Module Guide for Centrality in Graphs

Atiyeh Sayadi

March 2, 2024

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	**description**
AC	Anticipated Change
CC	Closeness Centrality
CIG	Centrality in Graphs
DAG	Directed Acyclic Graph
DC	Degree Centrality
M	Module
MG	Module Guide
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
UC	Unlikely Change

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

This software, like other softwares, may undergo changes. These changes are as follows:

AC1: In the future, it is possible that this software will be developed in another environment such as NetLogo.

AC2: This software can read the graph matrix directly from the input file as an adjacency matrix.

AC3: In the future, this software can also perform computations on directed graphs.

4.2 Unlikely Changes

The future changes of the software do not include the following items:

UC1: The input file will only be in the form of a text file.

UC2: Other centrality measures will not be calculated by this software.

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: Behaviour-Hiding Module

M3: Software Decision Module

Level 1	**Level 2**
Hardware-Hiding Module	-
Behaviour-Hiding Module	GUI File
Software Decision Module	Degree Closeness

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. In this stage, the system is decomposed into modules. The connection between requirements and modules is listed in Table 2.

Functional requirments:

FR1: Degree centrality for each node must be calculated accurately.

FR2: Closeness centrality for each node must be calculated accurately.

FR3: Degree centrality must be calculated for all nodes.

FR4: Closeness centrality must be calculated for all nodes.

Nonfunctional requirments:

NFR1: Accuracy

NFR2: Usability

NFR3: Maintainability

7 Module Decomposition

Modules are decomposed according to the principle of "information hiding" proposed by Parnas et al. (1984). The following is an introduction to the modules and a brief explanation of each of them.

7.1 Hardware Hiding Modules

7.2 Behaviour-Hiding Module

7.2.1 File (M1)

Secrets: Text file containing a graph matrix.

Services: This module reads the initial graph matrix from a text file and calculates its adjacency matrix.

Implemented By: CIG

Type of Module: Data Access Module.

7.2.2 GUI (M2)

Secrets: Graphical User Interface (GUI) for displaying outputs.

Services: This module utilizes the results of calculations from the Degree, Closeness, and File modules and displays the values separately on the nodes of the graph.

Implemented By: CIG

Type of Module: Graphical User Interface

7.3 Software Decision Module

7.3.1 Closeness(M3)

Secrets: Calculating the shortest paths from each node to all other nodes and then computing the closeness centrality for each node

Services: Using the output obtained from the File module, which is the adjacency matrix for the graph, it is necessary to first calculate the shortest path for each node to all other nodes according to the formula of closeness centrality. Then, based on this calculation, closeness centrality is computed. The output of this module will be in the form of a matrix of size n * 2, where n is the number of nodes in the graph. In the output matrix, the centrality of each node will be displayed against each node.

Implemented By: CIG

7.3.2 Degree(M4)

Secrets: Calculating the degree of each node in the graph and then calculating the degree centrality for each node in the graph.

Services: This module first determines the degree of each node in the graph using the output of the File module. Then, with the degree of each node and the total sum of degrees of all nodes in the graph, it calculates the degree centrality for each node. The output of this module will be in the form of a matrix of size n * 2, where n is the number of nodes in the graph. In the output matrix, the centrality of each node will be displayed against each node.

Implemented By: CIG

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**
FR1	M4
FR2	M3
FR3	M4, M1
FR4	M3, M1
NFR1	M1, M2, M3, M4
NFR2	M1, M2, M3, M4
NFR3	M1, M2, M3, M4

Table 2: Trace Between Requirements and Modules

AC	**Modules**
AC1	M1, M2, M3, M4
AC2	M1
AC3	M2

Table 3: Trace Between Anticipated Changes and Modules

9 Use Hierarchy Between Modules

Figure 1 illustrates a hierarchical view of the relationship between the modules.

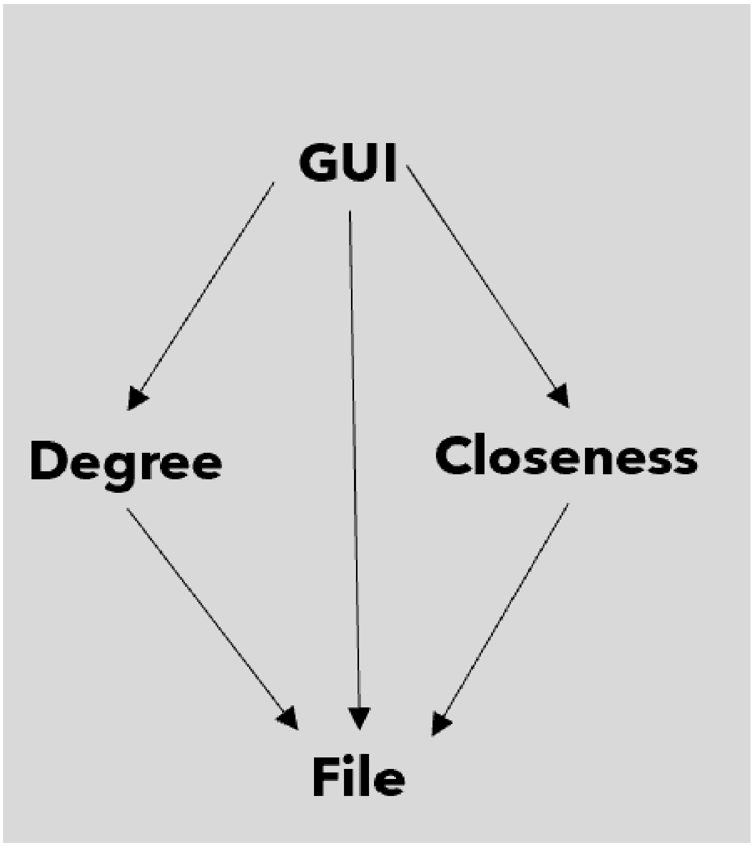

Figure 1: Use hierarchy among modules

10 User Interfaces

Designing the user interface for this software is only for displaying outputs, in such a way that the user, by clicking on each button, will be shown the graph associated with it. You can refer to the user interface image in Figure 2.

11 Design of Communication Protocols

12 Timeline

Table 4 displays the timeline for the development of each module and its respective developer.

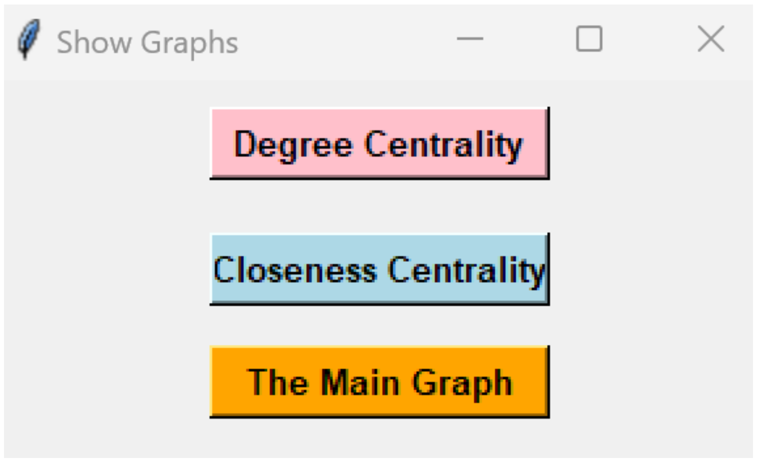

Figure 2: Graphical User Interface

Modules	**Time**	**Responsible**
File	11-15 March	Atiyeh Sayadi
Closeness	16-22 March	Atiyeh Sayadi
Degree	23-30 March	Atiyeh Sayadi
GUI	1-3 April	Atiyeh Sayadi

Table 4: Timeline

References

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

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.