# Module Guide for Live Neuro

Bo Liang

April 11, 2025

# 1 Revision History

| Date | Version | Notes |
|---|---|---|
| March 17 2025 | 1.0 | initial draft |
| Apr 11 2025 | 1.1 | modify |

# 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 |
| SRS | Software Requirements Specification |
| TRF | Temporal Response Function |
| Live Neuro | Explanation of program name |
| UC | Unlikely Change |

# Contents


**1 Revision History** **i**

**2 Reference Material** **ii**

2.1 Abbreviations and Acronyms . . . . . ii

**3 Introduction** **1**

**4 Anticipated and Unlikely Changes** **2**

4.1 Anticipated Changes . . . . . 2

4.2 Unlikely Changes . . . . . 2

**5 Module Hierarchy** **2**

**6 Connection Between Requirements and Design** **3**

**7 Module Decomposition** **3**

7.1 Hardware Hiding Modules (M1) . . . . . 3

7.2 Behaviour-Hiding Module . . . . . 3

7.2.1 Input Format Module (M2) . . . . . 3

7.2.2 Data Processing Module(M3) . . . . . 4

7.2.3 Visualization Module(M4) . . . . . 4

7.3 Software Decision Module(M5) . . . . . 4

**8 Traceability Matrix** **4**

**9 Use Hierarchy Between Modules** **5**

**10 User Interfaces** **6**

**11 Design of Communication Protocols** **6**

**12 Timeline** **7**


# List of Tables


1 Module Hierarchy . . . . . 3

2 Connection Between Requirements and Design . . . . . 3

3 Trace Between Requirements and Modules . . . . . 5

4 Trace Between Anticipated Changes and Modules . . . . . 5

# List of Figures

1 Use hierarchy among modules . . . . . 6

# 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 (**?**). We advocate a decomposition based on the principle of information hiding (**?**). 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 **?**, 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 (**?**). 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

## 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:** Hardware platform (e.g., OS compatibility)

**AC2:** Input data format (e.g., MEG/EEG)

**AC3:** Visualization methods (e.g., 2D to 3D)

**AC4:** Interaction methods (e.g., click/slide/drag/zoom)

## 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(MEG/EEG data), Output: File(data image)).

**UC2:** Core TRF-based stimulus-response 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.

**M1:** Hardware-Hiding Module

**M2:** Input Format Module

**M3:** Data Processing Module

**M4:** Visualization Module

**M5:** TRF Calculation Module

the GUI is provided by the Dash Library in Visualization module, so we don't need an independent GUI module

| Level 1 | Level 2 | Module ID |
|---|---|---|
| Hardware-Hiding Module | Hardware-Hiding Module | M1 |
| Behaviour-Hiding Module | Input Format Module | M2 |
| | Data Processing Module | M3 |
| | Visualization Module | M4 |
| Software Decision Module | TRF Calculation Module | M5 |

Table 1: Module Hierarchy

# 6 Connection Between Requirements and Design

The system is designed to fulfill the functional and non-functional requirements specified in the SRS. The following table traces requirements to their corresponding modules:

| Requirement (SRS) | Module ID |
|---|---|
| R1: Interactive visualization plots | M1,M2,M3,M4 |
| R2: Cross-linked interactive plots | M4,M5 |

Table 2: Connection Between Requirements and Design

# 7 Module Decomposition

The system is divided into the following modules:

## 7.1 Hardware Hiding Modules (M1)

**Secrets:** Hardware implementation details.

**Services:** Interfaces with system hardware for data input and display.

**Implemented By:** OS libraries

## 7.2 Behaviour-Hiding Module

### 7.2.1 Input Format Module (M2)

**Secrets:** Data format transformations.

**Services:** Provides a unified interface for loading neuroimaging datasets from local or remote file systems. Supports reading standardized formats (.fif) via library wrappers. Wrap Python libraries like mne, numpy, or pandas while shielding higher-level modules from direct I/O code.

**Implemented By:** Live Neuro.

**Type of Module:** Library

#### 7.2.2 Data Processing Module(M3)

**Secrets:** Data Processing and Statistical index calculation

**Services:** Validates structure and format of incoming data (FIF, NumPy arrays) Checks for missing values, corrupted content, or structural inconsistencies Preprocesses and transforms raw input (e.g., filtering, normalization) Raises meaningful errors or warnings when validation fails

**Implemented By:** Live Neuro(NumPy/Pandas).

**Type of Module:** Library

#### 7.2.3 Visualization Module(M4)

**Secrets:** Plot rendering engine

**Services:** Generates interactive plots and links data across views.

**Implemented By:** Live Neuro (Plotly/Nilearn).

**Type of Module:** Library

### 7.3 Software Decision Module(M5)

**Secrets:** TRF model implementation

**Services:** Computes predicted neural dipole currents data.

**Implemented By:** Eelbrain (Python backend).

## 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, M3, M4 |
| R2 | M4, M5 |

Table 3: Trace Between Requirements and Modules

| AC | Modules |
|---|---|
| AC1 | M1 |
| AC2 | M2 |
| AC3 | M4 |
| AC4 | M4 |

Table 4: Trace Between Anticipated Changes and Modules

# 9 Use Hierarchy Between Modules

The system follows a layered architecture where lower-level modules (hardware and input) are used by higher-level computational and visualization modules.

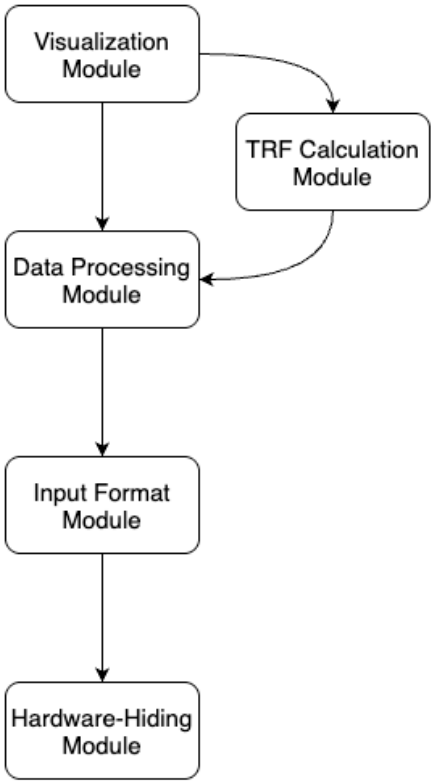


Figure 1: Use hierarchy among modules

# 10 User Interfaces

Live Neuro provides a user interface with features~~such as~~:

- Data loading from disk
- Data preprocessing and filtering
- Data plotting of MEG/EEG data
- Calculation of statistical index
- Linking between multiple plots.

# 11 Design of Communication Protocols

Live Neuro modules communicate via structured API calls, allowing for efficient data flow between processing and visualization components.

# 12 Timeline

The development schedule follows an agile approach, with iterative testing and refinement. A GitHub repository will be used for version control and issue tracking. The deadline for the Input Format Module, Data Processing Module, Visualization Module is March 18, March 22, and April 1, respectively.