Module Guide for PyERT

Team 17, Track a Trace
Zabrain Ali
Linqi Jiang
Jasper Leung
Mike Li
Mengtong Shi
Hongzhao Tan

April 5, 2023

1 Revision History

Date	Version	Notes
January 13, 2023	1.0	Edited Abbreviations and Acronyms, Introduction, Anticipated and Unlikely Changes, Module Hierarchy
January 14, 2023	1.1	Edited Connection Between Requirements and Design and Module Decomposition
January 18, 2023	1.2	Edited Traceability Matrix and Use Hierarchy Between Modules
April 5, 2023	2.0	Modified Document according to feedback from Revision 0

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
DAG	Directed Acyclic Graph
GIS	geographic information system
M	Module
MG	Module Guide
OS	Operating System
OSM	OpenStreetMap
PyERT	Explanation of program name
R	Requirement
SC	Scientific Computing
SHP	shapefile: a geospatial vector data format for GIS software
SRS	Software Requirements Specification
UC	Unlikely Change
2D	two-dimensional

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 in 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

AC1: The specific hardware on which the software is running. This is to add portability to the software as not everyone is going to have the same hardware to run it on.

AC2: The implementation for the methods for generating route choice set. This is because the original implementation in GERT uses an external software ArcGIS Pro for the route choice set. Since PyERT only uses open-source free libraries, the method for generating the route choice set will likely be different.

AC3: The implementation for the methods for generating route choice analysis variable. This is because the original implementation in GERT uses an external software ArcGIS Pro for the route choice analysis variables. Since PyERT only uses open-source free libraries, the method for generating route analysis variables will likely be different.

AC4: The specific algorithms that are used for geometric object analysis and manipulation. This is because the original implementation in GERT uses an external software ArcGIS Pro for the geometric object analysis and manipulation. Since PyERT only uses open-source free libraries, the method for geometric object analysis and manipulation will likely be different.

AC5: The specific algorithms that are used for network analysis. This is because the original implementation in GERT uses an external software ArcGIS Pro for the network analysis. Since PyERT only uses open-source free libraries, the method for network analysis will likely be different.

4.2 Unlikely Changes

UC1: ~~An unlikely change for the module design is~~ The implementation being designed around the open-source free library GeoPandas. This is unlikely to change as GeoPandas makes working with geospatial data easier in Pythong and extends the use of pandas to be able to perform the operations needed for the GPS databases.

UC2: ~~An unlikely change for the module design is~~ The design of building the modules based on the GeoDataFrame object, which is a Pandas DataFrame that has a column with geometry. This is unlikely to change because such a DataFrame object can be easily created using the GeoPandas open-source free library. This object has access to many functions, such as plotting the trajectory data, which will be used in many modules.

UC3: ~~An unlikely change for the module design is~~ The design of using the geojsonio package to transfer the GeoPandas DataFrame to GeoJSON format which will then be converted to a JSON string to display the GPS points. This is unlikely to change because geojsonio goes nicely with GeoPandas to display data in a GeoDataFrame. It allows the software to have a better form of visualization of the route and activity locations of that route based on the GeoDataFrame.

UC4: ~~An unlikely change for the module design is~~ The design of using Pyrosm to create the GeoDataFrame for driving network and buildings from OpenStreetMap PBF files. This is unlikely to change because Pyrosm allows someone to easily parse data from OpenStreetMap PBF files into GeoPandas GeoDataFrame. This makes it easier to fetch the data of large regions for network analysis.

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: GPS Data Preprocessing Module

M3: GPS Data Mode Detection Module

M4: Trip Segments and Activity Locations Extraction Module

M5: Route Choice Set Generator Module

M6: Route Choice Analysis Variables Generator Module

M7: Activity Locations Identification Module

M8: Network Data Utilities Module

M9: Main Function Module

M10: DataFrame Data Structure Module

M11: GeoDataFrame Data Structure Module

M12: Geometric Object Analysis and Manipulation Module

M13: Network Analysis Module

M14: OSM Network Dataset Reader Module

M15: Plotting Module

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	GPS Data Preprocessing Module
	GPS Data Mode Detection Module
	Trip Segments and Activity Locations Extraction Module
	Route Choice Set Generator Module
	Route Choice Analysis Variables Generator Module
	Activity Locations Identification Module
	Main Function Module
Software Decision Module	DataFrame Data Structure Module
	GeoDataFrame Data Structure Module
	Geometric Object Analysis and Manipulation Module
	Network Analysis Module
	OSM Network Dataset Reader Module
	Plotting Module

Table 1: Module Hierarchy

6 Connection Between Requirements and Design

The design of the system is intended to satisfy the requirements developed in the SRS (Ali et al., 2022). In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 3.

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. PyERT means the module will be implemented by the PyERT 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 the 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 (Ali et al., 2022)) 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 (Ali et al., 2022).

Implemented By: –

7.2.1 GPS Data Preprocessing Module (M2)

Secrets: The algorithms and functions for preprocessing the raw GPS data.

Services: Removes redundant GPS points from the raw GPS data from user's input as well as any outliers.

Implemented By: PyERT

Type of Module: Library

7.2.2 GPS Data Mode Detection Module (M3)

Secrets: The algorithms and functions detecting the modes(e.g. drive, walk, stop) of GPS data.

Services: Classifies the valid GPS data into travel or stop episodes.

Implemented By: PyERT

Type of Module: Library

7.2.3 Trip Segments and Activity Locations Extraction Module (M4)

Secrets: The algorithms and functions for extracting trip segments and activity locations from GPS data

Services: Extracts trip segments(GPS points in travel episodes generated by M3) and activity locations (GPS points in the stop episodes generated by M3).

Implemented By: PyERT

Type of Module: Library

7.2.4 Route Choice Set Generator Module (M5)

Secrets: The algorithms and functions for generating route choice sets from given trip segments.

Services: Generates route choice sets with the trip segments extracted by M4 and transportation network dataset from the user's input.

Implemented By: PyERT

Type of Module: Library

7.2.5 Route Choice Analysis Variables Generator Module (M6)

Secrets: The algorithms and functions for generating route choice analysis variables from the given route choice.

Services: Generates route choice analysis variables, including the length of the route in meters, the number of ~~each type of~~ left and right turns, the total number of roads in the route, the street name and length (in meters) of the longest leg, for the route choice generated by M5, with the data of the route choice and the transportation network dataset ~~from the user's input~~.

Implemented By: PyERT

Type of Module: Library

7.2.6 Activity Locations Identification Module (M7)

Secrets: The algorithms and functions for appending information for extracted activity locations.

Services: Appends information for activity locations extracted by M4, if such information has been provided by the transportation network dataset from the user's input.

Implemented By: PyERT

Type of Module: Library

7.2.7 Network Data Utilities Module (M8)

Secrets: The algorithms and functions for extracting transportation network, PAL and LU data from OSM API or OSM PBF file and preprocessing the extracted data to make it ready to be used by other modules.

Services: Extracts transportation network, PAL and LU data from OSM API or OSM PBF file and preprocesses the extracted data to make it ready to be used by other modules.

Implemented By: PyERT

7.2.8 Main Function Module (M9)

Secrets: The function for coordinating the running of the system.

Services: Provides the main function of the system which uses the services that are provided by other modules to read raw GPS data from CSV files, extract transportation network data from network datasets, process the raw GPS data to get trip and stop segments, match the segments onto transportation network and generate output files and visualizations.

Implemented By: PyERT

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 (Ali et al., 2022).

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

Implemented By: –

7.3.1 DataFrame Data Structure Module (M10)

Secrets: The data structure for a Pandas DataFrame data type.

Services: Provides 2D array(or can be seen as a table with rows and columns where values on the same column are of the same data type) manipulation, including building a 2D array, accessing or changing the value of a specific row, column or entry, etc.

Implemented By: Pandas

7.3.2 GeoDataFrame Data Structure Module (M11)

Secrets: The data structure for a GeoPandas GeoDataFrame data type, which is a type of specialized Pandas DataFrame with a 'geometry' column that contains geometric objects of types POINT, LINESTRING, MULTILINGSTRING, POLYGON, or etc.

Services: Provides the 2D array manipulations that are provided by Pandas DataFrame, simple methods that can be used to manipulate geometric objects, methods to read and generate SHP files, methods to convert GeoDataFrame to JSON string, etc.

Implemented By: GeoPandas

7.3.3 Geometric Object Analysis and Manipulation Module (M12)

Secrets: The algorithms to manipulate and analyze geometric objects (e.g. POINT, LINGSTRING, MULTILINESTRING, POLYGON, etc.). It is part of the GeoPandas package (see M11), but also can be imported and used separately.

Services: Provides methods to manipulate and analyze geometric objects, such as accessing the coordinates of a geometric object, calculating the distance between two geometric objects, etc.

Implemented By: Shapely

7.3.4 Network Analysis Module (M13)

Secrets: The algorithms and functions to extract and analyze transportation network dataset. See related journal article (Boeing, 2017) for details.

Services: Provides methods to extract and analyze transportation network dataset, such as accessing the coordinates of streets, amenities, land-use areas and buildings from OSM API, finding the shortest route on transportation network between two points given their coordinates, etc.

Implemented By: OSMnx

7.3.5 OSM Network Dataset Reader Module (M14)

Secrets: The algorithm to read network dataset from OSM in Protocolbuffer Binary (.pbf) format.

Services: Provides methods to extract transportation network data, buildings' information data or etc., from network datasets from OSM in Protocolbuffer Binary (.pbf) format.

Implemented By: Pyrosm

7.3.6 Plotting Module (M15)

Secrets: The algorithms for visualizing data of GeoJSON format.

Services: Provides methods to visualize GeoJSON data on geojson.io.

Implemented By: geojsonio

8 Traceability Matrix

This section shows three traceability matrices: between the modules and the functional and non-functional requirements and between the modules and the anticipated changes. See detailed description for the requirements of the system in SRS (Ali et al., 2022).

Req.	**Modules**
Req #1	M1, M9, M10, M11
Req #2	M2, M10
Req #3	M2, M10
Req #4	M3, M10
Req #5	M3, M10
Req #6	M4, M10, M11
Req #7	M4, M10, M11
Req #8	M1, M9, M5, M11, M12, M13, M14
Req #9	M1, M9, M7, M11, M14
Req #10	M1, M9
Req #11	M3, M10
Req #12	M1, M9, M6, M11, M12, M13, M14
Req #13	M1, M8, M11, M14
Req #14	M1, M8, M11, M13
Req #15	M1, M9, M10, M11

Table 2: Trace Between Functional Requirements and Modules

NFR	Modules
LF1	M1, M9
LF2	M1, M9
UH1	M1, M9, M2, M3, M4, M5, M6, M7
UH2	M1, M9
UH3	M1, M9, M2, M3, M4, M5, M6, M7
PR1	M1, M9
PR2	M1, M9
PR3	M10, M11, M12, M13, M14, M15
OE1	M1, M9, M2, M3, M4, M5, M6, M7, M10, M11, M12, M13, M14, M15
OE2	M10, M11, M12, M13, M14, M15
OE3	M10, M11, M12, M13, M14, M15
MS1	M9, M2, M3, M4, M5, M6, M7
MS3	M1, M9, M2, M3, M4, M5, M6, M7
SR1	M9, M2, M3, M4, M5, M6, M7, M10, M11, M12, M13, M14, M15
SR2	M9, M2, M3, M4, M5, M6, M7, M10, M11, M12, M13, M14, M15
SR3	M9, M2, M3, M4, M5, M6, M7
LR1	M10, M11, M12, M13, M14, M15

Table 3: Trace Between Non-Functional Requirements and Modules

AC	Modules
AC1	M1
AC2	M5
AC3	M6
AC4	M12
AC5	M13

Table 4: 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.

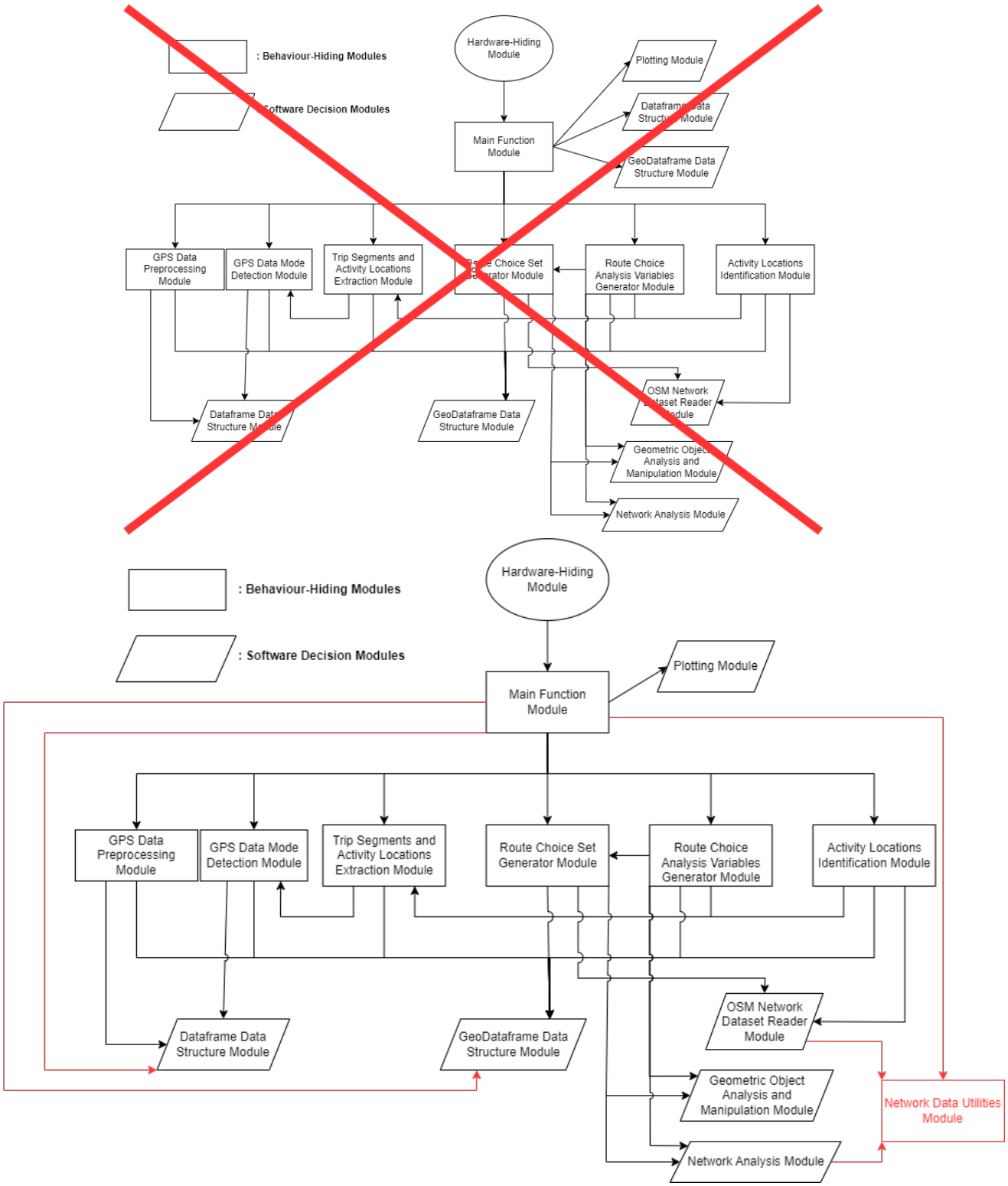

Figure 1: ~~Use Hierarchy Diagram~~Updated Use Hierarchy Diagram

References

Zabrain Ali, Linqi Jiang, Jasper Leung, Mike Li, Mengtong Shi, and Hongzhao Tan. Software requirements specification for software engineering: Pyert, 2022. URL https://github.com/paezha/PyERT-BLACK/blob/main/docs/SRS/SRS.pdf.

Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Computers, Environment and Urban Systems, 65:126–139, May 2017. URL https://www.sciencedirect.com/science/article/pii/S0198971516303970?via%3Dihub.

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.