Module Guide for Slope Stability Analysis Program

Henry Frankis

May 31, 2018

Contents

1 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). In the best practices for Scientific Computing (SC), Wilson et al. (2013) advise a modular design, but are silent on the criteria to use to decompose the software into modules. 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 laid out by Parnas et al. (1984), as follows:

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

In a rational design process (Parnas et al., 1984), the Module Guide (MG) is developed after completing the Software Requirements Specification (SRS). 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 is 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 2 lists the anticipated and unlikely changes of the software requirements. Section 3 summarizes the module decomposition that was constructed according to the likely changes. Section 4 specifies the connections between the software requirements and the modules. Section 5 gives a detailed description of the modules. Section 6 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 7 describes the use relation between modules.

2 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 2.1, and unlikely changes are listed in Section 2.2.

2.1 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. Anticipated changes are numbered by **AC** followed by a number.

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

AC2: The algorithm for the overall operation procedure of the program.

AC3: The format of the initial input data.

AC4: The format of the final output data.

AC5: Algorithm for determining the critical slip surface.

AC6: Criteria for a slip surface to be considered physically realistic/ kinematically admissible.

AC7: The algorithm for creating the slice points for slip surface analysis.

AC8: The weighting scheme for comparing slip surfaces factors of safety.

AC9: The algorithm for implementation of the Morgenstern Price Solver.

AC10: The algorithm for implementation of the RFEM Solver.

AC11: The algorithm for assigning soil properties to slices.

AC12: The implementation for the sequence (array) data structure.

AC13: The method of generating pseudo-random numbers.

AC14: The method of displaying the final output.

2.2 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. As an example, the model is assumed to follow the definition in the SRS. If a new model is used, this will mean a change to the input format, fit parameters module, control, and output format modules.

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

UC2: There will always be a source of input data external to the software.

UC3: Output data are displayed to the output device.

3 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table1. The modules listed below are the modules that will actually be implemented.

M1: Hardware-Hiding Module

M2: Control Module

M3: Input Format Module

M4: Output Format Module

M5: Genetic Algorithm Module

M6: Kinematic Admissibility Module

M7: Slip Slicer Module

M8: Slip Weighting Module

M9: Morgenstern Price Solver Module

M10: Rigid Finite Element Method Solver Module

M11: Slice Property Sorter Module

M12: Sequence Data Structure Module

M13: Random Number Generator Module

M14: Plotting Module

Note that M1 is a commonly used module and is already implemented by the operating system.

4 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 Table2. The program implements two distinct solution methods of calculating the factor of safety of the slope, the Morgenstern Price method and the Rigid Finite Element (RFEM) method, both developed in the SRS. The Morgenstern Price solver is implemented by M9, further information on the Morgenstern Price algorithm can be found in Zhu et al. (19 February 2005). The RFEM solver is implemented by M10, further information on the RFEM algorithm can be found in Stolle and Guo (20 May 2008). The critical slip identification goal (G2,IM4 in SRS) is implemented by use of a genetic algorithm M5, further information on use of a genetic algorithm in relation to a slope stability problem can be found in Li et al. (25 June 2010).

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Control Module
	Input Format Module
	Output Format Module
	Genetic Algorithm Module
	Kinematic Admissibility Module
	Slip Slicer Module
	Slip Weighting Module
	Morgenstern Price Solver Module
	RFEM Solver Module
	Slice Property Sorter Module
Software Decision Module	Sequence Data Structure Module
	Random Number Generating Module
	Plotting Module

Table 1: Module Hierarchy

5 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. *SSA* means the module will be implemented by the SSA 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. Whether or not this module is implemented depends on the programming language selected.

5.1 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

5.2 Behaviour-Hiding Module

Secrets: The contents of the required behaviors.

Services: Includes programs that provide externally visible behaviors 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: –

5.2.1 Control Module (M2)

Secrets: The algorithm for coordinating the running of the program.

Services: Provides the main program.

Implemented By: SSA

5.2.2 Input Format Module (M3)

Secrets: The format and structure of the input data.

Services: Reads the input data from an input file, and/or prompted command line inputs. Input data includes the x,y coordinates of the slope, with a set of coordinates for each layer. For each layer its soil properties of effective angle of friction, effective cohesion, dry unit weight, saturated unit weight, elastic modulus, and Poisson's ratio are stored in vectors of soil properties. If a piezometric surface exists in the slope, its coordinates and the unit weight of water are also included in the input. Lastly an expected range for the entrance and exit points of the critical slip surface are inputted.

Implemented By: SSA

5.2.3 Output Format Module (M4)

Secrets: The format and structure of the output data.

Services: Outputs the results of the calculations, including the factor of safety for the critical slip calculated by the Morgenstern Price Module (M9) and Rigid Finite Element Method Module (M10), and a plot of the critical slip surface on the slope geometry, with the showing the element displacements as calculated by the RFEM Module

Implemented By: SSA

5.2.4 Genetic Algorithm Module (M5)

Secrets: Algorithm to identify the slip surface that has the minimum factor of safety, based on the inputs.

Services: Searches the slope for the critical slip surface with the minimum factor of safety

Implemented By: SSA

5.2.5 Kinematic Admissibility Module (M6)

Secrets: Algorithm to determine if a given slip surface passes or fails a set of admissibility criteria.

Services: Some slip surfaces are physically unlikely or impossible to occur in a slip surface, such as slip surfaces containing sharp angles, or going above the slope surface. Ensures randomly generated or mutated slopes from the Genetic Algorithm Module (M5) are physically possible according to the criteria of the Kinematic Admissibility Module.

Implemented By: SSA

5.2.6 Slip Slicer Module (M7)

Secrets: Algorithm to determine the coordinates of where the slip surface interslice nodes occur.

Services: When preparing a slip surface for analysis by the Morgenstern Price Module (M9) or the RFEM Module (M10), the x-coordinates defining the boundaries of the slices are identified and stored in a vector.

Implemented By: SSA

5.2.7 Slip Weighting Module (M8)

Secrets: The weighting for each slip surface in a set of slip surfaces, based on each slip surfaces factor of safety.

Services: Weights a set of slip surfaces generated by the Genetic Algorithm Module (M5) based on their factors of safety. A slip surface with a low factor of safety will have a high weight as it is more likely to be or to lead to generation of the critical slip surface.

Implemented By: SSA

5.2.8 Morgenstern Price Solver Module (M9)

Secrets: The factor of safety of a given slip surface.

Services: Calculates the factor of safety of a given slip surface, through implementation of a Morgenstern Price slope stability analysis method.

Implemented By: SSA

5.2.9 RFEM Solver Module (M10)

Secrets: The algorithm to perform a Rigid Finite Element Method analysis of the slope.

Services: Calculate the global factor of safety, local slice factors of safety, and local slice displacements of a given slip surface under given conditions, through implementation of a rigid finite element slope stability analysis method.

Implemented By: SSA

5.2.10 Slice Property Sorter Module (M11)

Secrets: Algorithm to assigns soil properties to slices based on the location of the slice with respect to the different soil layers.

Services: When performing slip analysis with the RFEM Solver Module (M10) or Morgenstern Price Module (M9), the base and interslice surfaces of each slice in the analysis requires a soil constant. Slice Property Sorter Module identifies which soil layer the surface is in to assign properties from that soil layer, and uses a weighting scheme when the surface crosses multiple soil layers.

Implemented By: SSA

5.3 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 a data structure and algorithms used in the system that do not provide direct interaction with the user.

Implemented By: –

5.3.1 Sequence Data Structure Module (M12)

Secrets: The data structure for a sequence data type.

Services: Provides array manipulation, including building an array, accessing a specific entry, slicing an array etc.

Implemented By: Matlab

5.3.2 Random Number Generator Module (M13)

Secrets: Pseudo-random numbers between 0 and 1.

Services: Randomly produces numbers between 0 and 1, using a chaotic function with an external seed. Used when generating slip surfaces in the Genetic Algorithm Module M5.

Implemented By: Matlab

5.3.3 Plotting Module (M14)

Secrets: The data structures and algorithms for plotting data graphically.

Services: Provides a plot function.

Implemented By: Matlab

6 Traceability Matrix

This section shows two traceability matrices: between the modules and the requirements Table2 and between the modules and the anticipated changes Table3.

Req.	**Modules**
R1	M2 M3 M12
R2	M5 M13 M12
R3	M6 M12
R4	M5 M7 M12
R5	M9 M12
R6	M8 M12
R7	M5 M13 M12
R8	M5 M12
R9	M2 M4 M7 M12
R10	M4 M9 M10 M12
R11	M4 M14

Table 2: Trace Between Requirements and Modules

Req.	**Modules**
AC1	M1
AC2	M2
AC3	M3
AC4	M4
AC5	M5
AC6	M6
AC7	M7
AC8	M8
AC9	M9
AC10	M10
AC11	M11
AC12	M12
AC13	M13
AC14	M14

Table 3: Trace Between Anticipated Changes and Modules

7 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. Fig1 illustrates the use hierarchy between the modules. 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.

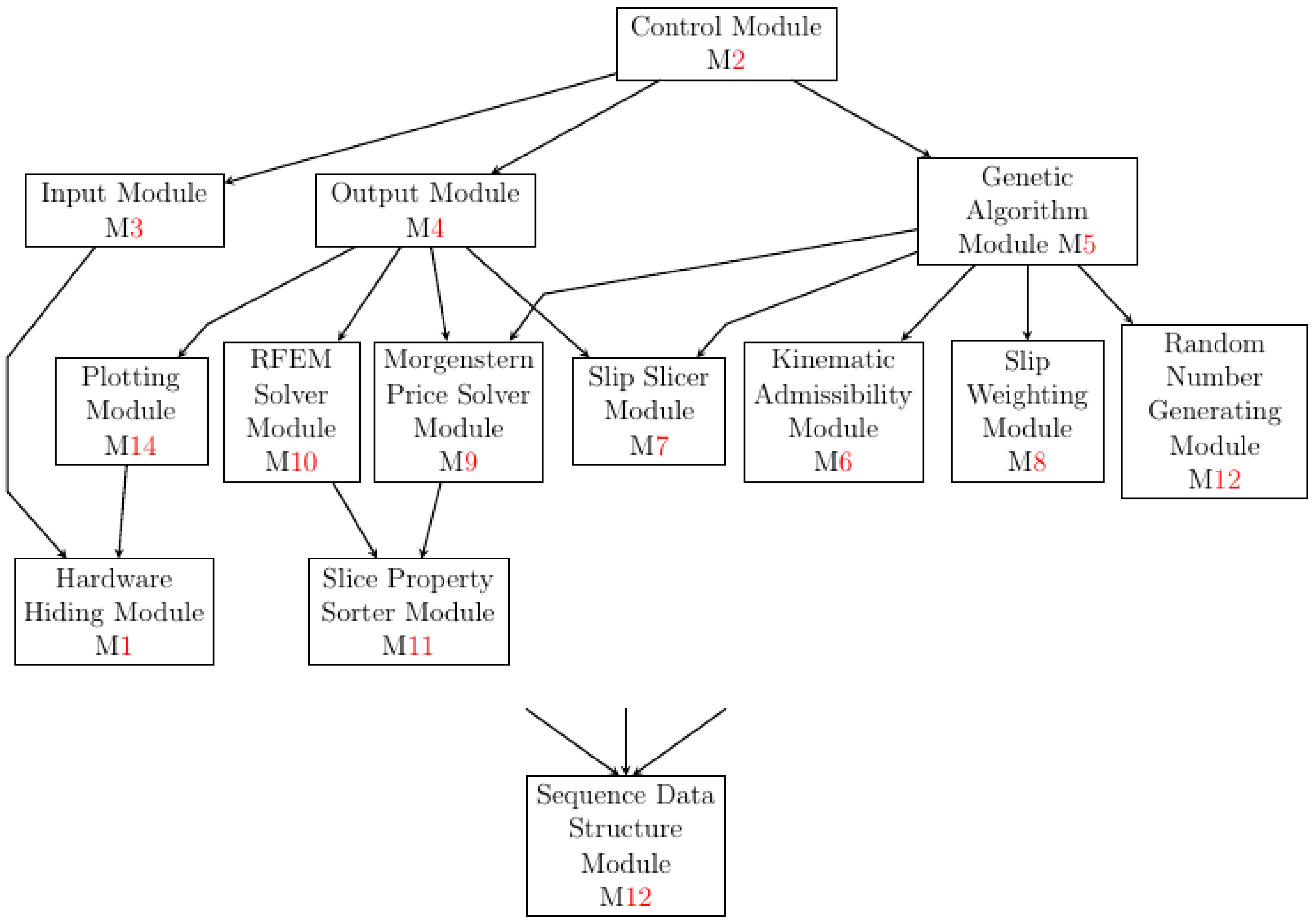
Control Module M2
Input Module M3
Output Module M4
Genetic Algorithm Module M5
Plotting Module M14
RFEM Solver Module M10
Morgenstern Price Solver Module M9
Slip Slicer Module M7
Kinematic Admissibility Module M6
Slip Weighting Module M8
Random Number Generating Module M12
Hardware Hiding Module M1
Slice Property Sorter Module M11
Sequence Data Structure Module M12

Figure 1: Use Hierarchy Diagram

References

Yu-Chao Li, Yun-Min Chen, Tony L.T Zhan, Dao-Sheng Ling, and Peter John Cleall. An efficient approach for locating the critical slip surface in slope stability analyses using a real-coded genetic algorithm. *Can. Geotech. J.*, (47):806–820, 25 June 2010.

D. L. Parnas, P. C. Clements, and D. M. Weiss. The modular structure of complex systems. In *ICSE '84: Proceedings of the 7th international conference on Software engineering*, pages 408–417, Piscataway, NJ, USA, 1984. IEEE Press. ISBN 0-8186-0528-6.

David L. Parnas. On the criteria to be used in decomposing systems into modules. *Comm. ACM, vol. 15, no. 2, pp. 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.

Dieter Stolle and Peijun Guo. Limit equilibrum slope stability analysis using rigid finite elements. *Can. Geotech. J.*, (45):653–662, 20 May 2008.

Greg Wilson, D.A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H.D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumblet, Ben Waugh, Ethan P. White, and Paul Wilson. Best practices for scientific computing. *CoRR*, abs/1210.0530, 2013.

D.Y. Zhu, C.F. Lee, Q.H. Qian, and G.R. Chen. A concise algorithm for computing the factor of safety using the morgenstern–price method. *Can. Geotech. J.*, (42):272–278, 19 February 2005.