Module Guide for EEGSourceLocalizer

Leila Mousapour

December 17, 2020

1 Revision History

Date	Version	Notes
10 Dec 2020	1.0	Frist version

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
UC	Unlikely Change
SL	Source Localization
SRS	Software Requirements Specification
EEGSourceLocalizer	EEG Source Localization Software

Contents

List of Tables

List of Figures

3 Introduction

Decomposing a system into modules is a commonly accepted approach to developing software. A module is a work assignment for a programmer or programming team (Parnas et al., 1984). We advocate a decomposition based on the principle of information hiding (Parnas, 1972). This principle supports design for change, because the "secrets" that each module hides represent likely future changes. Design for change is valuable in SC, where modifications are frequent, especially during initial development as the solution space is explored.

Our design follows the rules layed out by Parnas et al. (1984), as follows:

- System details that are likely to change independently should be the secrets of separate modules.
- Each data structure is implemented in only one module.
- Any other program that requires information stored in a module's data structures must obtain it by calling access programs belonging to that module.

After completing the first stage of the design, the Software Requirements Specification (SRS), the Module Guide (MG) is developed (Parnas et al., 1984). The MG specifies the modular structure of the system and is intended to allow both designers and maintainers to easily identify the parts of the software. The potential readers of this document are as follows:

- New project members: This document can be a guide for a new project member to easily understand the overall structure and quickly find the relevant modules they are searching for.
- Maintainers: The hierarchical structure of the module guide improves the maintainers' understanding when they need to make changes to the system. It is important for a maintainer to update the relevant sections of the document after changes have been made.
- Designers: Once the module guide has been written, it can be used to check for consistency, feasibility and flexibility. Designers can verify the system in various ways, such as consistency among modules, feasibility of the decomposition, and flexibility of the design.

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 specifies the connections between the software requirements and the modules. Section 7 gives a detailed description of the modules. Section 8 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules. Section 9 describes the use relation between modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The specific hardware on which the software is running.

AC2: The format of the initial input data.

AC3: The specification parameters.

AC4: The algorithm used to calculate the covariance of the EEG data.

AC5: The method of creating the head model (which currently is Boundary Element Method (BEM) and the parameters (conductivity of the brain tissues).

AC6: The source model parameters can change (the resolution of the brain segmentation, the distribution of dipoles etc.).

AC7: The algorithm used to perform source localization.

AC8: The method of plotting (plot the source power map on a 3D brain surface or sliced MRI etc.).

AC9: The format of the final output data.

AC10: The flow of main program.

AC11: The choice of data structures used for storing and manipulating the data.

AC12: The choice of MATLAB built-in algorithms and toolboxes used in the software.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen).

UC2: Average refereeing EEG data will not be included in this software. (it is required for EEG data to be re-referenced against the average of all the electrodes (AG) when intended to be used in source localization).

UC3: The source model would be the whole brain volume and this software will not include cortical surface (cortical sheet) source model.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented (Not including the software decision modules).

M1: Hardware-Hiding Module

M2: Input Parameters Module

M3: Output Format Module

M4: Plotting Module

M5: Covariance Calculator Module

M6: Head Model Module

M7: Lead Filed Module

M8: Source Localization Module

M9: Control Module

M10: Specification Parameters Module

M11: Various Algorithems Module

M12: Data Structure Module

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

Level 1	Level 2
Hardware-Hiding	
Behaviour-Hiding	Input Parameters
	Output Format
	Plotting
	Covariance Calculator
	Head Model
	Lead Filed
	Source Localization
	Control Module
	Specification Parameters Module
Software Decision	Matrix/Cell/Structure built-in MATLAB Data Structures
	Various built-in MATLAB and Fieldtrip toolbox algorithms
	Plotting

Table 1: Module Hierarchy

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The *Secrets* field in a module decomposition is a brief statement of the design decision hidden by the module. The *Services* field specifies *what* the module will do without documenting *how* to do it. For each module, a suggestion for the implementing software is given under the *Implemented By* title. If the entry is *OS*, this means that the module is provided by the operating system or by standard programming language libraries. *EEGSourceLocalizer* means the module will be implemented by the EEGSourceLocalizer software.

Only the leaf modules in the hierarchy have to be implemented. If a dash (–) is shown, this means that the module is not a leaf and will not have to be implemented.

7.1 Hardware Hiding Modules (M1)

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented By: OS

7.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviours.

Services: Includes programs that provide externally visible behaviour of the system as specified in the software requirements specification (SRS) documents. This module serves as a communication layer between the hardware-hiding module and the software decision module. The programs in this module will need to change if there are changes in the SRS.

Implemented By: –

7.2.1 Input Parameter Module (M2)

Secrets: The format and structure of the input data.

Services: Converts the input data into the data structure used by the other modules and check if the conditions are met.

Implemented By: EEGSourceLocalizer

7.2.2 Output Format Module (M3)

Secrets: The format and structure of the output data.

Services: Converts the output results into the data structure used by the input parameters module and check if the conditions are met.

Implemented By: EEGSourceLocalizer

7.2.3 Plotting Module (M4)

Secrets: The format and style of plotting the output source power.

Services: Slice the MRI image provided by the user and interpolate the power computed for every source point and plot the power on the sliced MRI image.

Implemented By: EEGSourceLocalizer

7.2.4 Covariance Calculator Module(M5)

Secrets: The algorithm used to compute the covariance of the EEG data.

Services: Computes the covariance of the averaged EEG data (averaged over all trials) which will be used in source localization phase.

Implemented By: EEGSourceLocalizer

7.2.5 Head Model Module (M6)

Secrets: The algorithm used to create the head model.

Services: Take the MRI and segment it to find the boundaries between the 3 tissues types: skull, scalp and brain with Boundary Element Method (BEM). Finally, creates a head model based on these 3 surfaces and the corresponding tissue conductivity.

Implemented By: EEGSourceLocalizer

7.2.6 Lead Field Module (M7)

Secrets: The source distribution and algorithm used to calculate the forward solution.

Services: It first align the electrodes with the head model and then generates the source model (a structure containing the location of the sources by segmenting brain volume which its boundary is in the head model structure). Afterwards, it computes the forward model for all the dipole locations of the 3D source model.

Implemented By: EEGSourceLocalizer

7.2.7 Source Localization Module (M8)

Secrets: The source distribution and

Services: It first generates the source model (a structure containing the location of the sources by segmenting brain volume which its boundary is in the head model structure). Afterwards, it computes the forward model for all the dipole locations of the 3D source model.

Implemented By: EEGSourceLocalizer

7.2.8 Control Module (M9)

Secrets: Execution flow of EEGSourceLocalizer

Services: The main module of the software where it calls the different modules in the appropriate order from getting inputs to calculating outputs and plotting the results.

Implemented By: EEGSourceLocalizer

7.2.9 Specification Parameter Module (M10)

Secrets: The constant values used in the code.

Services: Stores all the constant values, including values mention in the table of specification parameters in the SRS document (constraints and conditions on the input/output values, other configurations and hyper parameters for different modules etc.), for being called where needed at different modules.

Implemented By: EEGSourceLocalizer

7.3 Software Decision Module

Secrets: The design decision based on mathematical theorems, physical facts, or programming considerations. The secrets of this module are *not* described in the SRS.

Services: Includes data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

7.3.1 Various Algorithms Module (M11)

Secrets: The choice of algorithm and what implementation to use (built-in MATLAB or external toolboxes)

Services: As there are several algorithms used in this software (including the covariance calculation, LCMV, BEM etc.), we are not specifying them separately. However, for each of these algorithms we need to determine in this document that what implementation is used in this software: For all the algorithms needed in the EEGSourceLocalizer, we have used the "Fieldtrip" toolbox implementation .

Implemented By: Fieldtrip

7.3.2 Data Structure Module (M12)

Secrets: The choice of data types for all the parameters

Services: Appropriately storing different values across the software. MATLAB built-in data structures including matrix, cell and structure are used.

Implemented By: MATLAB

8 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements and between the modules and the anticipated changes.

Req.	Modules
R1	M1, M2, M9
R2	M2, M10, M9
R3	M5, M2, M9
R4	M6, M5, M7, M8, M9
R5	M3, M4, M10, M9

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M2
AC3	M10
AC5	M6
AC6	M7
AC7	M8
AC4	M5
AC9	M3
AC10	M9
AC8	M4
AC12	M11
AC11	M12

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

In this section, the uses hierarchy between modules is provided. Parnas (1978) said of two programs A and B that A *uses* B if correct execution of B may be necessary for A to complete the task described in its specification. That is, A *uses* B if there exist situations in which the correct functioning of A depends upon the availability of a correct implementation of B. Figure 1 illustrates the use relation between the modules. It can be seen that the graph is a directed acyclic graph (DAG). Each level of the hierarchy offers a testable and usable subset of the system, and modules in the higher level of the hierarchy are essentially simpler because they use modules from the lower levels.

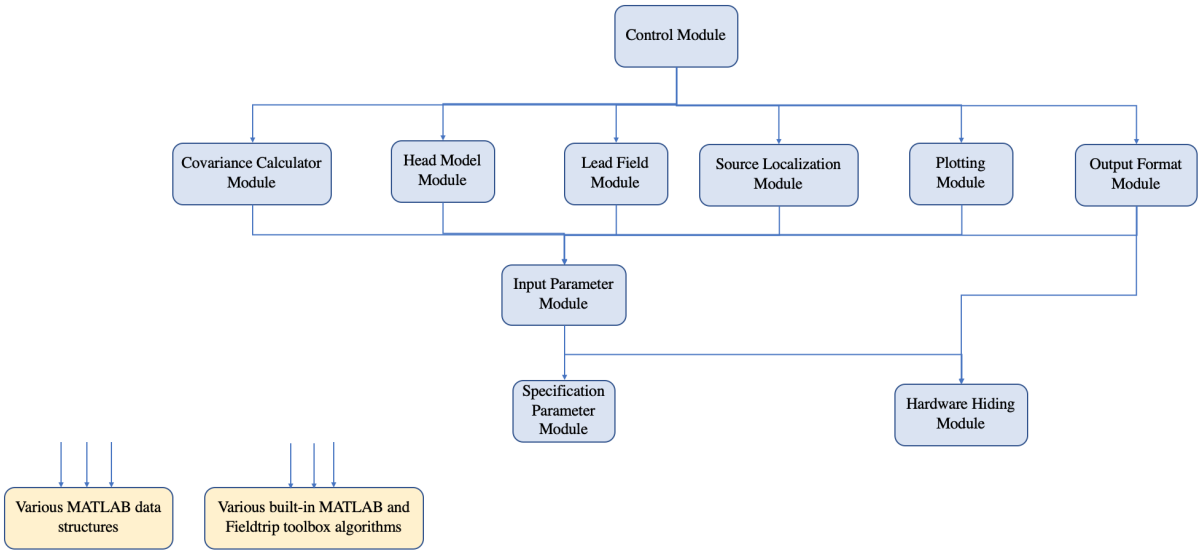

Figure 1: Use hierarchy among modules

References

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM*, 15(2):1053–1058, December 1972.

David L. Parnas. Designing software for ease of extension and contraction. In *ICSE '78: Proceedings of the 3rd international conference on Software engineering*, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. ISBN none.

D.L. Parnas, P.C. Clement, and D. M. Weiss. The modular structure of complex systems. In *International Conference on Software Engineering*, pages 408–419, 1984.