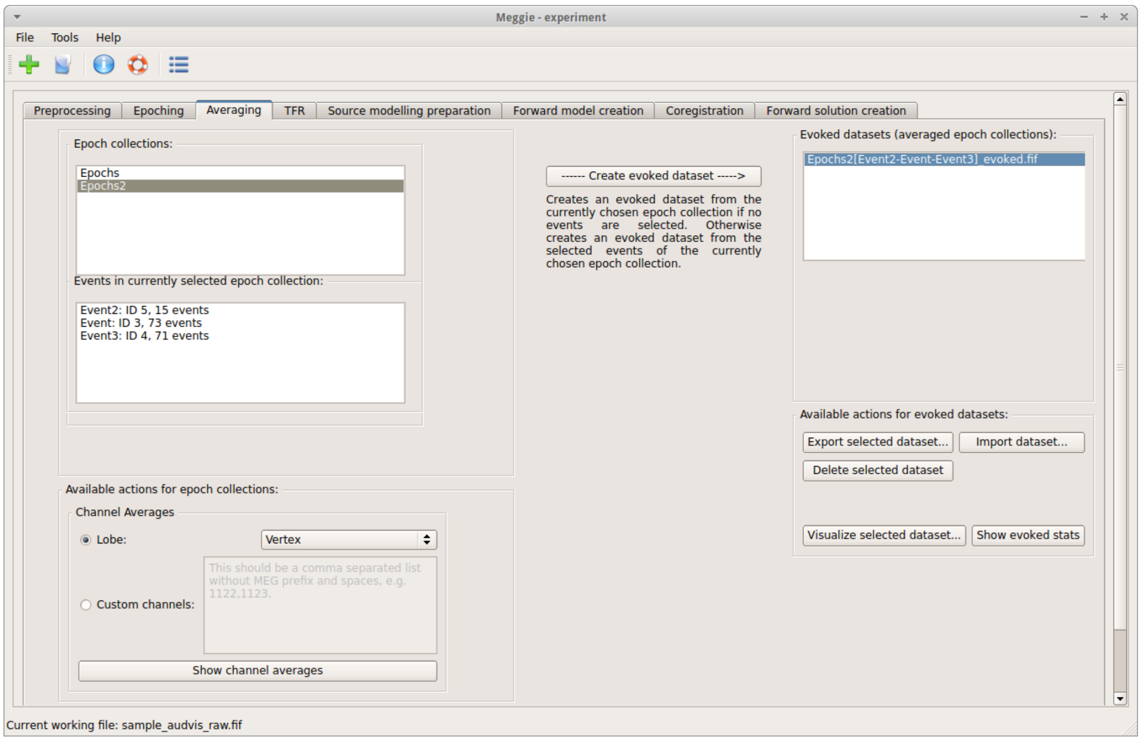
Mag [✓] 4000,00 fT

EEG [ ] 40,00 uV

EOG [ ] 250,00 uV

```
Event 34649, 32
Event 42938, 32
Event 51803, 32
Event 59677, 32
Event 67782, 32
Event 75624, 32
Event 85111, 32
Event 93018, 32
Event 101540, 32
Event 109324, 32
Event 118369, 32
Event 126470, 32
Event 134287, 32
Event 142815, 32
Event 152336, 32
Event 168672, 32
Event2 28771, 1
Event2 30450, 1
Event2 32101, 1
Event2 33712, 1
Event2 35428, 1
Event2 37007, 1
Event2 38711, 1
Event2 40405, 1
Event2 42126, 1
Event2 43781, 1
Event2 45486, 1
Event2 47095, 1
Event2 48722, 1
Event2 50269, 1
Event2 51881, 1
Event2 53494, 1
Event2 55100, 1
```

Figure 6: Epoch creation dialog.


Meggie - experiment
File
Tools
Help
Preprocessing
Epoching
Averaging
TFR
Source modelling preparation
Forward model creation
Coregistration
Forward solution creation
Epoch collections:
Events in currently selected epoch collection:
Available actions for epoch collections:
Channel Averages
Show channel averages
------ Create evoked dataset ----->
Evoked datasets (averaged epoch collections):
Available actions for evoked datasets:
Export selected dataset...
Import dataset...
Delete selected dataset
Visualize selected dataset...
Show evoked stats

Epoch collections:
Epochs
Epochs2

Events in currently selected epoch collection:
Event2: ID 5, 15 events
Event: ID 3, 73 events
Event3: ID 4, 71 events

Lobe: ● Vertex

Custom channels: ○ This should be a comma separated list without MEG prefix and spaces, e.g. 1122,1123.

Creates an evoked dataset from the currently chosen epoch collection if no events are selected. Otherwise creates an evoked dataset from the selected events of the currently chosen epoch collection.

Evoked datasets (averaged epoch collections):
Epochs2[Event2-Event-Event3]_evoked.fif

Current working file: sample_audvis_raw.fif

Figure 7: Averaging tab of the main window.

4 Magnetoencephalography

Magnetoencephalography (MEG) is used for mapping brain activity by recording magnetic fields produced by neuronal currents. This chapter shortly introduces theory behind MEG technique, including instrumentation, data acquisition, data preprocessing, data epoching and averaging, and spectral analysis; and its applicability in mapping brain activity compared to other popular techniques. The steps follow the guidelines provided by (Gross et al. 2013), which are applicable for basic research, and there are no recommendations related to hardware, software, data handling, participant questionnaires, or consent forms. There won't be comprehensive introduction to the methods since the scope of this thesis is in the development of Meggie.

4.1 MEG in Brief

MEG has a millisecond time resolution (Hari, Parkkonen, and Nangini 2010) which is suitable for tracking accurate neuronal timing during human perceptions, cognitive processes, motor activity and social interaction. The activities are caused by neuronal currents in brain. Neuron, a nerve cell, is specialized in transmitting these neuronal currents. It consists of the cell body (the soma), the dendrites (threadlike extensions) and the axon (a single long fiber) (Hämäläinen et al. 1993). Basically, the dendrites receive neuronal currents and carry them to the soma, and the axon sends the currents to other nerve cells, near or distant. Neuronal currents are electrical and, concidering electromagnetism, produce weak magnetic fields that are detectable outside the head (Hari, Parkkonen, and Nangini 2010).

These weak cerebral magnetic fields, about one billionth of the magnitude of the Earth's, can be detected by superconducting quantum interference device (SQUID) sensors (Hari, Parkkonen, and Nangini 2010; Hansen, Kringelbach, and Salmelin 2010). The SQUIDs convert magnetic flux into a recordable electric voltage (Hari, Parkkonen, and Nangini 2010). With their sensitivities to source localizations, the SQUIDs have enough sensitivity for practical MEG work (Hansen, Kringelbach, and Salmelin 2010). Another device with enough sensitivity, similar to SQUIDs, is an atomic magnetometer (Xia et al. 2006).

Elekta (headquarters in Stockholm, Sweden) MEG device, a helmet, consists of more than 300 SQUID sensors that are evenly located to cover all the areas of the cortex, the outermost layered structure of neural tissue in brain. Superconductivity is guaranteed with liquid helium in a container that is inside the device and above the sensor array. The distance from the head surface to the sensor coils should be minimized for optimal results, 2–3 cm in this case. A special container called Dewar is required to maintain about 300 Celsius difference across this relatively small distance. The device is placed into a magnetically shielded room to reduce disturbances caused by external sources. (Hansen, Kringelbach, and Salmelin 2010)

4.2 Data Acquisition

In MEG study, data acquisition must be carefully arranged to ensure optimal data quality. (Gross et al. 2013) suggests that the following elements are taken into account :

- **System setup**. Includes settings for sensors, helium level, interference suppression system, and ambient magnetic interference level.
- **Subject preparation**. Give basic instructions about the tasks, suggest to avoid eye movements/blinks, and remove magnetic materials.
- **Experimental design**. Single long trial or multiple trials. Event-related studies, time-frequency (TFR) studies.
- **General acquisition setup**. High sampling rate (3-4 times higher than the highest frequency of interest) and filter pass-band, recording electro-oculogram (EOG), electrocardiogram (ECG), electromyogram (EMG), eyetracking, digitized head shape.

Typically, two different type of oscillatory activity of the cortex are investigated from the acquired data (David, Kilner, and Friston 2006):

1. **Evoked oscillations (event-related potentials/fields)**. During the measurement, the subject is given short stimuli, e.g. shown pictures, that are phase-locked to the signal. Averaging these phase-locked events ensures that the background noise gets averaged out while the interesting signal from the cortical responses is preserved. To be able to estimate evoked power, the time-frequency power of the averaged events needs to be calculated.

2. **Induced oscillations (single long trial)**. This is the oscillatory activity of the neurons that is not phase-locked to the stimulus. A time-frequency power is calculated for the oscillating elements of the signal. The time-frequency power is then averaged to be able to estimate the induced oscillations. During the measurement the subject might have been reading a book or listening to music.

4.3 Data Preprocessing

MEG data preprocessing aims to remove data components unrelated to brain activity. Gross et al. classify interfering artefacts into three categories (Gross et al. 2013):

1. **System-related artefacts**. Caused by the helmet, for example noisy or broken sensors.
2. **External artefacts**. External magnetic fields arise disturbances. These magnetic fields are generated by power lines, traffic, elevators, etc.
3. **Physiological artefacts**. Caused by eye movements, eye blinks, cardiac and muscular activity, and head movements. The movement of magnetic objects or particles attached or implanted to the body may give rise to artefacts (e.g. eye make-up, hair spray, magnetized dental fillings).

Gross et al. also present two generally accepted strategies to deal with artefacts (Gross et al. 2013). First, identifying artefacts through visual inspection with or without automatic detection procedure, or using a combination of the both. The data segments contaminated by these artefacts are not included in further analysis. The second strategy reduces artefacts that are not caused by brain activity by means of several signal-processing techniques. These techniques preserve signals that originate from the brain and are mostly based on linear transformations (Gross and Ioannides 1999) or regression techniques applied to the sensor data (Ille, Berg, and Scherg 2002; Parra et al. 2005; Schlögl et al. 2007; Wallstrom et al. 2004). The linear transformations are calculated with several techniques explained later in this chapter.

In Meggie, signal-processing strategy is applied to removal of external (SSS method 4.3.1) and physiological (SSP method 4.3.2) artefacts, the visual inspection of contaminated data segments is applied to removal of system-related artefacts with a possibility to use automatic detection, and there are general line noise filtering methods, including band-pass, low-pass and high-pass filtering.

The order of the previously mentioned three artefact categories is also the advisable order of removal (Gross et al. 2013). The line noise filtering method can be applied any time when processing the data, but preferably between the first and second categories. Meggie supports this by visually sorting the components to the suggested order and disabling/enabling certain steps when necessary.

Preprocessing of evoked and induced type of data don't differ significantly. These two cases will be examined more closely in further analysis steps when splitting the data into epochs, averaging, and applying source space analysis. However, there is no need for close examination of the further analysis steps in this thesis since the focus is on preprocessing, but a brief introduction to them is given in the next two sections.

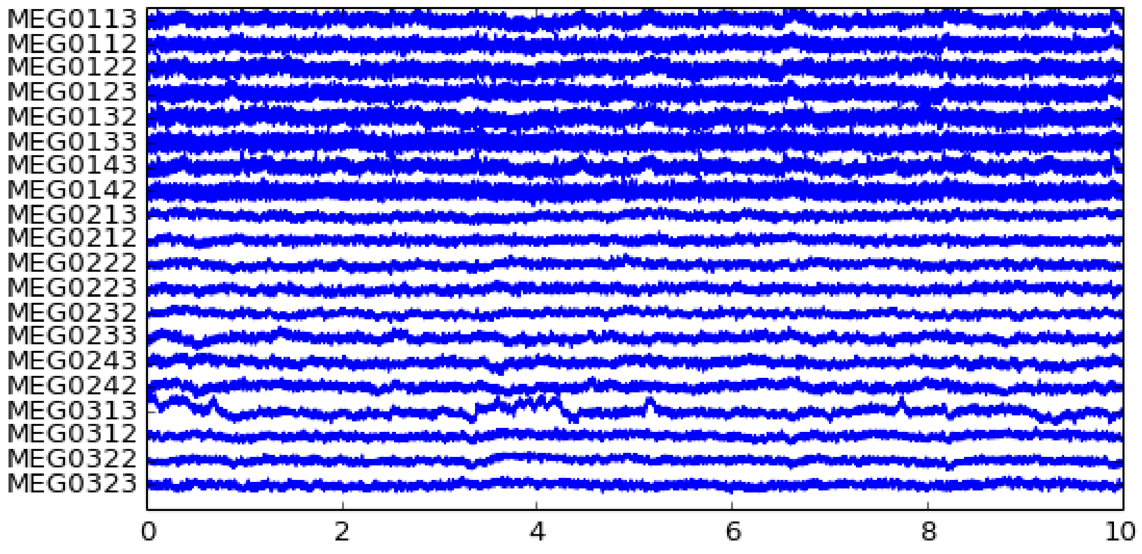
4.3.1 Signal-space Separation

Signal space separation (SSS) (Samu Taulu, Matti Kajola, and Juha Simola 2004) is the underlying method for removal of the external artefacts and it's provided by MaxFilter. The method is based on creating a subspace for the signals external to the sensor helmet and another subspace for the signals whose sources are inside the sensor helmet. The creation of the subspaces is possible using the known physical properties of magnetic fields. These subspaces are linearly independent which enables separating the measured data to contributions from outside and inside of the sensor helmet. After the separation the outside contribution can be removed (Hansen, Kringelbach, and Salmelin 2010). The effect of the method can be seen in figure 8.

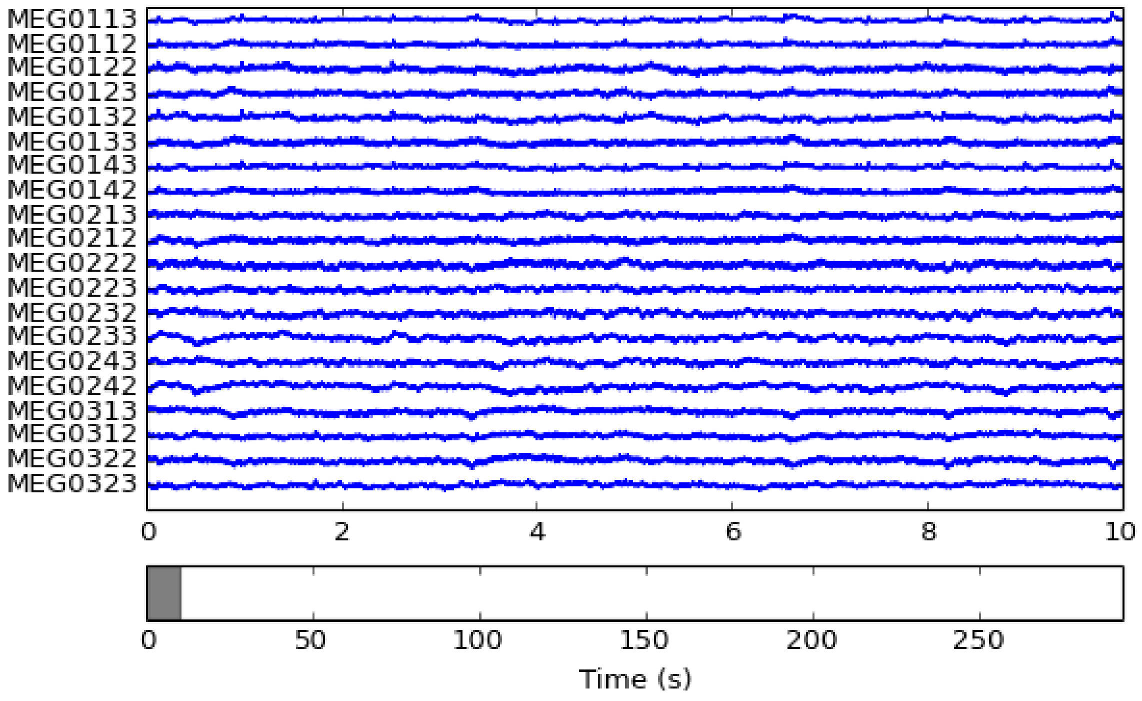
The linearly independent subspaces can be found using an equation

$$V(r) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \alpha_{lm} \frac{Y_{lm}(\theta, \varphi)}{r^{l+1}} + \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \beta_{lm} r^{l} Y_{lm}(\theta, \varphi),$$

where V is a harmonic scalar potential, α_{lm} and β_{lm} are scalars and and $r = \|r\|$ (S. Taulu, M. Kajola, and J. Simola 2003). The first part of the equation consists of the sources inside the helmet and the second part consists of the sources outside the helmet.



(a) SSS not applied.



(b) SSS applied.

Figure 8: RawFIFF.plot() function of MNE software package used on a measurement data before and after SSS calculation provided by MaxFilter.

4.3.2 Signal-space Projection

Signal-space projection (SSP) method estimates an interference subspace and applies a linear projection operator to the MEG data to remove the physiological (EOG and ECG) interferences (Gramfort et al. 2014). The method is provided by MNE.

In SSP method we first select k component vectors to form the matrix $K = (s_1, s_2, ..., s_k)$, where $1 \leq k < M$ and M is the number of component vectors that form the measured signal vector $m(t)$. The measured signals

$$m(t) = \sum_{i=1}^{M} a_i(t)s_i + n(t) = Sa(t) + n(t),$$

where $a_i(t)s_i$ is a component vector and $n(t)$ is noise, are then divided into two parts using formulas $s_{||} = P_{||}m$ and $s_{\perp} = P_{\perp}m$, where $P_{||}$ and $P_{\perp}$ are the projection operators, which are the actual tools for carrying out the SSP. The singular decomposition of the matrix $K = U \bigwedge V^{\top}$ is used to construct the operators. An orthonormal basis for the column space of K is spanned by the first k columns of U, therefore

$$P_{||} = U_k(U_k)^{\top}, P_{\perp} = I - P_{||}.$$

$s_{||}$ belongs to the subspace spanned by the component vectors in matrix K and it is an estimate of the signals produced by the known sources, and $s_{\perp}$ includes the rest of the measured signals $m(t)$ which cannot be spanned by K. The matrix K must be formed from characterised artefacts, e.g. EOG and ECG, to be able to study the artefact free $s_{\perp}$. (Uusitalo and Ilmoniemi 1997)

4.3.3 Principal Component Analysis

SSP method is often determined by principal component analysis (PCA) that constructs the projection operator (Gramfort et al. 2014). Often when large multivariate datasets are analyzed, such as MEG measurement data, it is desirable to reduce their dimensionality. The PCA does this by replacing the original variables by a smaller number of principal components, which are linear combinations of the original variables and present their variations (Jolliffe 2005).

Principal components are obtained using a basis transformation (Schölkopf, Smola, and Müller 1997). The idea is to diagonalize an estimation of the covariance matrix of the original data x_k, $k = 1, \dots, l$, $x_k \in R^N$, $\sum_{k=1}^{l} x_k = 0$, defined as

$$C = \frac{1}{l} \sum_{j=1}^{l} x_j x_j^\top.$$

The principal components are the new coordinates in the Eigenvector basis (Schölkopf, Smola, and Müller 1997). Simply put, the principal components are the new coordinate axes, where the origin is in the center of the data points and they are angled in a way that there is maximum variation among the points. The 'eigvector2' in figure 9 is a principal component. Because there is more variation among the data points in the 'eigvector2' angle than in the 'eigvector1' angle, the dimensionality reduction is done by projecting the data points onto the 'eigvector2'.

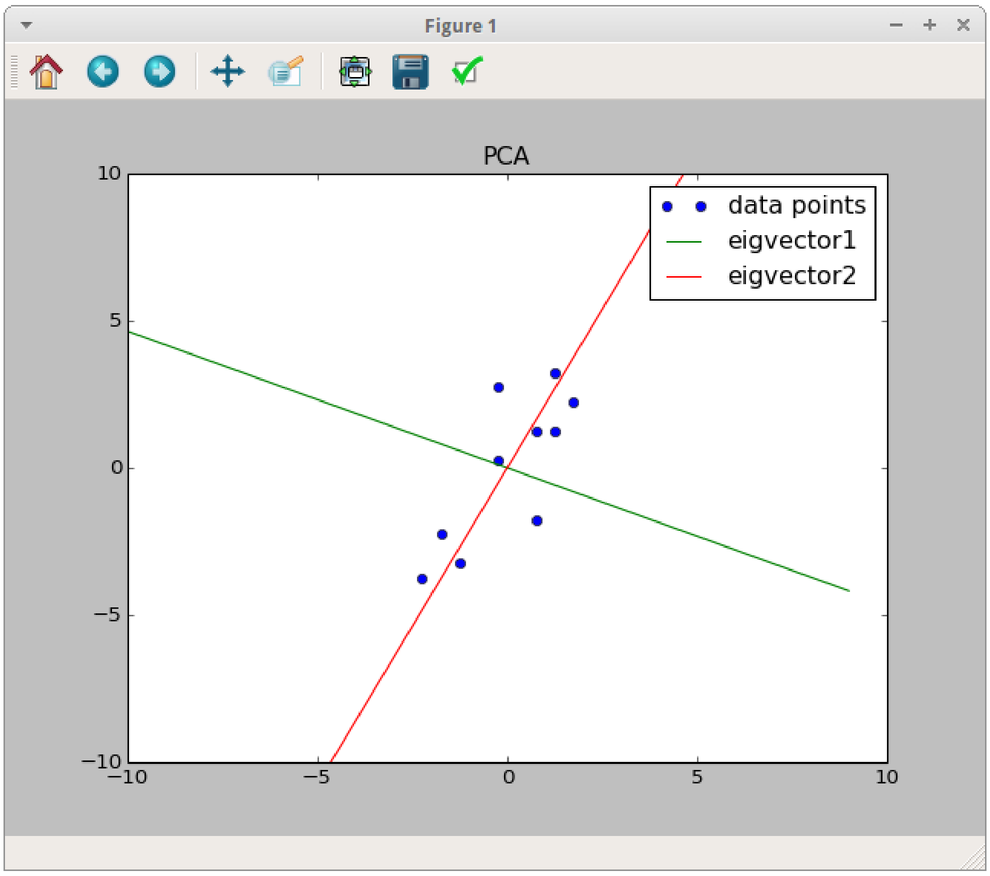


Figure 9: Normalized data with the eigenvectors.

4.4 Data Epoching and Averaging

In traditional MEG analysis the focus is on brain activity caused by short stimuli and averaging the data about these events (trial averaging), or epochs as defined earlier. It is desirable that the dataset has consistent brain responses to a certain type of stimulus to create an epoch collection. Averaging this epoch collection across trials increases the signal-to-noise ratio of the interesting brain responses, because we can assume that the rest of the data is uncorrelated noise, thus being timely inconsistent with the brain responses and will be averaged out. (Gross et al. 2013)

Gross et al. point out that the traditional trial averaging is no longer tenable in the light of recent research, since there have been studies (Laskaris, Fotopoulos, and Ioannides 2004; Mathewson et al. 2009; Ploner et al. 2006; Romei, Gross, and Thut 2012; Dijk et al. 2008) demonstrating that ongoing brain activity at the time of stimulus representation significantly affects behavior. Trial averaging removes all of this behavioral aspect from the analysis. However, it's still more common than single-trial analysis, which analyses the impact of sensory input with the dynamically changing state of the brain, but Gross et al. expect to see stronger focus on single-trial analysis in future publications. Even so, trial averaging has a proven clinical utility. (Gross et al. 2013)

There is a possibility for both trial averaging and single-trial analysis in Meggie.

4.5 Spectral Analysis

Spectral analysis transforms signals into the (time-)frequency domain where (transient) oscillatory signal components are more readily observable. Also, the oscillatory signal components have a more compact representation which is a key motivator to this analysis step. (Gross et al. 2013)

In spectral analysis, the estimation is done either by using parametric or non-parametric methods (Gross et al. 2013). Meggie uses a non-parametric time frequency analysis.

4.6 MEG in Comparison to EEG and fMRI

In addition to MEG, there exists several other noninvasive techniques for mapping brain activity. Two of the most popular ones, EEG and fMRI, are compared to MEG in this section.

MEG has both excellent time resolution and decent spatial resolution. Spatial resolution of MEG is sub-centimeter under favourable conditions (Hämäläinen et al. 1993). Although, (Hari, Parkkonen, and Nangini 2010) state that "At present, no single noninvasive brain imaging method is accurate both in time and space.". EEG has excellent time resolution but poor spatial resolution and fMRI has poor time resolution but excellent spatial resolution. MEG specializes capturing activation in fissural cortex where currents produce stronger magnetic fields because of their tangential movement to skull (Hari, Parkkonen, and Nangini 2010). These areas can't be reached as easily with other electrophysiological means, not even with intracranial recordings (Hari, Parkkonen, and Nangini 2010).

MEG monitors cortical activation sequences with less interference from the skull and brain tissues than EEG. However, there is a good synergy between MEG and EEG because of their unique sensitivities to sources. MEG focuses on tangential currents and EEG reflects currents of all orientations. (Hari, Parkkonen, and Nangini 2010)

fMRI views neuronal phenomena indirectly by tracking the oxygenation of blood flowing near active neurons (Hari, Parkkonen, and Nangini 2010). On the other hand, MEG and EEG signals are captured directly from neuronal electrical activity. This means that fMRI signals show relative neuronal activity whereas MEG and EEG signals show absolute neuronal activity.

(Hari, Parkkonen, and Nangini 2010) give more detailed information about MEG versus EEG and fMRI, and (Hämäläinen et al. 1993) compare differences more closely between MEG and EEG.

In the end, the best possible method for a problem at hand is ambiguous and a combination of the three presented methods might generate better results than any of the methods individually. fMRI provides spatial resolution, and MEG and EEG provide temporal resolution to the problem.

5 Methodology

The main theoretical phase of the thesis focuses on the evaluation of the architecture of Meggie, although the practical work was aimed at the development. The reason for this is that there was no theoretical approach, as in clear guidelines, to guide the solutions made during the development phase. This lead to unclear documentation and reasoning about the practices and solutions, which indeed would have been educational for the reader. However, the development phase had the most beneficial role of this thesis, and the requirements of that phase will be listed and given an explanation for their implementation in chapter 7.

This chapter presents the methods used to evaluate the architecture of Meggie and how the development phase was carried through. Memory management is also included, because it's part of the architectural design of Meggie and further displays the interaction between the model classes. It also greatly impacted on the design decisions during the development.

5.1 Theoretical Phase

The overall architecture of Meggie will be viewed with the 4+1 view model designed by Philippe Kruchten (Kruchten 1995). The selected diagrams will be a process diagram and sequence diagram representing the logical view, an activity diagram representing the process view, a component diagram representing the development view, a deployment diagram representing the physical view, and the description of the architecture will be illustrated with a small set of use cases and/or scenarios. In addition, there are several quality attributes, chosen from the Kruchten's article, that will be examined under the chapter 6.

5.2 Practical Phase

As mentioned several times earlier, the practical phase of this thesis is based on the further development of Meggie. The Implementation chapter 7 includes the requirements that were implemented during the development, and every requirement is given a short explanation for their implementation. The requirements are elicited and divided according to the scenarios given by the customer before the practical phase began. The scenarios were 'Enable batch processing for the removal of EOG/ECG artefacts', 'Correct the preprocessing functionality' and 'Correct the epoching and averaging functionality'. Also, a section for nonfunctional requirements is added to cover the issues with the memory management. The scenario-based approach was applied to reduce complexity in this interdisciplinary development (Weidenhaupt et al. 1998). Scenarios may represent various stakeholder roles in a system (Kazman et al. 1996). Clearly the three scenarios above were aimed for the developers of Meggie, and mainly for the author of this thesis.

Since the memory management had an important role in the architectural design, it will have its own section under the chapter 7. The memory management focuses on those functionalities of Meggie that require reading the measurement data files or releasing memory from the files occupying the memory. The functionalities include creating an experiment, adding a subject, activating a subject, etc., and all those cases will be inspected separately.

6 Architecture

This chapter represents a 4+1 architectural view model of Meggie after the completion of the development phase of this thesis. This allows the developers to understand the architecture of the application which hasn't been viewed before this thesis. Also, there are observations on weak architectural designs of which some are corrected and explained in the last section of this chapter.

Some of the quality attributes from the Kruchten's article are included, given a simple reasoning over them without going into details or formulas. The inspected quality attributes are availability, performance, reliability, modifiability and scalability. Their definitions can be found in the book (Wieger 2013). Although the effect of the memory management falls under the quality attribute efficiency, it is inspected under the performance.

6.1 Logical View

Logical view displays the functionalities that are provided to the end users of an application (Kruchten 1995). Process diagram and several sequence diagrams were chosen to represent the logical view of Meggie. Several use case scenarios are selected for the sequence diagrams to cover the key elements of the software. The selected scenarios are 'Create a new experiment', 'Add a subject' and 'EOG batch process'.

In figure 10 and 11 the 'ACTIONS' area of the blocks display the provided actions/functionalities of Meggie. The blocks that have a darker background colour are used for visualizing purposes and have no effect on the data flow. Each block except the visualizing blocks have 'INPUT' and 'OUTPUT' areas to display the input file needed for the actions and the output file produced by the actions. The order of the actions in the process diagram follows the steps given in section 4.3 and the functionalities of Meggie are placed in this order to help the user understand the process flow.

Figure 10 represents the preprocessing steps available for reducing the disturbances of the MEG measurement data. After the preprocessing steps are completed, the output is a fully preprocessed MEG file, and the user continues to the functionalities provided by the epoching and averaging steps in figure 11.

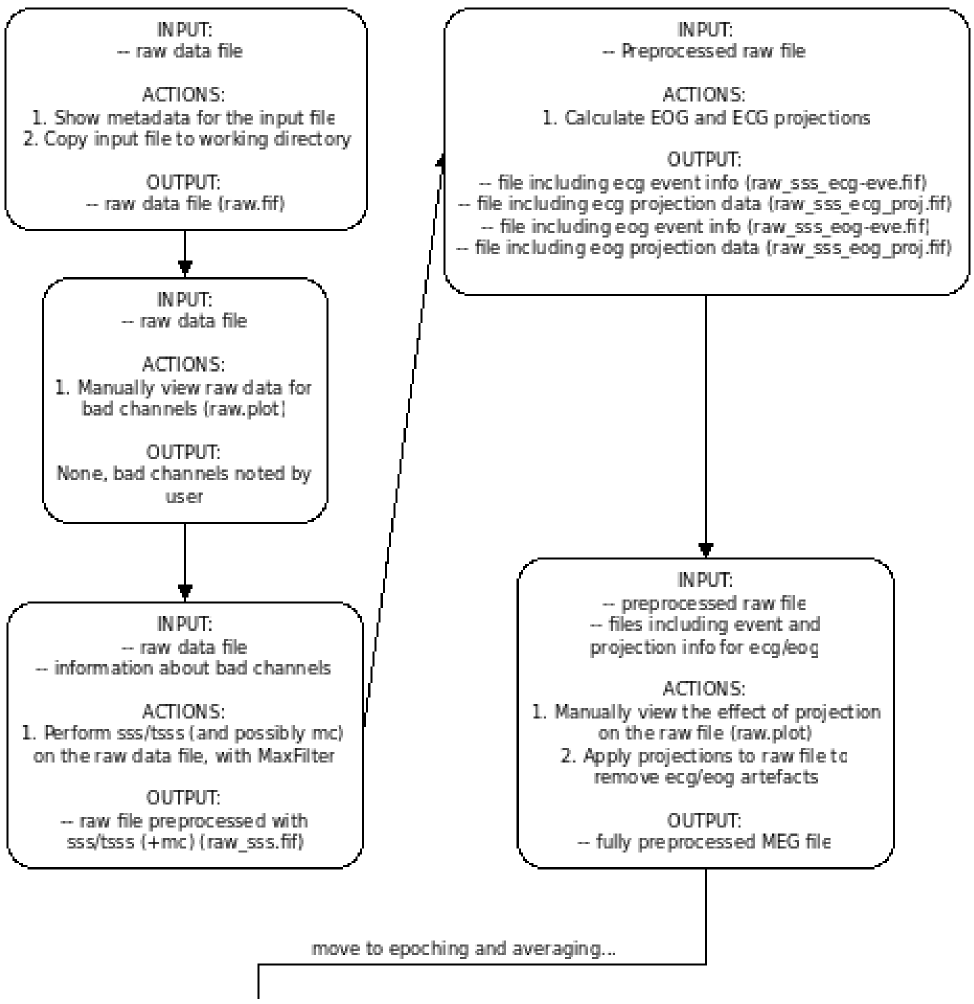

INPUT:
-- raw data file
ACTIONS:
1. Show metadata for the input file
2. Copy input file to working directory
OUTPUT:
-- raw data file (raw.fif)
INPUT:
-- raw data file
ACTIONS:
1. Manually view raw data for bad channels (raw.plot)
OUTPUT:
None, bad channels noted by user
INPUT:
-- raw data file
-- information about bad channels
ACTIONS:
1. Perform sss/tsss (and possibly mc) on the raw data file, with MaxFilter
OUTPUT:
-- raw file preprocessed with sss/tsss (+mc) (raw_sss.fif)
INPUT:
-- Preprocessed raw file
ACTIONS:
1. Calculate EOG and ECG projections
OUTPUT:
-- file including ecg event info (raw_sss_ecg-eve.fif)
-- file including ecg projection data (raw_sss_ecg_proj.fif)
-- file including eog event info (raw_sss_eog-eve.fif)
-- file including eog projection data (raw_sss_eog_proj.fif)
INPUT:
-- preprocessed raw file
-- files including event and projection info for ecg/eog
ACTIONS:
1. Manually view the effect of projection on the raw file (raw.plot)
2. Apply projections to raw file to remove ecg/eog artefacts
OUTPUT:
-- fully preprocessed MEG file
move to epoching and averaging...

Figure 10: Process diagram displaying the preprocessing steps.

Since the process diagram mostly includes the MEG analysis data flow, the experiment and subject creation functionalities are displayed with the help of the sequence diagrams in figures 12 and 13. These sequence diagrams also display the overall architecture of how the data is stored into the system running Meggie and being available for the ongoing processing steps.

The user creates an experiment folder with the steps seen in the sequence diagram in figure 12. After the creation, the user wants to add a subject to the existing experiment using the steps provided by the diagram in figure 13. The subject that is added last is then activated, which puts Meggie into a state where the user can access most of the processing steps, such as epoch creation and EOG batch processing (fig. 14). Although, it's not wise to create epochs before the preprocessing is done, so it's advisable to follow the processing sequence presented in chapter 4. The 'EOG batch process' scenario was chosen to represent the architectural design solutions made during the development phase of this thesis. It is also identical to the scenario 'ECG batch process' and, after all, these two scenarios were the main reason for the topic of this thesis.

The sequence diagrams show generally logical order of operations between the classes. Although, minor illogical call tracks can be seen, e.g. in ' Add a subject' scenario where the measurement file is opened calling MNE via FileManager class, but it is saved via Subject class. As seen in the sequence diagrams, FileManager, Experiment and Subject classes handle the most of the things happening in the model, such as reading files from the disk, saving files to the disk, and transferring objects and attributes to other model or view classes when needed. Here the words model and view derive from the model-view-controller (MVC) architectural pattern. The user's actions on the dialogs and main window have either straight call to MNE methods or they go via Caller or FileManager class. This is one of the non-standard design solutions that creates confusion, and the developer must decide whether it's important to transfer a specific call via the mentioned classes.

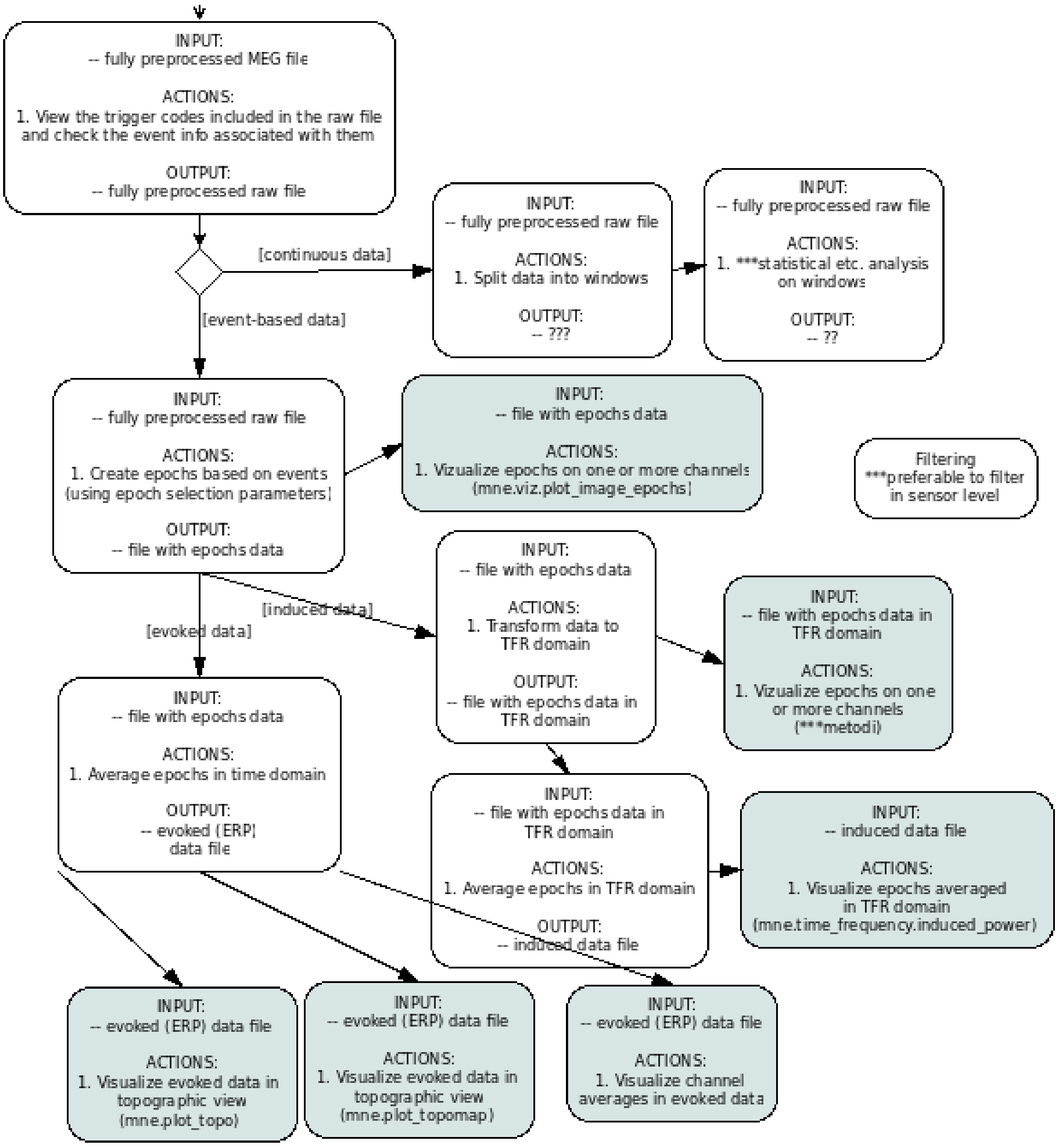

INPUT:
-- fully preprocessed MEG file
ACTIONS:
1. View the trigger codes included in the raw file and check the event info associated with them
OUTPUT:
-- fully preprocessed raw file
[continuous data]
[event-based data]
INPUT:
-- fully preprocessed raw file
ACTIONS:
1. Split data into windows
OUTPUT:
-- ???
INPUT:
-- fully preprocessed raw file
ACTIONS:
1. ***statistical etc. analysis on windows
OUTPUT:
-- ??
INPUT:
-- fully preprocessed raw file
ACTIONS:
1. Create epochs based on events (using epoch selection parameters)
OUTPUT:
-- file with epochs data
INPUT:
-- file with epochs data
ACTIONS:
1. Vizualize epochs on one or more channels (mne.viz.plot_image_epochs)
Filtering
***preferable to filter in sensor level
[induced data]
[evoked data]
INPUT:
-- file with epochs data
ACTIONS:
1. Transform data to TFR domain
OUTPUT:
-- file with epochs data in TFR domain
INPUT:
-- file with epochs data in TFR domain
ACTIONS:
1. Vizualize epochs on one or more channels (***metodi)
INPUT:
-- file with epochs data
ACTIONS:
1. Average epochs in time domain
OUTPUT:
-- evoked (ERP) data file
INPUT:
-- file with epochs data in TFR domain
ACTIONS:
1. Average epochs in TFR domain
OUTPUT:
-- induced data file
INPUT:
-- induced data file
ACTIONS:
1. Visualize epochs averaged in TFR domain (mne.time_frequency.induced_power)
INPUT:
-- evoked (ERP) data file
ACTIONS:
1. Visualize evoked data in topographic view (mne.plot_topo)
INPUT:
-- evoked (ERP) data file
ACTIONS:
1. Visualize evoked data in topographic view (mne.plot_topomap)
INPUT:
-- evoked (ERP) data file
ACTIONS:
1. Visualize channel averages in evoked data

Figure 11: Process diagram displaying the epoching and averaging steps.

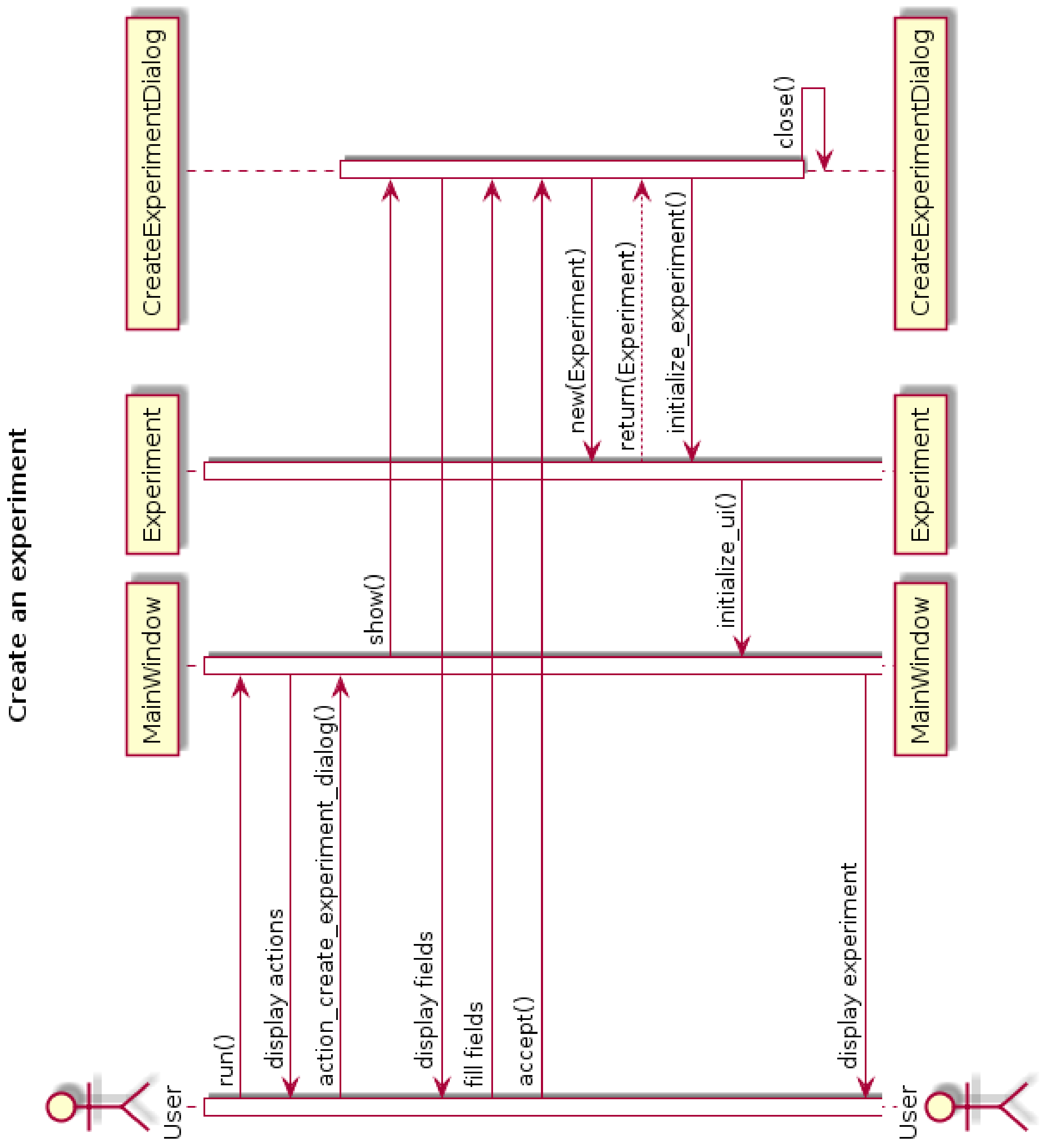


Figure 12: Sequence diagram of the use case 'Create a new experiment'.

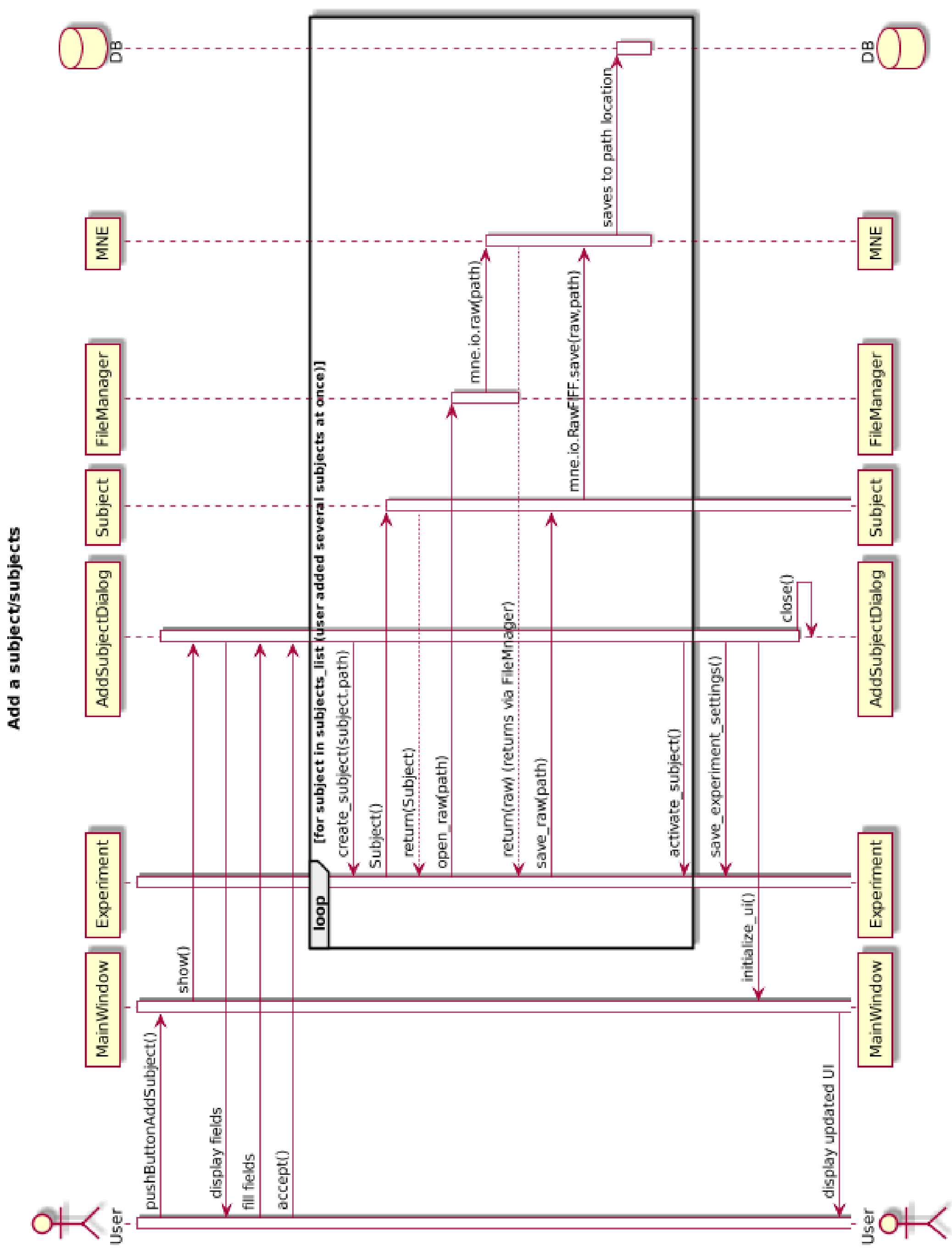


Figure 13: Sequence diagram of the use case ’Add a subject’.

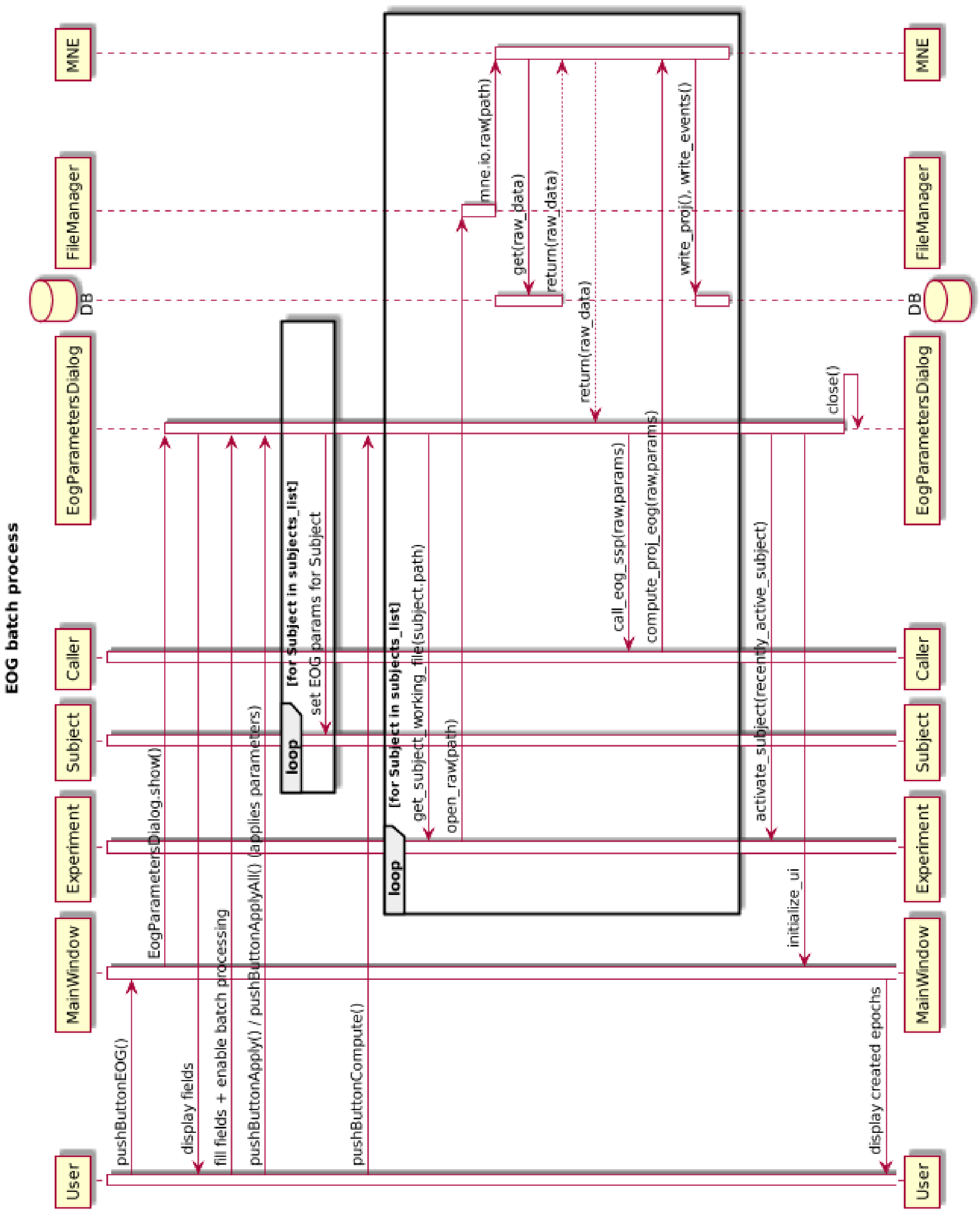

EOG batch process
User
MainWindow
Experiment
Subject
Caller
EogParametersDialog
DB
FileManager
MNE
pushButtonEOG()
EogParametersDialog.show()
display fields
fill fields + enable batch processing
pushButtonApply() / pushButtonApplyAll() (applies parameters)
loop
[for Subject in subjects_list]
set EOG params for Subject
pushButtonCompute()
loop
[for Subject in subjects_list]
get_subject_working_file(subject.path)
open_raw(path)
mne.io.raw(path)
get(raw_data)
return(raw_data)
return(raw_data)
call_eog_ssp(raw,params)
compute_proj_eog(raw,params)
write_proj(), write_events()
close()
activate_subject(recently_active_subject)
initialize_ui
display created epochs

Figure 14: Sequence diagram of the use case ’EOG batch process’.

6.2 Process View

Process view addresses system performance and availability (Kruchten 1995). Things that need to take into account include concurrency, distribution, system integrity and fault-tolerance. The process view in this section represents the actions that are available for the user at certain phases of MEG analysis (fig. 15). The performance issues are inspected in section 6.4.

Meggie is packaged with the source code files including the code_meggie and ui packages (fig. 16), and the external libraries MNE and PyQt4. The remaining external software FreeSurfer (Martinos Center for Biomedical Imaging, Charlestown, Massachusetts, North America) and Canopy (Enthought, Austin, Texas, North America) are freeware and must be manually installed to the system intended to run Meggie. The users are notified of the correct versions of the packages and they shouldn't downgrade or upgrade the versions, unless Meggie is further developed. Meggie is tested and should be run on Linux-based operating system Fedora or Ubuntu. If the user belongs to the target group, he/she probably has most of the needed packages and knows how to install the rest. After the installation is done, the user starts the application by running the run.py file in the Meggie root directory. Meggie holds a preferences dialog where the user can set the correct locations for the external tools.

Meggie is an offline, single desktop application, so it is always runnable after the installation. The actions of Meggie are limited while executing long-lasting calculations, which forces the user to wait for them to finish. Allowing the execution of two or more simultaneous actions might result in incorrect calculations, especially when modifying the same file.

Most of the actions of Meggie are available right after a subject is added to the experiment for the first time, as displayed in the activity diagram in figure 15. The reason for this is that there is no simple way to check the measurement file whether some actions are already completed. A filename check works as long as the actions are made within Meggie and if the user doesn't rename the file afterwards. Anyhow, the user might need to add a subject file where the origin and the completed actions are unknown to him. Therefore, the solution was to let the user decide whether a file needs a certain action by viewing the raw channel data which is always enabled when a subject is activated. A completion of several actions requires a file generated from the measurement file that is then further processed or visualized with those follow-up actions, thus aren't available at any time, e.g. 'Visualize averaged dataset' action in figure 15.

Enabling the actions at any time during the analysis, the researcher is able to execute unique research purposes. This is why we didn't want Meggie to limit the user's actions, but keep in mind that the user is also experienced in neuroscience and won't execute dull analysis chains. Although, since students are one of the target groups of the application, it would be advicable to give the user an additional information about correct action sequences in Meggie which should be concidered in the further development.

Meggie doesn't crash during erroneous calculation, but attempts to prevent any modifications to the files and resets the state as it was before the action. The user is notified of an error with a messagebox including an explanation of the user's action that caused the error, a suggestion to fix the error and a traceback to its origin. All in all the recovery process is short in Meggie.

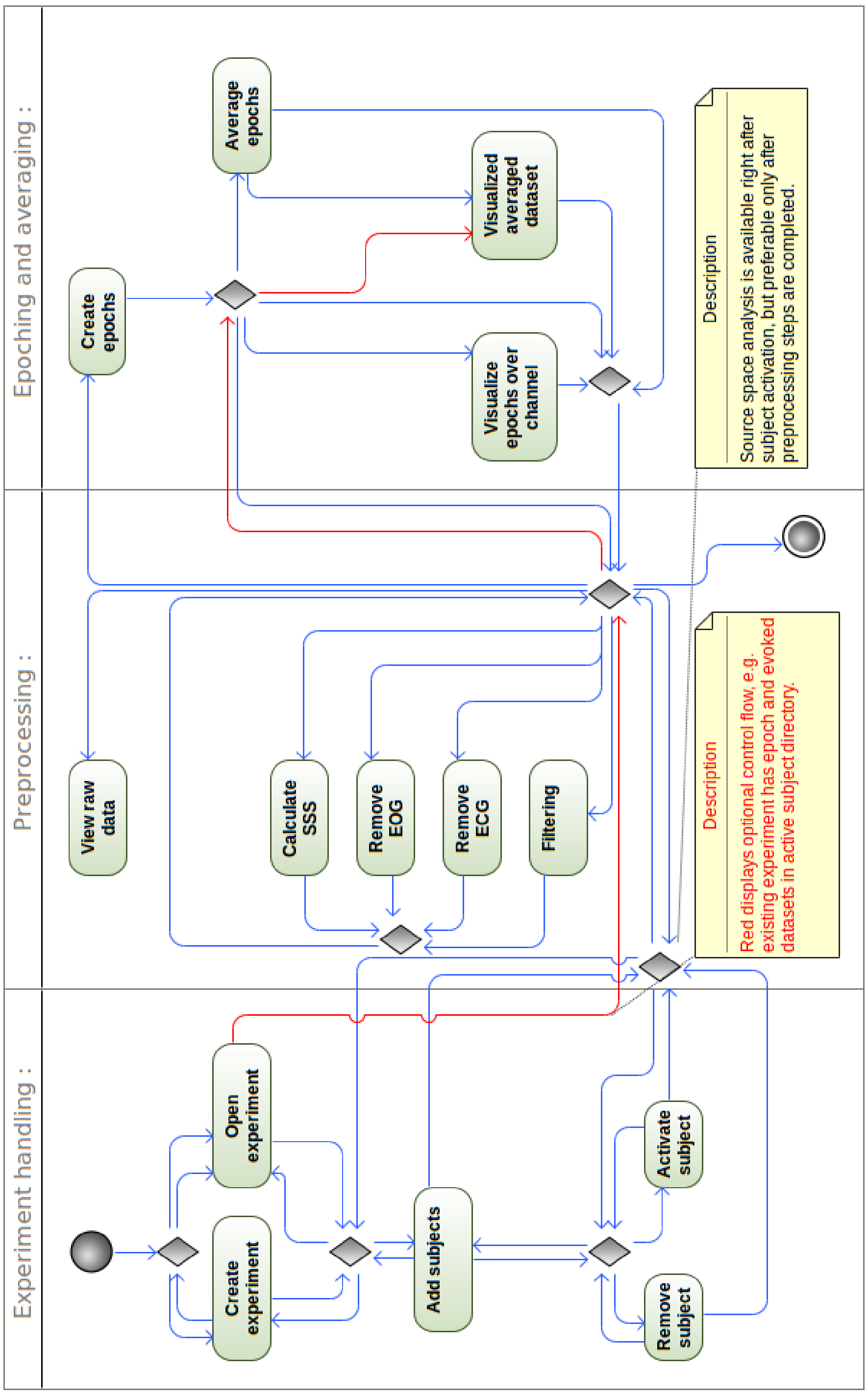

Experiment handling :
Preprocessing :
Epoching and averaging :
Create experiment
Open experiment
Add subjects
Remove subject
Activate subject
View raw data
Calculate SSS
Remove EOG
Remove ECG
Filtering
Create epochs
Average epochs
Visualize epochs over channel
Visualized averaged dataset
Description
Red displays optional control flow, e.g. existing experiment has epoch and evoked datasets in active subject directory.
Description
Source space analysis is available right after subject activation, but preferable only after preprocessing steps are completed.

Figure 15: Activity diagram of Meggie.

6.3 Development View

Development view shows the application modules and their organization in the software-development environment, as seen by the developers. The issues that need to be addressed are the internal requirements of the application, such as ease of development, software management, reuse and constraints of the toolset used. (Kruchten 1995)

All the class files of Meggie are divided into several small packages which are then divided into two large packages, as seen in figure 16. The other large package, code_meggie, is composed of source code classes that create the model of Meggie, and the other, ui package, includes all the UI source code classes and their main methods that create the view of Meggie. The words model and view derive from the MVC architectural pattern. The external software package mostly provides methods from MNE-Python package, but Meggie uses several methods from MNE-C, FreeSurfer and matplotlib. The dependency arrows in the diagram are drawn with different colours to simplify the figure: blue for the model, red for the view and green for the external packages.

From the beginning of the development, the goal has been to separate the codes of the view and the model, which has been covered with a requirement. As previously mentioned, the classes of Meggie are divided into packages code_meggie and ui according to the goal. This makes the modifications of the UI more simple, such as rearranging existing dialogs or adding new components to the dialogs or main window. The model classes only provide the needed information to the view and the view only displays what is provided. However, Meggie still has a few lines of code in the view classes that shouldn't belong in them, i.e. epoch creation code in the MainWindow class.

Since MNE is an external package and currently under development, there are modifications in the functionalities it provides. The developers of Meggie must keep that in mind when updating to the newest versions and should run basic tests to the modified functionalities. Anyhow, most of the updated functionalities run with a better performance or are easier to implement in Meggie.

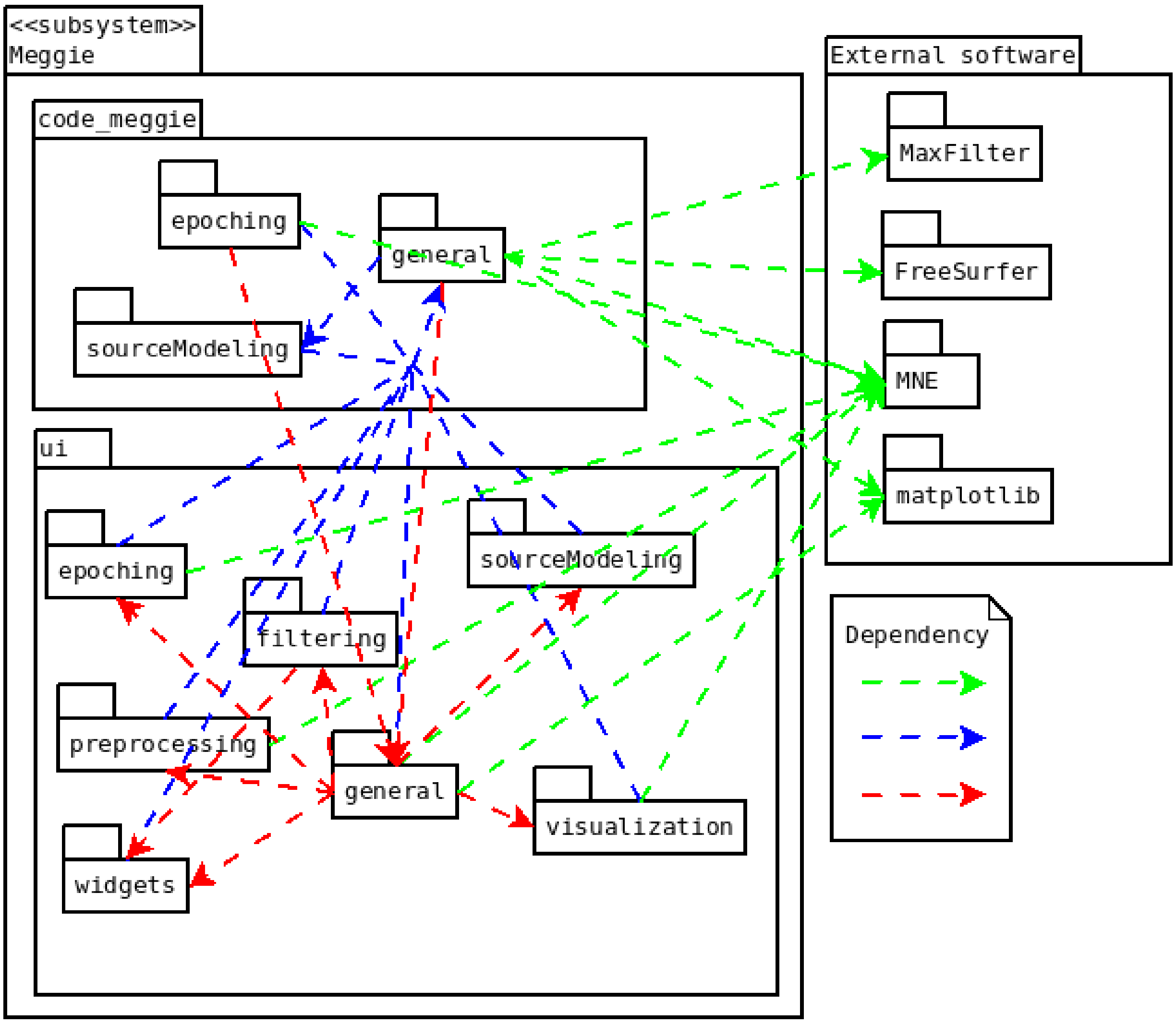


Figure 16: Package diagram of Meggie.

6.4 Physical View

The physical view is used for displaying the non-functional requirements of a software (Kruchten 1995). It addresses system availability, reliability, performance and scalability (Kruchten 1995). Since the availability issues were covered in section 6.2, the reliability and performance are going to be inspected here. The view that provides sufficient information is a deployment diagram.

The reliability of Meggie is not guaranteed to all of the processing steps, which is mainly affected by two factors. The first is the nonexisting testing phase and small number of test data files, and the second is the constantly changing external package MNE. Many new errors occurred after the newest version 0.8.6 was put to use, which was after the practical phase of this thesis, and Meggie couldn't recover from some of them without a restart. Ignoring the unexpected errors, Meggie should be able to run through a full analysis chain to a measurement file without crashing, even if errors occur during the runtime. Meggie recovers fast from the errors and is again available for further actions.

The performance of Meggie centers to the memory management and the used external libraries. The external libraries provide all the long lasting computations that modify the measurement files in Meggie, which leaves the developers unable to have control over them. Also, during those computations all the actions of Meggie are disabled. Fixing this issue with threading is considered to the further development of the application.

As mentioned in the earlier sections, the measurement files must be preloaded into the system memory to allow faster data manipulations. This requirement was set by the MNE software that we, developers, have decided to implement. Furthermore, allowing multiple subjects in one experiment created a problem of having multiple measurement files at once in the memory. This lead to the need for the memory management and decision of having only one measurement file at once in the memory. Altogether, this slows the subject activation sequence by forcing to release the memory from the previously active subject before another subject is activated and its measurement file is read into the memory. A similar problem occurs when trying to remove the EOG or ECG artefacts from multiple subjects during one batch process. The projection operators must be calculated for one measurement file at a time that is first read into the memory and then released before the calculation for another one. This allows to run Meggie in a system with a rather low memory (3 GB was enough in the system used for the batch processing development and testing phase with files less than 500 MB) and processing capabilities (AMD Athlon(tm) 64 X2 Dual Core Processor 5600+), but still the computations should be faster with a higher system memory (8 GB is recommended). Also, the OS will have a lot more free memory for its use, so the user is able to complete other tasks meanwhile. Without taking care of this problem, the 3 GB memory in the testing system created a lot of problems with an experiment that had only two subjects in it, making the OS unresponsive at times.

Scalability issue centers on the capability of Meggie to accommodate more measurement files in an experiment and different sizes of the files. Since the memory management makes sure there is only one measurement file in the system memory, the user can add as many subjects to an experiment as there is available disk space where the working directory is set. Although, the size of the measurement file may vary depending on the length of the measurement and may exceed the amount of memory in the system running Meggie. Preloading the file limits the size of the measurement file in that case. Setting preload as false would prevent this problem in case of too big files, but according to the customer representative, the lengths of the measurements are short enough to support preloading.

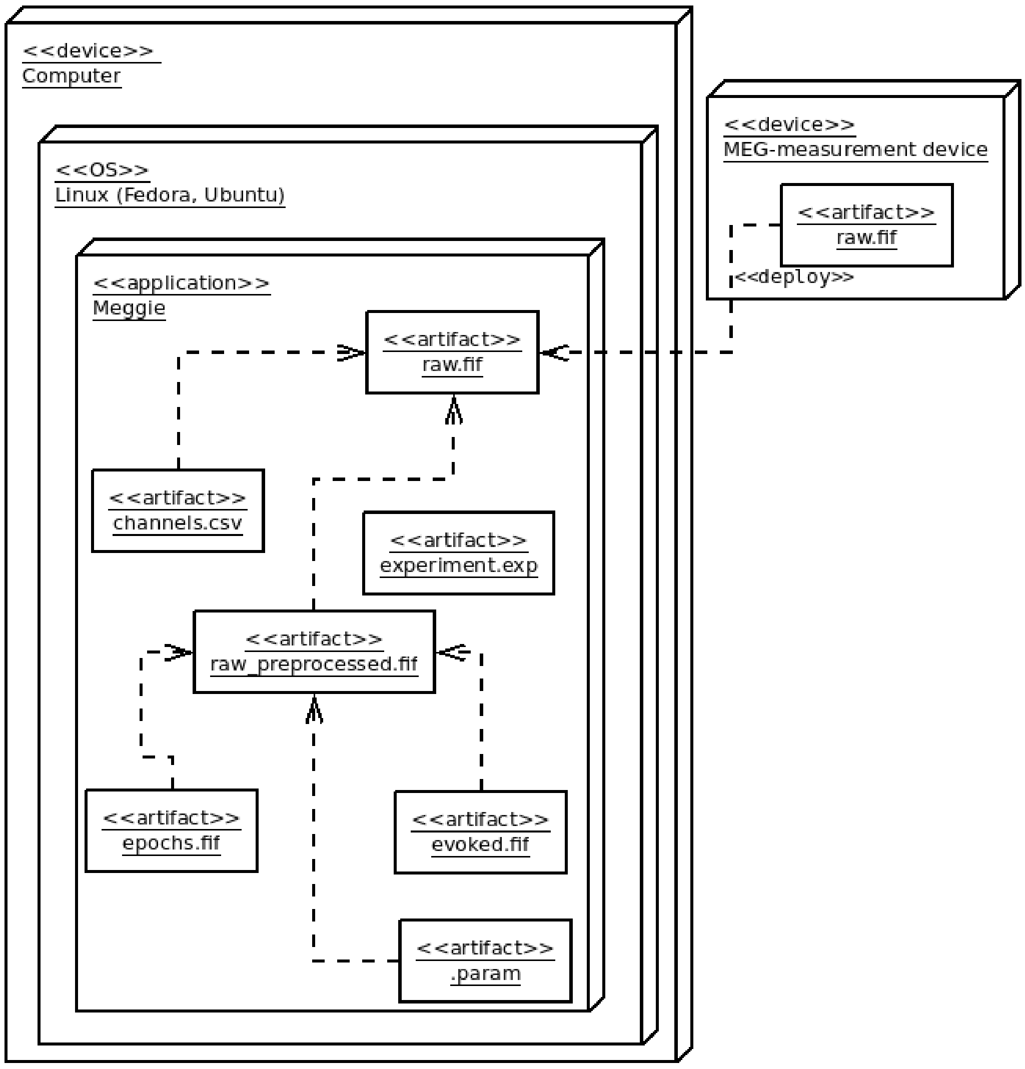


Figure 17: Deployment diagram of Meggie.

6.5 Scenarios

Scenarios are used to show the association between the elements in the four different views, and they are considered as the abstractions of the most important requirements (Kruchten 1995). Sufficient information from the scenarios can be captured with interaction diagrams (Booch 2006). They are quite similar to the sequence diagrams in section 6.1, but provide more detail about how the classes interact with each other. Anyway, there will be no UML diagrams for the scenarios, because the only processing unit is Meggie with its components, which would result in interaction diagrams rather similar to the sequence diagrams.

Only the major events and interactions should be captured at the scenario level of analysis (Booch 2006). Therefore, the aim of this section is to bring up the most critical components of the scenarios on a development level perspective. The main scenario for the development was 'Enable batch processing for the removal of EOG/ECG artefacts' which was then divided into several subscenarios that were easier to elicit into requirements. The selected important subscenarios that will be presented here are 'Add a subject', 'Remove a subject', 'Activate a subject' and 'Calculate ECG/EOG projections for multiple subjects in one batch process'. The two other scenarios mentioned earlier in chapter 5 were left out of this section, because they had no impact on the architecture.

The 'Add a subject' scenario suggests that there should be a simple class, Experiment, that is able to store multiple instances of a Subject class in it. This lead to the creation of the Subject class as seen in the sequence diagram in figure 2b. Now, if the user adds several subjects into an existing experiment, an instance of the Subject class is created for each subject and those instances are stored into an instance of the Experiment class. This makes it simple to call them for further uses, including the working_file attribute within the Subject class that stores the raw.fif measurement file. All in all, this enables easier implementation for the rest of the scenarios that were implemented. The creation of the Subject class also resulted in the creation of Epoch and Evoked classes for easier tracking of the epoch collection and evoked data files of particular subjects.

Adding a subject creates a folder that stores the subject data under the currently open experiment folder. This enables easier controllability of a single subject data. The scenario

’Remove a subject’ benefits from this by making it simple to remove the subject folder and its contents under the experiment tree. The removal of the attributes and instances considering that subject must be handled in the code level too. The instance that was created for the added subject is also deleted from the list of subjects in the runtime instance of the experiment. After the removal, the experiment instance is then serialized in order to match with its current state.

The decision was to have only one subject viewable on the main window at a time. Therefore, there was a need to somehow separate the view between the different subjects, which lead to the scenario ’Activate a subject’. This implies that the user should be able to choose a subject from the list of added subjects in Meggie and then activate it to be able to complete actions for it. It is possible to keep on track with the activate subject by using the .pro settings file that is created by serializing the runtime instance of the experiment that also stores the currently active subject name. The .pro file is saved under the experiment folder. The file is unpickled every time the experiment is loaded and an experiment instance is created to correspond with the received attributes, including the active subject name. The main window is then initialized to represent the state of the previously active subject.

The existence of the Subject instances supports the ’Enable batch processing for the removal of EOG/ECG artefacts’ scenario, because the Subject class holds dictionaries for the ECG and EOG calculation parameters. When completing the batch process, the parameters can be called from the Subject instances as seen in the sequence diagram in figure 14. The implementation for the batch process was simply chosen to be sequential instead of parallel. A parallel calculation of several preloaded measurement files would easily consume all the memory of the system running Meggie. The implementation of the sequential calculation wasn’t too complicated after the subject activation functionality was completed. The basic idea is that Meggie activates one subject at a time of the subjects that are added to the calculation and completes the calculation for that subject before continuing to the next one.

All the scenarios needed a UI level implementations to allow the user to complete the needed actions. The activity diagram of Meggie in figure 15 includes an area ’Experiment handling’ that shows all the actions the user is able to complete considering the scenarios.

6.6 Architectural Changes

During the architecture evaluation, some of the weakest design solutions were reimplemented. The biggest change to the architecture will be inspected in this section.

The subject creation and activation sequence was, perhaps, the weakest design solution that occurred during the evaluation. Since there was a lot of code in the UI classes to handle the creation and activation, the whole sequence was confusing enough to take it under the development again, without ignoring the fact that the solution was far from a model-view-controller (MVC) architecture.

There were several similarly named methods in the Experiment class that had shuffled purposes. Also, some of the attributes in the class were only used in confusing ways and they had to be set in a strict order to make them work correctly. Direct calls from the subject creation dialog and subject activation button of the main window to the Experiment class, without transferring the calls through the MainWindowMain class, got rid of the mentioned problems and allowed the removal of unnecessary code from the UI classes.

7 Implementation

This chapter covers the implementation of the optimization of the preprocessing steps in Meggie which is the main goal of this thesis. There is a view to the current state of the application and information about the utilized tools and test data. Also, the requirements that supported the development are covered here.

7.1 Tools

This section presents the tools used for the development phase. The utilized external software were MNE-Python and MaxFilter. Although MaxFilter hasn't been available for testing, it will be shortly introduced here in order to provide an understanding of the functionalities that Meggie should cover in its final state.

7.1.1 MNE

MNE is an academic software package for processing MEG and EEG (electroencephalography) data. The package includes tools for all phases of MEG and EEG data processing. These tools are provided as MNE-C (compiled C code), MNE-Matlab and MNE-Python. The Matlab toolbox facilitates access to the FIF (functional image file) format data files employed in Meggie and enables development of custom analysis tools based on the intermediate results computed with the MNE tools. The FIF format can store any type of information in a single file. (Gramfort et al. 2014)

7.1.2 MaxFilter

The other applied software package MaxFilter has been implemented in Meggie, but due to its closed source state there has been no testing possibility. It's intended to be used with Elekta Neuromag MEG products in suppressing magnetic interferences coming from inside and outside of the sensor array, in reducing measurement artefacts, in transforming data between different head positions, and in compensating disturbances due to head movements

(Elekta 2006). From the MaxFilter methods, Meggie should utilize expansion origin setting, expansion order setting, bad channels exclusion, data skipping, SSS/tSSS filtering, motion compensation and head position calculation.

7.2 Test Data

The MNE software is accompanied by a sample dataset (sample_audvis_raw.fif) which is recorded from one subject with combined M/EEG conducted at the Martinos Center of Massachusetts General Hospital. The data was recorded with a Neuromag VectorView system (Elekta Oy, Helsinki, Finland), see section 4.1 for more detail. The subject was shown auditory and visual stimuli in a random sequence with an added smiley face to which the subject reacted with a button press. The dataset can be used as a standard validation dataset for M/EEG methods, which also favors reproducibility of results. The dataset includes responses to left-ear auditory stimulus, right-ear auditory stimulus, left visual field stimulus, right visual field stimulus, the smiley face and the button press. (Gramfort et al. 2014)

7.3 Requirements

Realization of the requirements is based on the scenarios set by the customer representative of Meggie, original requirements specification of Hoksotin project and collected TODO-list from the further development process.

7.3.1 Batching

Requirements to enable the batch processing:

1. **The user must be able to add subjects to an active experiment.** Experiment class handled a single subject data before the development phase. To enable multiple subjects, a new class called Subject was created to handle the subject data. A list was added to the Experiment class to store multiple Subject instances in it. A new subject dialog was also created for the user to be able to add subjects to an active experiment and a button for opening the dialog. The experiment creation dialog now includes all

but the subject measurement file browser from the old dialog. The experiment needs to be created before the subjects can be added.

2. **The user must be able to choose multiple subjects at once to be added to an active experiment in one batch process.** A file browser for multiple selection was added on the newest subject dialog. If there are multiple files chosen by the user, Meggie reads them one at a time and saves them into the active experiment folder. The last one saved will also be set as the active subject.
3. **The user must be able to view added subjects of an active experiment.** A separate list was added to the mainwindow to display the added subjects of an active experiment.
4. **The user must be able to remove subjects from an active experiment.** The user can select a subject from the list of subjects and remove the selected subject with a new 'Remove subject/subjects'-button. The main window is initialized to clear all the fields corresponding to the removed subject and the folder that stored the subject files is also removed under the experiment hierarchy. The instance of the Subject class that stored information of the removed subject is also removed from the subject list in the currently active Experiment instance.
5. **A subject must be set as an active subject when it's added to an active experiment.** The reason for this requirement is that when adding a new subject, the measurement file of the subject must be read and added into the system memory to save it under the currently active experiment's directory. This leads to a huge memory consumption, if there already is a measurement file of an active subject in the memory. Therefore, adding a new subject triggers the subject activation sequence that initializes the main window with the newly activated subject info and removes the previous subject file from the system memory. More information about the memory management can be seen in section 7.4.
6. **The user must be able to activate any subject of an active experiment.** The user can activate a subject from the list of subjects with a new 'Activate subject/subjects'-button.
7. **There must be only one active subject at a time.** Again, the reason for this requirement is the memory management problem, which was explained in the requirement

no. 5. Also, the mainwindow is made to display info for one subject only. Activating a subject triggers the activation sequence that was also explained in requirement no. 5.

8. **Activating a subject must display its state as it was previously left when processing the subject.** This is a default save/load feature and it's done automatically according to the user's actions. The pickle module is used for serializing the Experiment class. Pickling is done every time when the files are saved under the active experiment directory. This enables loading the subjects with the most recently processed files.
9. **The user must be able to view metadata of an activated subject.** There are fields on the mainwindow which display the metadata of an active subject. They are updated every time a subject is activated.
10. **The user must be prevented to open the measurement data processing tabs unless there is an activated subject.** The reason for this is to guide the user to prevent dull actions. The processing tabs are disabled as long as a subject is activated. The functionalities and UI of Meggie changed and in the newest version the requirement is not implemented.
11. **The user must be able to perform computation for preprocessing steps (EOG/ECG) for multiple measurement files at once.** Initially, the batch processing functionality for the practical work of the thesis was aimed at the removal of EOG artefacts. However, after it was clear that the removal of ECG artefacts shouldn't require a lot of extra work, it was decided to take it under the development too. An area for batch processing was added to both EOG and ECG dialog. A radiobutton was also added to both dialogs to enable/disable the batch process area. The area includes a list of the subjects that are currently in the active experiment. After selecting a subject on the list, the user can then modify its parameters on the EOG/ECG dialog. The 'Apply'- and 'Apply to all'-button save the parameters to the selected subject or to all of the subjects on the list. The computation is then executed for all the subjects that the user wants to include. The user can remove subjects from the list to exclude them from the computation.
12. **The user must be able to view metadata of all the subject files that he/she browsed with the file browser in the subject dialog.** The 'Add subject' dialog includes a list of the browsed files which can be selected one at a time to enable the 'Show file info'

button.

13. **Opening an existing experiment must activate the subject that was active last time when the experiment was closed.** The active_subject attribute of an active experiment is saved into the settings file (.pro) by serializing the operational Experiment instance every time a subject is activated. This allows Meggie to load the settings file, including the active_subject attribute, from the root folder of the experiment that is opened.

7.3.2 Preprocessing

Requirements listed to improve and correct the funtionality of the preprocessing steps:

1. **The user must be able to distinguish imported epochs files from the epochs created from the currently active subject.** It's important to see whether the epochs are created from the current working file or from any other file. The functionality is a bit twisted, because the epochs that don't have parameter files under the epochs directory can be seen written with red colour on the epochs list on the main window. This is because the epochs file doesn't have info of the source file it was created from, but a short description in case the author of the measurement had written it there.
2. **The user must be able to see where an imported epochs file is from, e.g. metadata.** There is a field for the description of the subject written by the author of the measurement. This seemed to be the only way to gain some information from the source file.
3. **mne_browse_raw must be replaced with the MNE raw.plot function.** Not implemented. The external library MNE is constantly under development and the new features are deployed which lead to this requirement. This required only a single line of code. Anyhow, after implementing this functionality the customer representative wanted to change back to the mne_browse_raw function.
4. **Magnitude spectrum must be replaced with the MNE raw.plot_psds -function.** Not implemented. Delayed until further development.
5. **The user must be able to easily realize in which order to fill the fields on epochs creation dialog.** The fields in epoching dialog were reorganized for a more intuitive usability.

6. **The user must be able to see which parameters were used in different phases of preprocessing.** The parameters for calculating EOG and ECG projections are saved under the subject folder and the user can view them by opening the EOG or ECG dialog and selecting a subject on the batch list. The epoch collection parameters are shown on the epoching tab after selecting a collection on the epoch collections list.

7.3.3 Epoching and Averaging

Requirements listed to improve and correct the funtionality of the epoching and averaging steps:

1. **The user must be able to understand in which order to complete the operations on averaging tab.** Not implemented. Simply forgotten and the customer didn't give clear guidance to this requirement.
2. **The user must be able to calculate statistics of evoked responses.** Not implemented. Delayed until further development.
3. **The user must be able to save the statistics of evoked responses in csv-format.** Not implemented. Delayed until further development.
4. **The evoked responses must be saved automatically when created.** The button that creates evoked responses also saves them under the subject folder, so the user is not responsible for that.

7.3.4 Nonfunctional Requirements

These requirements are mostly listed to cover issues with the memory management.

1. **The code that generates the UI must be separated from the code that forms the model.** This was followed all the way through the development phase and most of the old code that didn't belong to the UI classes was moved to the model classes.
2. **Measurement data files must be preloaded into the system memory before the calculations.** Usually programs that process large files read the data into memory stream to prevent huge memory requirements. Although, in Meggie we chose to preload the data into memory for faster data manipulation with the functionalities provided by

MNE. See the Memory management section 7.4 for details.

3. **There must be only one measurement file in the system memory at once.** Implemented. See the Memory management section 7.4 for details.

7.4 Memory Management

Memory management of Meggie utilizes the mechanisms of the Python garbage collector (gc) module. All the larger files, such as source, epoch collection and evoked files, require management. Subject, Epoch and Evoked classes are used to store references to those files. The references are simply set to None when the memory the large files consume needs to be released again. The garbage collector triggers when there are no references to an object, thus releasing the memory allocated to the object. The memory usage in the testing system was checked using command 'free' in Terminal and occasionally the number of references to objects was tracked with the functionalities of the gc module. There was only one case in which the garbage collector didn't automatically release the memory and the gc.collect() function had to be called manually.

7.4.1 Main Cases

The main decision considering memory management was to read all the files of a single subject at once into the system memory. Those files include the current working file of an active subject, epoch collection files created from the active subject and evoked datasets averaged from the epoch collection files. This lead to several cases that needed more attention: activating a subject, removing an active subject, removing a non-active subject, opening an existing experiment and creating a new experiment. Also, there was a need for batch processing functionality which lead to two unique cases: adding several subjects into an experiment at once and removing EOG/ECG artefacts from multiple subjects in one batch process.

Activating a subject checks if there already is an active subject with its measurement files in the memory. If so, the files of the active subject must be released from the memory before the other subject is activated. This is the subject deactivation sequence, which applies the release_memory() function in the Experiment class, which only sets the references to the measurement files as None, but leaves the objects alive.

Removing an active subject destroys the Subject instance which results in removing the epoch collection and evoked files as well, because the Subject class stores Epoch and Evoked instances in it. Removing a non-active subject applies the same idea, but differs in that the measurement file references are already set to None and their memory release is handled beforehand. The objects are destroyed in the same way as in the removal of an active subject.

Opening an existing experiment and creating a new experiment are handled simply by setting the Experiment instance reference in the MainWindow class to the newly created or opened Experiment instance. This leads to having no references to the previously handled Experiment instance, if there was one, in the same session. Since the Experiment class stores the Subject instances, they are also removed in the process and no further actions are needed to release their memory usage.

When several subjects are added into an experiment at once they must be read into the system memory to get vital data for further processing steps (e.g. channel list). Therefore, it's not possible to copy the measurement file into the folder without reading the file, although it would save some time.

To remove EOG/ECG artefacts from multible subjects in one batch process, the only possible way seemed to be by calculating one subject at a time, since several measurement files may consume all the memory of the system. After the calculation is finished for one subject, the memory consumed by the subject is released before continuing to the next one. When the last subject is calculated, it is set as the active subject, since its measurement file is already in the memory. This avoids excessive reading of another subject file and shortens the wait time for the user.

7.4.2 Problems in Memory Management

There were many problematic cases during the memory management, but the three cases here are good examples of what it was all about. The last case was quite illogical and the solution was created by randomly testing different ways to implement it.

An instance of an EpochParameterDialog was created by adding a reference to an active subject measurement file to its attributes. A reference to this instance was then set in the MainWindow class attribute self.epochParameterDialog, which, after deleting the active subject, remained in the MainWindow. This prevented Python garbage collection to collect also the measurement file from memory when deactivating the subject. This was fixed by not setting the measurement file reference to the EpochParameterDialog instance, but instead getting the measurement file from the active subject in the EpochParameterDialog code.

Earlier, the Experiment class set Experiment instance to Caller class to help with setting attributes to the Experiment. This resulted in having the previously open experiment and it's active subject, including the measurement files, in the system memory after opening or creating a new experiment. By removing the setting of the Experiment attribute to Caller class, we can just set the current Experiment as None and call the Python garbage collector to release the memory to the OS. For unknown reason, in this case the garbage collector didn't automatically release memory and had to be called manually.

Completing measurement file calculations inside a for loop seemed not to release the memory to the OS from the previously handled file, even if the references were removed from the file and the garbage collector was called manually. This lead to the creation of a separate method that handles the calculation for one subject only and this method is called inside a for loop from another method. Every time the calculation method finishes, the garbage collector releases the memory to the OS from the previously handled file.

8 Conclusions

Meggie is a MEG measurement data preprocessing, visualizing and analyzing application, and currently (spring 2015) it's close to the beta phase. The development started from scratch in the beginning of 2013 and is still ongoing. The application includes the most of the important features, but insufficient testing doesn't guarantee their correctness. Although, several important functionalities are missing. Meggie should be soon available for testing in the MEG laboratory located in the University of Jyväskylä. The overall goal of the application is to simplify the steps needed to complete unique MEG data processing pipelines. The graphical user interface should allow even uninitiated users to be able to understand the required steps.

The functionalities center on the external package MNE that provides extensive set of methods to process MEG data. The methods implemented in Meggie include physiological artefact rejection, filtering, epoching, data averaging and source analysis. The other important package MaxFilter, which would be used in the preprocessing phase, is not included due to its closed source state. Therefore, the data analyzed with Meggie has to be preprocessed in advance.

Further development should be focused on adding the missing features, testing the existing ones and fixing the errors. The major missing or incomplete features are the MaxFilter component, evoked stats calculation dialog and several source space analysis methods. The shortcomings in the architecture evaluation chapter should be taken into consideration, too. The development tools and external packages must be up-to-date, since their development is ongoing and the updated functionalities make the application run with a better performance. Also, there were no usability tests to verify if the steps are intuitive for the user, which should add a reasonable amount of work to that area. However, since several important features are missing, it's advisable to work on the usability after the features are implemented.

This thesis presented the application at its current state in the spring 2015, after the completion of the practical phase. The phase included the development of several important features and a glimpse to that phase is provided in the form of the implemented requirements with their short explanations. The theoretical phase went through the architecture of the application with diagrams to create an extensive view and explain the chosen quality attributes.

The contributional aspect of this thesis is mostly for the developers, who are willing to further develop the application in order to optimize the analysis steps of MEG measurement data in the MEG laboratory in Jyväskylä. By going through the MEG and user interface presentation chapters, even uninitiated persons might be able to execute a simple analysis pipeline. Therefore, the thesis may be used as a replacement for manual. In addition, general pipelines are provided to the analysis of MEG data, which might interest the developers in the field of computational neuroscience. The field that is rapidly growing and new scientific software is constantly needed, as well as documentations like this.

Bibliography

Aliranta, K., and J. Pesonen. 2013. “Meggie Application Report”. Accessible via the author of the thesis, number 1.0.0 ().

Booch, Grady. 2006. *Object Oriented Analysis & Design with Application.* Pearson Education India. http://kmvportal.co.in/Course/OOAD/object-oriented-analysis-and-design-with-applications-2nd-edition.pdf.

David, Olivier, James M. Kilner, and Karl J. Friston. 2006. “Mechanisms of evoked and induced responses in MEG/EEG”. *NeuroImage* 31 (4): 1580–1591. ISSN: 1053-8119. http://www.sciencedirect.com/science/article/pii/S1053811906001170.

Dijk, Hanneke van, Jan-Mathijs Schoffelen, Robert Oostenveld, and Ole Jensen. 2008. “Prestimulus Oscillatory Activity in the Alpha Band Predicts Visual Discrimination Ability”. *The Journal of Neuroscience* 28 (8): 1816–1823. http://www.jneurosci.org/content/28/8/1816.abstract.

Elekta. 2006. *MaxFilter User’s Guide.* http://www.megwiki.org/images/a/aa/MaxFilter-2.0.pdf.

Gewaltig, Marc-Oliver, and Robert Cannon. 2014. “Current Practice in Software Development for Computational Neuroscience and How to Improve It”. *PLoS Comput Biol* 10, number 1 (): e1003376. http://dx.doi.org/10.1371/journal.pcbi.1003376.

Gramfort, Alexandre, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen, and Matti S. Hämäläinen. 2014. “{MNE} software for processing {MEG} and {EEG} data”. *NeuroImage* 86:446–460. ISSN: 1053-8119. http://www.sciencedirect.com/science/article/pii/S1053811913010501.

Gross, J., S. Baillet, G R Barnes, R N Henson, A Hillebrand, O. Jensen, K. Jerbi, et al. 2013. “Good practice for conducting and reporting MEG research”. *Neuroimage* 65 (): 349–363. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3925794/.

Gross, J., and A. A. Ioannides. 1999. "Linear transformations of data space in MEG". *Physics in Medicine and Biology* 44 (8): 2081. http://stacks.iop.org/0031-9155/44/i=8/a=317.

Hansen, Peter, Morten Kringelbach, and Riitta Salmelin. 2010. "MEG: An Introduction to Methods". http://dx.doi.org/10.1093/acprof:oso/9780195307238.001.0001.

Hari, Riitta, Lauri Parkkonen, and Cathy Nangini. 2010. "The brain in time: insights from neuromagnetic recordings". *Annals of the New York Academy of Sciences* 1191 (1): 89–109. ISSN: 1749-6632. http://dx.doi.org/10.1111/j.1749-6632.2010.05438.x.

Hevner, A., and S. Chatterjee. 2010. "Design Research in Information Systems". http://dx.doi.org/10.1007/978-1-4419-5653-8_2.

Hämäläinen, Matti, Riitta Hari, Risto J. Ilmoniemi, Jukka Knuutila, and Olli V. Lounasmaa. 1993. "Magnetoencephalography—theory, instrumentation, and applications to noninvasive studies of the working human brain". *Rev. Mod. Phys.* 65, number 2 (): 413–497. http://link.aps.org/doi/10.1103/RevModPhys.65.413.

Ille, Nicole, Patrick Berg, and Michael Scherg. 2002. "Artifact Correction of the Ongoing EEG Using Spatial Filters Based on Artifact and Brain Signal Topographies". *Journal of Clinical Neurophysiology* 19 (2): 113–124. http://journals.lww.com/clinicalneurophys/Fulltext/2002/03000/Artifact_Correction_of_the_Ongoing_EEG_Using.2.aspx.

Jolliffe, Ian. 2005. *Principal Component Analysis.* John Wiley & Sons, Ltd. ISBN: 9780470013199. http://dx.doi.org/10.1002/0470013192.bsa501.

Kazman, R., G. Abowd, L. Bass, and P. Clements. 1996. "Scenario-based analysis of software architecture". *Software, IEEE* 13, number 6 (): 47–55. ISSN: 0740-7459. doi:10.1109/52.542294.

Kruchten, P. B. 1995. "The 4+1 View Model of architecture". *Software, IEEE* 12, number 6 (): 42–50. ISSN: 0740-7459. doi:10.1109/52.469759.

Laskaris, N. A., S. Fotopoulos, and A.A. Ioannides. 2004. “Mining information from event-related recordings”. *Signal Processing Magazine, IEEE* 21, number 3 (): 66–77. ISSN: 1053-5888. doi:10.1109/MSP.2004.1296544.

Mathewson, Kyle E., Gabriele Gratton, Monica Fabiani, Diane M. Beck, and Tony Ro. 2009. “To See or Not to See: Prestimulus α Phase Predicts Visual Awareness”. *The Journal of Neuroscience* 29 (9): 2725–2732. http://www.jneurosci.org/content/29/9/2725.abstract.

Parra, Lucas C., Clay D. Spence, Adam D. Gerson, and Paul Sajda. 2005. “Recipes for the linear analysis of {EEG}”. *NeuroImage* 28 (2): 326–341. ISSN: 1053-8119. http://www.sciencedirect.com/science/article/pii/S1053811905003381.

Ploner, Markus, Joachim Gross, Lars Timmermann, Bettina Pollok, and Alfons Schnitzler. 2006. “Oscillatory activity reflects the excitability of the human somatosensory system”. *NeuroImage* 32 (3): 1231–1236. ISSN: 1053-8119. http://www.sciencedirect.com/science/article/pii/S1053811906006641.

“PyQt4 Reference Guide”. 2014. Visited on March 31, 2015. http://pyqt.sourceforge.net/Docs/PyQt4/index.html.

Romei, Vincenzo, Joachim Gross, and Gregor Thut. 2012. “Sounds Reset Rhythms of Visual Cortex and Corresponding Human Visual Perception”. *Current Biology* 22 (9): 807–813. ISSN: 0960-9822. http://www.sciencedirect.com/science/article/pii/S0960982212003120.

Schlögl, A., C. Keinrath, D. Zimmermann, R. Scherer, R. Leeb, and G. Pfurtscheller. 2007. “A fully automated correction method of {EOG} artifacts in {EEG} recordings”. *Clinical Neurophysiology* 118 (1): 98–104. ISSN: 1388-2457. http://www.sciencedirect.com/science/article/pii/S1388245706014313.

Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. 1997. “Kernel principal component analysis”. In *Artificial Neural Networks — ICANN’97,* edited by Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, 1327:583–588. Lecture Notes in Computer Science. Springer Berlin Heidelberg. ISBN: 978-3-540-63631-1. http://dx.doi.org/10.1007/BFb0020217.

Taulu, S., M. Kajola, and J. Simola. 2003. “The Signal Space Separation method”. *(Biomed. Tech.), (To appear in Proceedings of 14th Conference of the International Society for Brain Electromagnetic Topography (ISBET)),* number 48 (). /brokenurl#arXiv:physics/0401166.

Taulu, Samu, Matti Kajola, and Juha Simola. 2004. “Suppression of Interference and Artifacts by the Signal Space Separation Method”. *Brain Topography* 16 (4): 269–275. ISSN: 0896-0267. http://dx.doi.org/10.1023/B:BRAT.0000032864.93890.f9.

Uusitalo, M. A., and R. J. Ilmoniemi. 1997. “Signal-space projection method for separating MEG or EEG into components”. *Medical and Biological Engineering and Computing* 35 (2): 135–140. ISSN: 0140-0118. http://dx.doi.org/10.1007/BF02534144.

Wallstrom, Garrick L., Robert E. Kass, Anita Miller, Jeffrey F Cohn, and Nathan A Fox. 2004. “Automatic correction of ocular artifacts in the EEG: a comparison of regression-based and component-based methods”. *International Journal of Psychophysiology* 53 (2): 105–119. ISSN: 0167-8760. http://www.sciencedirect.com/science/article/pii/S0167876004000510.

Weidenhaupt, K., K. Pohl, M. Jarke, and P. Haumer. 1998. “Scenarios in system development: current practice”. *Software, IEEE* 15, number 2 (): 34–45. ISSN: 0740-7459. doi:10.1109/52.663783.

Wieger, Karl. 2013. *Software Requirements 3.* Redmond: Microsoft Press,U.S. ISBN: 9780735679665 0735679665. http://www.worldcat.org/search?qt=worldcat_org_all&q=0735679665.

Xia, H., A. Ben-Amar Baranga, D. Hoffman, and M. V. Romalis. 2006. “Magnetoencephalography with an atomic magnetometer”. *Applied Physics Letters* 89, 211104 (21): pages. http://scitation.aip.org/content/aip/journal/apl/89/21/10.1063/1.2392722.