Module Guide for MISEG

Ao Dong

December 13, 2019

1 Revision History

Date	Version	Notes
Nov 25	1.0	Initial draft
Dec 11	1.1	Modification according to feedback
Dec 13	1.2	Modification according to feedback

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	**description**
1D	one-dimensional
3D	three-dimensional
AC	Anticipated Change
DAG	Directed Acyclic Graph
dcm4che	A DICOM Toolkit and Library
M	Module
MG	Module Guide
$\mathbb{N}$	Natural Number
OS	Operating System
R	Requirement
SC	Scientific Computing
SRS	Software Requirements Specification
MISEG	Explanation of program name
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. Theother shows the relation between anticipated changes and the modules. Section9 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 approachadapted here is called design for change.

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

AC2: The format of the [Does the word "initial" have significance here? I think you can just say "input data". —SS] [I agree. Deleted "initial" —Author] input data. MISEG convert DICOM image frames to bitmap images before segmentation with dcm4che library. It may convert to other image formats with other libraries.

AC3: The format of the [As above, do you need the word "final"? —SS][I agree. Deleted "final" —Author] output data.

AC4: MISEG only uses global threshold with Otsu's Method, as explained in the SRS Dong (2019). [If you mention Otsu's method, then you should give a citation to where this method is introduced/explained. I also wonder what is the secrethere.Is it a secret the Otsu's method is being used, or will the interfaceexpose thatdecision. —SS] [I suppose "using 2 thresholds to segment theimage" is not a secret, but "these 2 thresholds are calculated with Otsu's method" is a secret. So yes, it's a secret the Otsu's method is being used, but the interface won't expose that decision. —Author] It could include the method using local thresholds in the future. It also only uses multi-threshold method with 2 thresholds. It could use the method with more thresholds in the future. MISEG only uses threshold segmentation method. It could use all the methods in Table 1 in the future.

AC5: MISEG only verify the resolution and grayscale value of processed image and segmentation image, it can make more thoroughly verificationin the future.

AC6: The constant values used in the program.

AC7: MISEG only has segmentation function. It could have functions for all the sections in analysis in Figure 1 in the future.

AC8: The implementation for the sequence (array) data structure.

Generation	Category		
	Region-based	**Boundary Following**	**Pixel Classification**
1^{st}	• Region growing	• Edge tracing (heuristic)	• Intensity threshold
2^{nd}	• Deformable models • Graph search	• Minimal path • Target tracking • Graph search • Neural networks • Multiresolution	• Statistical pattern recognition • C-means clustering • Neural networks • Multiresolution
3^{rd}	• Shape models • Appearance models • Rule-based • Coupled surfaces		• Atlas-based • Rule-based

Table 1: Segmentation Methods Withey and Koles (2007)

AC9: Image Data Structure has 2 $\mathbb{N}$ and a 1D sequence now. MISEG manipulates multi-frame images with a sequence of image data created from it. It may use other image data structures in the future, such as a 3D sequence representing multi frame images.

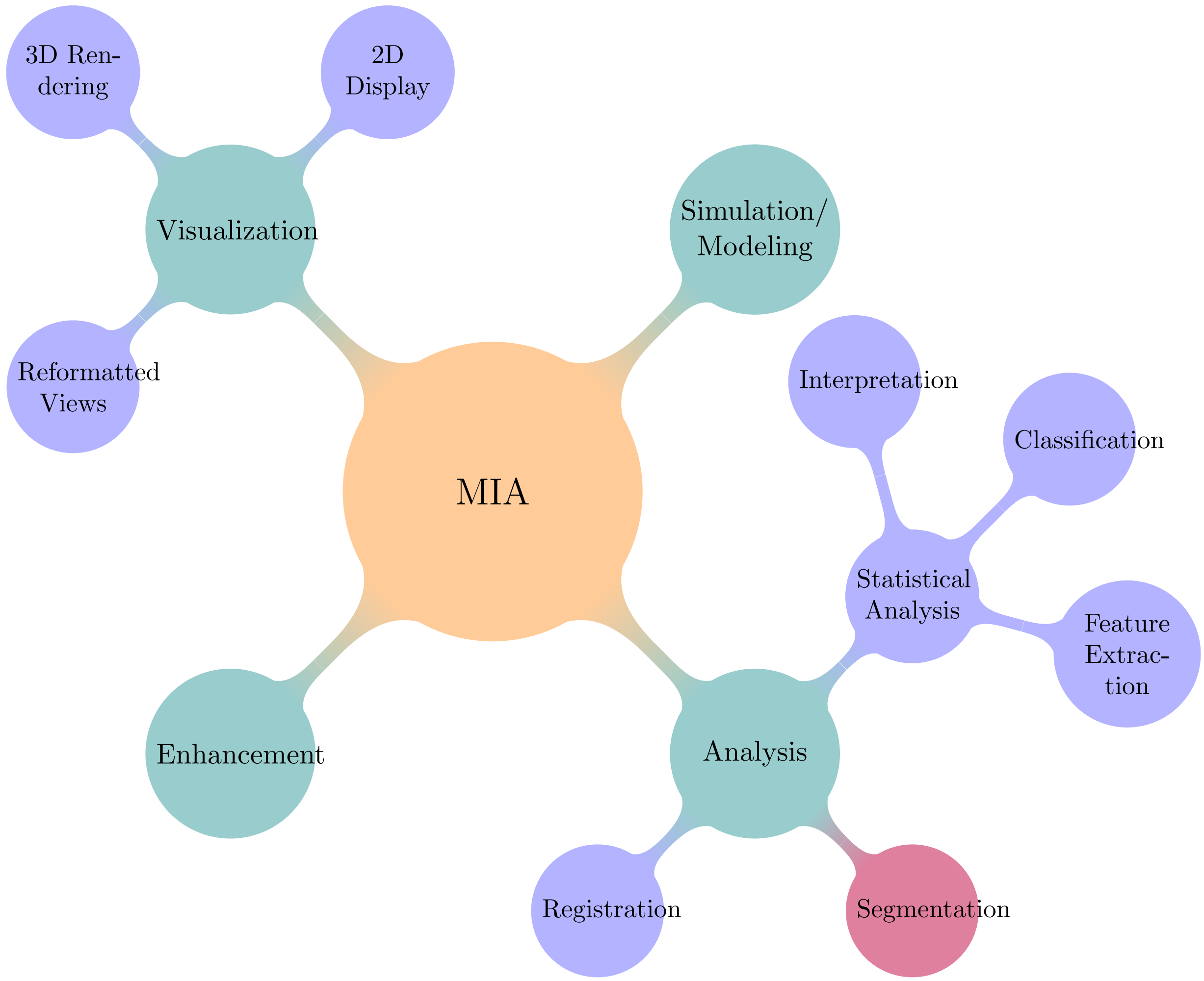

Figure 1: Major functions of MIA Dong (2019)

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 and/or Keyboard, Output: File, Memory, and/or Screen).

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

UC3: The goal of the system is related to medical imaging analysis.

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: Hardware-Hiding Module

M2: Input Module

M3: Output Module

M4: Optimal Thresholds Calculation

M5: Image Verification

M6: Constant Values

M7: Control Module

M8: Sequence Data Structure

M9: Image Data Structure

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Module Output Module Optimal Thresholds Calculation Image Verification Constant Values Control Module
Software Decision Module	Sequence Data Structure Image Data Structure

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. 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. *MISEG* means the module will be implemented by the MISEG 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 haveto be implemented.

7.1 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

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) 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: –

7.2.1 Input Module (M2)

Secrets: The format and structure of the input data and the processed input data.

Services: Converts the input data into the data structure used by the input module, stores the processed data, and uses another module to verify the validity of the processed data.

Implemented By: MISEG

7.2.2 Output Module (M3)

Secrets: The algorithm and process of generating the output data.

Services: Verify the validity of the optimal threshold values, verify the user's choice, use optimal threshold values to generate segmentation file according to chosen method.

Implemented By: MISEG

7.2.3 Optimal Thresholds Calculation (M4)

Secrets: The algorithm and process of calculation.

Services: Get user's choice of segmentation method, then calculate and display optimal threshold value(s) accordingly.

Implemented By: MISEG

7.2.4 Image Verification (M5)

Secrets: The algorithm and process of verification.

Services: Verify the validity of the processed data and segmentation data.

Implemented By: MISEG

7.2.5 Constant Values (M6)

Secrets: The way storing and modifying the constant values.

Services: Provide constant values for other modules.

Implemented By: MISEG

7.2.6 Control Module (M7)

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

Services: Provides the main program.

Implemented By: MISEG

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.

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

Implemented By: –

7.3.1 Sequence Data Structure (M8)

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: Java

7.3.2 Image Data Structure (M9)

Secrets: The algorithm and process of recording and exchanging image information.

Services: Provides the data structure to record and exchange image information.

Implemented By: MISEG

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, M5, M6, M7, M8, M9
R2	M3, M5, M6, M8, M9
R3	M4, M8, M9
R4	M3, M5, M6, M7, M8, M9
R5	M3, M6, M8, M9

Table 3: Trace Between Requirements and Modules

AC	Modules
AC1	M1
AC2	M2
AC3	M3
AC4	M4
AC5	M5
AC6	M6
AC7	M7
AC8	M8
AC9	M9

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 ifcorrect 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 2 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.

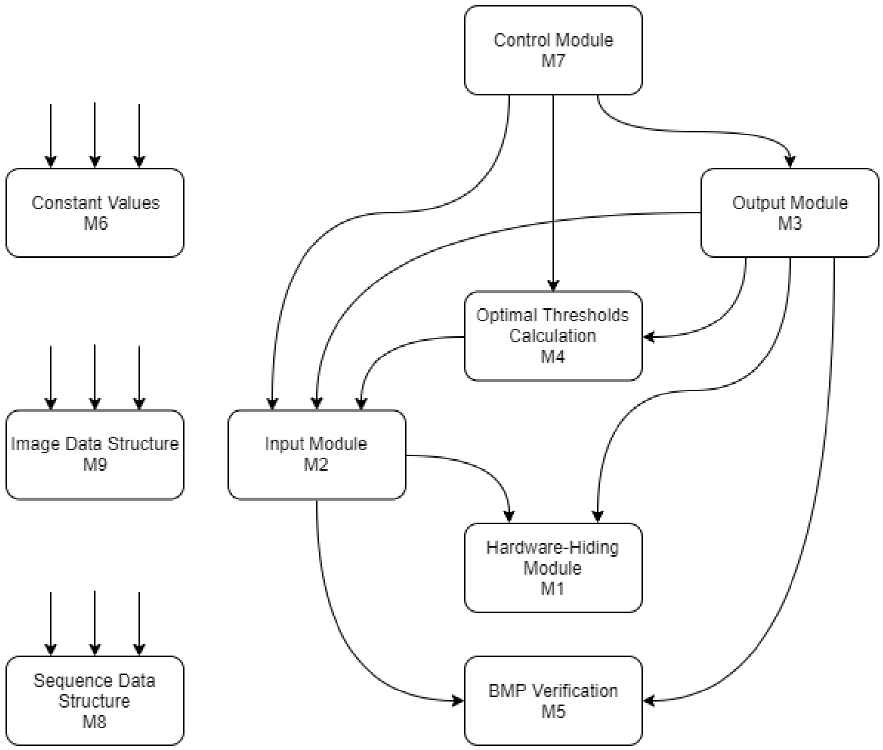

Figure 2: Use hierarchy among modules

[MG looks good. —SS]

References

Ao Dong. Software requirements specification for medical image segmentation library. 2019. URL https://github.com/Ao99/MISEG/blob/master/docs/SRS/SRS.pdf.

D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058, December 1972. ISSN 0001-0782. doi: 10.1145/361598.361623. URL http://doi.acm.org/10.1145/361598.361623.

D.L. Parnas. Designing software for ease of extension and contraction. In *Proceedings of the 3rd International Conference on Software Engineering*, ICSE '78, pages 264–277, Piscataway, NJ, USA, 1978. IEEE Press. URL http://dl.acm.org/citation.cfm?id=800099.803218.

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.

D. J. Withey and Z. J. Koles. Medical image segmentation: Methods and software. In *2007 Joint Meeting of the 6th International Symposium on Noninvasive Functional Source*

Imaging of the Brain and Heart and the International Conference on Functional Biomedical Imaging, pages 140–143, Oct 2007. doi: 10.1109/NFSI-ICFBI.2007.4387709. URL https://ieeexplore.ieee.org/abstract/document/4387709.