Module Guide for Model–to–Brain Alignment for Speech Recognition

Xiao Shao

April 21, 2026

1 Revision History

Date	Version	Notes
March 25, 2026	1.0	Initial Draft
April 21, 2026	1.1	Update according to smith's feedback

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
API	Application Programming Interface
BIDS	Brain Imaging Data Structure
CLI	Command Line Interface
CV	Cross-Validation
DL	Deep Learning
HPC	High-Performance Computing
M	Module
MEG	Magnetoencephalography
MG	Module Guide
MIS	Module Interface Specification
mTRF	Multivariate Temporal Response Function
OS	Operating System
ROI	Region of Interest
SRS	Software Requirements Specification
TRF	Temporal Response Function
UC	Unlikely Change
WER	Word Error Rate
Model–to–Brain Alignment for Speech Recognition	Program name

Table 1: Abbreviations and acronyms used in this document

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 audio dataset access schema, including the file-system layout, naming conventions, and metadata fields used to locate and load audio recordings.

AC2: The MEG dataset access schema, including the file-system layout, trial or segment organization, and metadata fields used to locate and load MEG data.

AC3: The experiment configuration schema and validation rules, including how file paths, model choices, lag-window parameters, and cross-validation settings are represented and checked.

AC4: The training procedure used to optimize the speech model parameters.

AC5: The checkpoint serialization format and loading strategy for trained checkpoints and frozen models.

AC6: The procedure used to extract hidden states and convert them into predictor representations for downstream alignment and analysis.

AC7: The statistical analysis protocol used to fit encoding models, select regularization, compute cross-validated scores, and quantify baseline-referenced gains.

AC8: The choice of visualization routines and output formats for plots, summary tables, and saved figures.

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 and output remain file-based rather than interactive or device-driven.

UC2: The system remains an offline scientific workflow for model-to-brain alignment of speech data, rather than a real-time or online prediction system.

UC3: No dedicated database system is required. Datasets and outputs are managed through the file system.

UC4: No external communication protocol is required beyond local file access.

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: Audio Data Module

M2: MEG Data Module

M3: Input Parameters Module

M4: Model Training Module

M5: Frozen Model Module

M6: Representation Extraction Module

M7: Statistical Analysis Module

M8: Plotting Module

Level 1	**Level 2**
Hardware-Hiding Module	Audio Data Module
	MEG Data Module
Behaviour-Hiding Module	Input Parameters Module
	Model Training Module
	Frozen Model Module
	Representation Extraction Module
	Statistical Analysis Module
Software Decision Module	Plotting Module

Table 2: Module Hierarchy

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS. At this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

Requirements (SRS)	**Module ID**
R1 (*Accept inputs*)	M1, M2, M3
R2 (*Train model*)	M3, M1, M4
R3 (*Extract predictor stream*)	M5, M6
R4 (*Align predictor matrix*)	M6
R5 (*Build design matrix*)	M7
R6 (*Fit TRF*)	M7
R7 (*Predict and score*)	M7
NFR1 (*Usability*)	M3, M7, M8
NFR2 (*Maintainability*)	M1, M2, M3, M4, M5, M6, M7, M8
NFR3 (*Portability*)	M1, M2, M3, M4, M5, M6, M7, M8
NFR4 (*Accuracy*)	M5, M6, M7

Table 3: Connection Between Requirements and Module IDs

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. *Model–to–Brain Alignment for Speech Recognition* means the module will be implemented by the Model–to–Brain Alignment for Speech Recognition 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

These modules hide decisions related to external data storage and file-system organization.

7.1.1 Audio Data Module (M1)

Secrets: The storage format, indexing scheme, and metadata organization of the audio datasets.

Services: Loads and serves audio recordings, dataset splits, sampling rates, and utterance-level metadata to the rest of the system.

Implemented By: Model–to–Brain Alignment for Speech Recognition

Type of Module: Abstract Object

7.1.2 MEG Data Module (M2)

Secrets: The storage format, trial structure, sensor metadata, and stimulus linkage used by the MEG dataset.

Services: Loads and serves MEG responses, subject lists, sensor counts, sampling rates, and trial-level metadata for downstream analysis.

Implemented By: Model–to–Brain Alignment for Speech Recognition

Type of Module: Abstract Object

7.2 Behaviour-Hiding Module

These modules provide the externally visible scientific behaviour of the system, including parameter management, model preparation, predictor construction, and statistical evaluation.

7.2.1 Input Parameters Module (M3)

Secrets: The schema and validation rules for experiment configuration, including how paths, dataset identifiers, model settings, lag-window parameters, and cross-validation settings are represented and checked.

Services: Stores, validates, and provides experiment configuration parameters to other modules.

Implemented By: Model–to–Brain Alignment for Speech Recognition

Type of Module: Abstract Object

7.2.2 Model Training Module (M4)

Secrets: The training workflow used to obtain optimized model parameters from configured audio data.

Services: Trains speech models and provides trained model parameters together with training summaries.

Implemented By: Model–to–Brain Alignment for Speech Recognition

Type of Module: Library

7.2.3 Frozen Model Module (M5)

Secrets: The checkpoint loading and parameter-freezing strategy used to prepare trained models for representation extraction.

Services: Loads trained model checkpoints and provides frozen models for representation extraction.

Implemented By: Model–to–Brain Alignment for Speech Recognition

Type of Module: Abstract Object

7.2.4 Representation Extraction Module (M6)

Secrets: The procedure used to extract layer-wise hidden representations and convert them into predictor matrices aligned to the MEG sampling grid.

Services: Extracts hidden representations from frozen models and constructs aligned predictor matrices for downstream analysis.

Implemented By: Model–to–Brain Alignment for Speech Recognition

Type of Module: Library

7.2.5 Statistical Analysis Module (M7)

Secrets: The project-specific evaluation workflow used to construct analysis matrices, fit encoding models, perform cross-validation, aggregate scores, and compute baseline-referenced comparisons.

Services: Performs the statistical evaluation of aligned predictors under a common protocol, including design-matrix construction, model fitting, prediction, score computation, and baseline comparison.

Implemented By: Model–to–Brain Alignment for Speech Recognition using external scientific Python libraries (EElbrain).

Type of Module: Library

7.3 Software Decision Module

These modules hide implementation choices that support the scientific workflow without changing the externally visible behaviour required by the SRS.

7.3.1 Plotting Module (M8)

Secrets: The mapping from analysis outputs to figures and tables, including figure layout, plotting conventions, and output serialization format.

Services: Generates project-specific visual summaries and tables from the analysis outputs produced by M7, including layer-wise scores and baseline-referenced improvements.

Implemented By: Model–to–Brain Alignment for Speech Recognition using external plotting libraries (Eelbrain).

Type of Module: Library

8 Traceability Matrix

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

Requirement	Modules
R1(R-Inputs)	M1, M2, M3
R2(R-InputValidation)	M1, M2, M3, M6, M7
R3(R-ConstructPredictors)	M5, M6
R4(R-FairEval)	M3, M7
R5(R-Scoring)	M7
R6(R-QuantifyGain)	M7, M8
NFR1(NFR-Usability)	M3, M8
NFR2(NFR-Maintainability)	M1, M2, M3, M4, M5, M6, M7, M8
NFR3(NFR-Portability)	M1, M2, M3, M8
NFR4(NFR-Accuracy)	M5, M6, M7

Table 4: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M2
AC3	M3
AC4	M4
AC5	M5
AC6	M6
AC7	M7
AC8	M8

Table 5: 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.

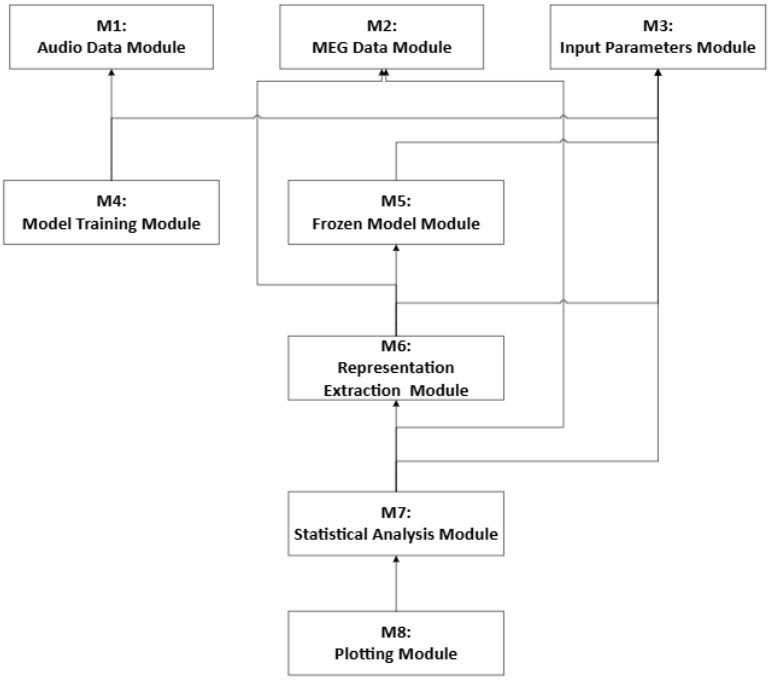

Figure 1: Use hierarchy among modules

10 User Interfaces

The primary user interface of Model–to–Brain Alignment for Speech Recognition is a script-based or command-line scientific workflow. Users provide configuration parameters, dataset locations, and model and analysis settings through files or command-line arguments. The main user-visible outputs are analysis summaries, score tables, and saved figures. No dedicated graphical user interface is required.

11 Design of Communication Protocols

N/A. Model–to–Brain Alignment for Speech Recognition does not require a communication protocol beyond local file access and standard operating-system services.

12 Database Design

NA

13 Timeline

The development schedule and division of responsibilities will be maintained in the project repository or shared project planning documents. This section may be updated later with milestones for data preparation, model training, representation extraction, statistical analysis, and report generation. See Github Project: https://github.com/Krozix-2026/cas741-speech-trf

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.