Project Title: System Verification and Validation Plan for MISEG

Ao Dong

December 17, 2019

1 Revision History

Date	Version	Notes
Oct 27	1.0	Initial Draft
Nov 11	1.1	Minor Revision
Dec 11	1.1	Modification according to feedback

Contents

List of Tables

List of Figures

2 Symbols, Abbreviations and Acronyms

symbol	description
2D	Two-Dimensional
DICOM	Digital Imaging and Communications in Medicine
Git	a distributed version-control system for tracking changes in source code during software development
IDE	Integrated Development Environment
JIRA	a proprietary issue tracking product
JUnit	a unit testing framework for the Java programming language.
$k^\star$	optimal threshold value found by Otsu' Method
$k_1^\star$	optimal threshold value found by Otsu' Method with multiple thresholds
$k_2^\star$	optimal threshold value found by Otsu' Method with multiple thresholds
L	number of the discrete levels of the feature value
MG	Module Guide
MIA	Medical Imaging Applications
MIS	Module Interface Specification
MISEG	Medical Imaging Segmentation Library
Redmine	a free and open source, web-based project management and issue tracking tool
SourceForge	a web-based service that offers software developers a centralized online location to control and manage free and open-source software projects
SRS	Software Requirements
T	Test
Trac	an open-source, Web-based project management and bug tracking system
VnV	Verification and Validation

This document describes procedures concerning the testing of one software of the MIA family for compliance with the requirements. It also describes how the quality of the program is assured.

Some general information such as introduction to the software and testing objectives are included in Section 3. Verification plans and testdescriptions are in Section 4 and 5, respectively.

3 General Information

3.1 Summary

The software going through the test is Medical Imaging Segmentation (MISEG).

Segmentation, separation of structures of interest from the background and from each other Bankman (2000). Image segmentation is the process of partitioningan image into different meaningful segments. In medical imaging, these segments often correspond to different tissue classes, organs, pathologies, or other biologically relevant structures Forouzanfar et al. (2010).

MISEG uses one of many segmentation algorithms - the Intensity Threshold method. It also uses Otsu's Method to find the optimal threshold value(s). Afterreceiving input medical image from the users, MISEG calculates the optimalthreshold value(s), and output the processed segmentation image.

3.2 Objectives

The requirements that the software has to be verified against can be found in the SRS document. All the functional and nonfunctional requirements should be tested, with test descriptions in Section 5.

The goal of verifying and validating is to increase confidence in the software implementation. The most important qualities to focus on are nonfunctional requirements, such as correctness and usability.

This document will be used as a starting point for the verification and validation report. The test cases presented within this document will be executed and the output will be analyzed to determine if the software is implemented correctly.

3.3 Relevant Documentation

- SRS Dong (2019c)
- Unit VnV Plan **?**
- MG Dong (2019a)
- MIS Dong (2019b)

[Your design documents are also relevant. You can "fake it" and list them here too. —SS] [I've cited all relevant documents here —Author]

4 Plan

The following sections provide more detail about the VnV of the MISEG family. Information about the testing participants is provided, and the verification plans for SRS, design, implementation and validation plan for software are described.

4.1 Verification and Validation Team

- Ao Dong
- Prof. Spencer Smith
- Peter Michalski
- Zhi Zhang
- Sasha Soraine
- Sharon Wu

  [You can be more specific here. Some students have explicitly been assigned to work on your project. You should also list the course instructor as part of the VnV team. —SS] [added relevant individuals' names here —Author]

4.2 SRS Verification Plan

The SRS can be reviewed with the help from the author's professor and classmates. Teamwork will be done systematically by reviewing each others SRS orother documents. The whole process can be done through GitHub by reviewing andsubmitting issues. Reviewers can give revision suggestions to the author, and the author has the responsibility to check all the submitted issues and make necessary adjustments accordingly.

4.3 Design Verification Plan

During the writing of SRS, the identification of verification activity is considered parallel. This enables the writer to make sure that the specificationin the SRS is verifiable. Any future changes to the SRS should not compromise the verifiability. [This last sentence is not a complete thought. —SS][adjusted —Author]

Some details need to be identified, such as measurement methods, test environment, development strategy, resources, tools, and facilities. Before making the final plan, the proposed plan can be reviewed by the VnV team, and issues can be submitted to improve the plan.

Usually the plan should be ready before the implementation stage. However, during the implementation, if specifications need to be modified in SRS, the plan might need to be updated accordingly.

The specifications and test plan shall be well-documented. There can be preliminary test plan to make improvements to the final plan.

[This is not a very specific plan. Who is going to review your design? Howare they going to do the review? Peter had some specific ideas for doing thisthat you might like to borrow. —SS]

[Peter's paper only has two sentences here. I borrowed the following idea anyway. —Author]

The verification of the design shall help achieving the objectives described in Section 3.2 **?**. Specifically, the verification should ensure that all the functional requirements and nonfuncionaly requirements can be fully tested.

[I borrowed Sasha's idea as follows. —Author]

Three methods can be used for the design verification **?**:

- Rubber duck testing. This verification shall be done by the author. It includes examining the whole MG and MIS and explaining the software process aloud to a (imaginary) rubber duck.
- Expert review. This verification shall be done by Prof. Smith and the MG and MIS reviewers Zhi Zhang and Sharon Wu. The design shall be verified to be able to meet all the requirements in the SRS. The reviewing process shall use anissue tracking and version control tool such as GitHub.
- Task-based peer review. This verification shall be done by Peter Michalskiand other interested peers, who can complete individual tasks by verifying certain parts of the whole software design.

4.4 Implementation Verification Plan

Specific verification methods will be carefully chosen for functional and nonfunctional requirements respectively. For instance, both Static Verification and Dynamic Verification will be used.

Static aspects such as code conventions, software metrics calculation, anti-pattern detection will be analyzed for some nonfunctional requirements in Section 5.2. Both manual and automatic techniques will be used for investigation, mathematical calculations, logical evaluation, etc. Regarding the automatic techniques, some static code analyzer can be used, such as PMD (`https://pmd.github.io/`), [Have a look at the writing checklist - additional spaces are needed before the opening brackets. —SS] [adjusted —Author] LGTM (`https://lgtm.com/help/lgtm/about-lgtm`) and Deep Dive (`https://discotek.ca/deepdive.xhtml`).

After selecting a group of test cases consisting of tests data, dynamic verification will be used by execution of the system or its units. By finding out the output test results, we can execute testings for the functional requirements in Section 5.1. By methods like

questionnaire andinterview, we can analyze the nonfunctional requirements listed in Section 5.2.

4.5 Software Validation Plan

One possible validation approach is interviewing Dr. Michael Noseworthy to findout if MISEG is really what the users need.

5 System Test Description

5.1 Tests for Functional Requirements

The functional requirements described in the SRS Dong (2019c). [It is a maintenance nightmare to copy and paste between documents. You can just reference the original document. Using the original files in the Blank Project Template repo with make, it is possible to cross-reference between documents. —SS] [I cited SRS here, and deleted the copied words —Author]

MISEG shall verify that the input data are valid, shall guarantee that theoutput is consistent with the input and meet the same standard, and shall provide correct calculation and output.

R1 will be tested in Section 5.1.1, R3 will be tested in 5.1.2, R5 in Section 5.1.3 and R2 and R4 in Section 5.1.4.

5.1.1 Input verification

According to R1 in the SRS, MISEG shall verify that the input data are valid. A valid input image must be 12-bit or 16-bit grayscale DICOM image. An error message shall be displayed if input data are invalid.

Part of the test for nonfunctional requirements including Robustness in Section 5.2.3 is also done here.

In this test, various types of input file will be tested, MISEG shall onlytake 12-bit or 16-bit grayscale DICOM image with width and height greater than 0 and with valid pixel values as input, as described in the Section Input Data Constraints in the SRS. Taken incorrect format or file type, it shall display an error message.

Input Verification Test

1. Invalid filename extensions

   Control: Automatic

   Initial State: MISEG is started and running

Input: the prepared files invalid.txt, invalid.pdf, invalid.jpg in the same folder as this document; or a random file with an invalid filename extension, such as .txt, .pdf and .jpg, which shall not be .dcm nor .dcm30.

Output: MISEG shall display warning that this file is not supported.

Test Case Derivation: successfully display error message.

How test will be performed: it will be performed by the test team with help of JUnit. The test team should load the whole project with an IDE and run the JUnittest file /test/InputFilenameTest.java. [You should make this test automatic. Using a unit testing framework, a test case can be defined that is considered successful is the correct exception is raised. —SS] [Modified to use a "fake" JUnit test module which might won't be in the folder soon —Author]

2. Invalid file format

   Control: Automatic

   Initial State: MISEG is started and running

   Input: the prepared file invalid.dcm and invalid.dcm30 in the same folder as this document; or a random file whose filename extension has been changed from an invalid one to .dcm and .dcm30, but the file format is not 12-bit or 16-bit DICOM image.

   Output: MISEG shall display warning that this file might be damaged or theformat is not supported.

   Test Case Derivation: successfully detect the data format in the file and display error message.

   How test will be performed: it will be performed by the test team with help of JUnit. The test team should load the whole project with an IDE and run the JUnittest file /test/InputFormatTest.java. [Again, this can be automated using a unit testing framework. It can again be handled with exceptions. —SS] [Modified —Author]

3. Valid input file

   Control: Manual

   Initial State: MISEG is started and running

   Input: the prepared file valid.dcm and valid.dcm30 in the same folder as this document; or a file with an valid filename extension, such as .dcm and .dcm30, and the file format is 12-bit or 16-bit DICOM image.

   Output: MISEG shall allow this file as an input, and display a success message.

   Test Case Derivation: successfully accept the valid file and display a message.

   How test will be performed: it will be performed by the test team manually, and will be repeated multiple times.

5.1.2 Calculation

MISEG shall provide correct calculate according to Instance Models according to the user's choice of which method to use, single ormultiple global thresholds. MISEG shall also display the correct optimalthreshold value(s) $k^\star$ or $k_1^\star$ and $k_2^\star$ accordingly.

Part of the test for nonfunctional requirements including Correctness and Verifiability in Section 5.2.2 is also done here.

In this test, calculated values will be cross-checked with results from ITK-SNAPMedical Image Segmentation Tool (`https://sourceforge.net/projects/itk-snap/`). [I don't like the "suchas" here. Now is the time to make specific decisions. If you are going to use VTK say that, if you are going to use something else, say that. Being specific here does not mean you are tied to the decision, you could always change your mind later and "fake" the documentation. —SS] [Changed it to a specific software —Author]

Calculation Test

1. Display single threshold value

   Control: Automatic

   Initial State: MISEG is started and running, /test/sample_image.dcm has been taken as the input. [I would like you to be more specific. What it the image? Where can I find it? A sample image from the VTK documentation would be fine, but you need to make a specific decision. Also, I believe that the input image is part of input, not the initial state. This same comment about identifying the specific image occurs throughout your test cases. It would be fine if you used the same image for multiple tests. —SS] [Changed it to an specific input image which is also "fake", it won't actually be in the folder soon. —Author] [I prefer to remain the input part here because this test assume that an input is successfully taken. If any errors happen during the input part, they should be outside the scope of this test. —Author]

   Input: user shall choose the first one from the two options: Single or Multiple Global Thresholds.

   Output: accordingly, MISEG shall calculate and display one optimal threshold value $k^\star$, where $k^\star \in \{1, 2, 3, ..., L-2\}$. The value will be compared with output from VTK to show the correctness.

   Test Case Derivation: successfully detect the users' choice and display potentially correct number of values.

   How test will be performed: the same input image will be used for calculation inVTK, and the output optimal threshold values from VTK will be used as control value. The percentage of difference between the outputs from MISEG and VTKwill be calculated. It will be performed by the test team with assistance from JUnit, and will be repeated multiple times.

2. Display double threshold values

   Control: Automatic

   Initial State: MISEG is started and running, /test/sample_image.dcm has been taken as the input.

   Input: user shall choose the second one from the two options: Single or MultipleGlobal Thresholds.

   Output: accordingly, MISEG shall calculate and display two optimal threshold values - $k_1^\star$ and $k_2^\star$, where $k_1^\star \in [1, k_2^\star - 2]$ and $k_2^\star \in [k_1^\star + 2, L - 2]$. The values will be compared with output from VTK to show the correctness.

   Test Case Derivation: successfully detect the users' choice and display potentially correct number of values.

   How test will be performed: the same input image will be used for calculation inVTK, and the output optimal threshold values from VTK will be used as control value. The percentage of difference between the outputs from MISEG and VTKwill be calculated. It will be performed by the test team with assistance from JUnit, and will be repeated multiple times.

5.1.3 Output

MISEG shall output segmentation image. In this test, given a valid input, an output image is expected.

Output Test

1. Existence of output file

   Control: Automatic

   Initial State: MISEG is started and running, a valid input image such as valid.dcm or valid.dcm30 is taken, user has chosen which threshold method to use, optimal threshold value(s) have been displayed.

   Input: User shall start the next step.

   Output: MISEG shall output a file, and this output file shall be found with correct filename extension .bmp.

   Test Case Derivation: successfully detect the users' choice and provide an output file with valid filename extension.

   How test will be performed: it will be performed by the test team with help of JUnit. The test team should load the whole project with an IDE and run the JUnittest file /test/OutputFileWriteTest.java. [This can be automated. —SS] [adjusted —Author]

5.1.4 Output verification

MISEG shall guarantee that the output file is the same resolution as the input file, and shall verify that the output data are valid and meet the format standards. The output image must be 2D 8-bit grayscale image and the pixel format must be the byte image, where the feature value must be the gray intensity value stored as an 8-bit integer giving a range of possible values from 0 to 255.

Part of the test for nonfunctional requirements including Correctness and Verifiability in Section 5.2.2 is also done here.

In this test, output image will be cross-checked with results from other software such as VTK.

Output verification test

1. Valid file format of output file

   Control: Automatic

   Initial State: MISEG is started and running, a valid input image a valid input image such as valid.dcm or valid.dcm30 is taken, user has chosen which threshold method to use, optimal threshold value(s) have been displayed, a file has been output.

   Input: the input and output file.

   Output: the result of whether the output image is an 8-bit grayscale image with the same resolution as the input file.

   Test Case Derivation: successfully output file with correct filename extensions.How test will be performed: it will be performed by the test team with the help of JUnit and OpenCV Java library. The test team should load the whole project with an IDE and run the JUnit test file /test/OutputFormatTest.java, which is built using OpenCV Java library for image reading and JUnit for testing. [This test can also be automated. That should be your goal. I don't think it will be that difficult if you use the right library. (Steven Palmer's chemical speciation example from one of the recent years used a library to do this.) —SS] [adjusted —Author]

2. Correctness of output file

   [What documented here is the very trivial way of comparing the output with the control image. More sophisticated and accurate method should be add to here later —Author]

   Control: Automatic

   Initial State: MISEG is started and running, a valid input image a valid input image such as valid.dcm or valid.dcm30 is taken, user has chosen which threshold method to use, optimal threshold value(s) have been displayed, a file has been output.

   Input: the output file.

   Output: the percentage of pixel value difference between this file and the output file from VTK using the same threshold value(s).

Test Case Derivation: successfully compare the output file with control image from VTK.

How test will be performed: it will be performed by the test team with the help of JUnit and OpenCV Java library. The test team should load the whole project with an IDE and run the JUnit test file /test/OutputCorrectnessTest.java, which is built using OpenCV Java library for image reading and JUnit for testing. The resolution and pixel values of input and output images will be compared, and thedifference will be measured.

5.2 Tests for Nonfunctional Requirements

The functional requirements described in the SRS Dong (2019c).

[Really just referencing the section is enough, since reproducing the text creates a maintainability problem. —SS] [I cited SRS here, and deleted the copied words —Author]

All the qualities of MISEG will be tested in the following subsections. Most qualities can be measured by the grade sheet in tables, such as Table 1 for Installability. In some cases a superscript $*$ isused to indicate that a response of this type should be accompanied by explanatory text. For instance, if problems were caused by uninstall, the reviewer should note what problems were caused. An (I) precedes the test case orquestion description when its measurement requires a successful installation Smith et al. (2018).

5.2.1 Installability

Installability is the degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment ISO/IEC (2011).

Installability test

1. Installation and uninstallation on Windows system

   Type: Manual

   Initial State: a virtual machine of fresh Windows 10 operating system

   Input/Condition: MISEG installation package, install command and uninstallcommand after installation

   Output/Result: the degree of effectiveness and efficiency with which MISEGcan be successfully installed and/or uninstalled

   How test will be performed: Installability can be measured by the grade sheet inTable 1. It will be performed by the test team manually.

2. Installation and uninstallation on Mac system

   Type: Manual

   Initial State: a virtual machine of fresh macOS 10.14 operating system

Questions	Sets of Answers
Are there installation instructions?	{yes,no}
Are the installation instructions linear?	{yes, no, N/A}
Is there something in place to automate the installation?	{yes*, no}
Is there a means given to validate the installation?	{yes*, no}
How many steps were involved in the installation?	$\mathbb{N}$
How many software packages need to be installed?	$\mathbb{N}$
Run uninstall, if available. Any obvious problems?	{yes*, no, n/a}
Overall Impression	{1 .. 10}

Table 1: Installability Grade Sheet Smith et al. (2018)

Input/Condition: MISEG installation package, install command and uninstallcommand after installation

Output/Result: the degree of effectiveness and efficiency with which MISEGcan be successfully installed and/or uninstalled

How test will be performed: same as Installation and uninstallation on Windows system.

3. Installation and uninstallation on Linux system

   Type: Manual

   Initial State: a virtual machine of fresh Ubuntu Linux 18.04 operating system

   Input/Condition: MISEG installation package, install command and uninstallcommand after installation

   Output/Result: the degree of effectiveness and efficiency with which MISEGcan be successfully installed and/or uninstalled

   How test will be performed: same as Installation and uninstallation on Windows system.

[I agree that installability needs to be done with manual tests. —SS]

5.2.2 Correctness and Verifiability

The term correctness is often mentioned as a degree to which software meets the requirement specification IEEE (1990). Verifiability is sometimes referred to as testability, since the focus is on measuring how easily the properties of a software can be checked or proven Smith et al. (2018).

Correctness is tested in the tests for functional requirements included in Section 5.1.2 and 5.1.4. [Yes I agree with assessing correctness, but youaren't really measuring verifiability.

Your tests could be very difficult to perform. Just running tests just not prove that the software is verifiable. —SS] [Added more contents as follows for verifiability —Author]

Correctness and verifiability will also be measured by the grade sheet in Table 2. It will be performed by the test team manually.

Questions	Sets of Answers
Are external libraries used?	{yes*, no, unclear}
Does the community have confidence in this library?	{yes, no, unclear}
Any reference to the requirements specifications of the program?	{yes*, no, unclear}
What tools or techniques are used to build confidence of correctness?	string
(I) If there is a getting started tutorial, is the output as expected?	{yes, no*, n/a}
Overall impression?	{1 .. 10}

Table 2: Correctness and Verifiability Grade Sheet Smith et al. (2018)

5.2.3 Robustness

A program is robust if it behaves "reasonably", even in circumstances that were not anticipated in the requirements specification - for example, when it encounters incorrect input data or some hardware malfunction Ghezzi et al. (1991).

It will be measured by the grade sheet in Table 3. It will be performed by the test team manually.

Questions	Sets of Answers
(I) Does the software handle garbage input reasonably?	{yes, no*}
(I) For any plain text input files, if all new lines are replaced with new lines and carriage returns, will the software handle this gracefully?	{yes, no*, n/a}
Overall impression?	{1 .. 10}

Table 3: Robustness Grade Sheet Smith et al. (2018)

[Technically, this isn't robustness testing if your requirements explicitly say what to do in these input error cases. I cannot remember whether your requirements are specific about these scenarios. —SS] [I changed it to your questionnaire. Before finding an better method, I thinkit's the best way of measuring. —Author]

5.2.4 Usability

Usability is the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of

use ISO/IEC (2011).

It will be measured by 4 qualities: Learnability, Memorability, Efficiency and Satisfaction.

Usability Test

1. Learnability

   Type: Manual

   Initial State: MISEG is started and running.

   Input/Condition: A new user to MISEG is asked to learn the software by himself/herself and to accomplish the task of inputting image, choosing the multiple-threshold calculation method and outputting image. If user guide exists, it shall be provided. [I like this, but I think you can be more specific. What calculation method are they asked to choose? —SS] [I specified to choose from one of the methods — Author]

   Output/Result: time to completion, number of misoperations and percentage of success will be measured.

   How test will be performed: results will be recorded in the grade sheet in Table8. It will be performed by the test team manually.

2. Memorability

   Type: Manual

   Initial State: MISEG is started and running.

   Input/Condition: 2 weeks after the new user finish the Learnability test, he or she shall be asked to accomplish the same tasks again. No guide shall be provided.

   Output/Result: time to completion, number of misoperations and percentage of success will be measured, and percentage of improvements will be calculated

   How test will be performed: results will be recorded in the grade sheet in Table8. It will be performed by the test team manually.

3. Efficiency

   Type: Manual

   Initial State: MISEG is started and running.

   Input/Condition: a proficient user to MISEG is asked to accomplish the task of inputting image, choosing calculation method and outputting image. No guide shall be provided.

   Output/Result: time to completion and number of misoperations will be measured

   How test will be performed: results will be recorded in the grade sheet in Table8. It will be performed by the test team manually.

4. Satisfaction

   Type: Manual

   Initial State: MISEG is started and running.

   Input/Condition: a user to MISEG is asked to answer additional questions and provide a overall satisfaction grade to the software

   Output/Result: answer from the user

   How test will be performed: results will be recorded in the grade sheet in Table8. It will be performed by the test team manually.

5.2.5 Maintainability

Maintainability is the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers ISO/IEC (2011).

Maintainability Test

1. Development process check

   Type: Manual

   Initial State: N/A

   Input/Condition: Testers need to review the whole development process, and answer questions related to the ease of maintainability.

   Output/Result: answers and grades to the table.

   How test will be performed: testers shall check the GitHub repo of MISEG for the effectiveness of version control and issue tracking on the software and the documents; they shall check the existence and completeness of the documents such as SRS, SysVnVPlan, MG, MIS. Results will be recorded in the grade sheet inTable 4. It will be performed by the test team manually.

5.2.6 Portability

Portability is the degree of effectiveness and efficiency with which a system, product or component can be transferred from onehardware, software or other operational or usage environment to another ISO/IEC (2011).

Portability Test

1. Portability on Windows system

   Type: Manual

Questions	Sets of Answers
Is there a history of multiple versions of the software?	{yes, no, unclear}
Is there any information on how code is reviewed, or how to contribute?	{yes*, no}
Is there a changelog?	{yes, no}
What is the maintenance type?	{corrective, adaptive, perfective, unclear}
What issue tracking tool is employed?	{Trac, JIRA, Redmine, e-mail, discussion board Git, none, unclear}
Are the majority of identified bugs fixed?	{yes, no*, unclear}
Which version control system is in use?	{svn, cvs, git, github, unclear}
Is there evidence that maintainability was considered in the design?	{yes*, no}
Are there code clones?	{yes*, no, unclear}
Overall impression?	{1 .. 10}

Table 4: Maintainability Grade Sheet Smith et al. (2018)

Initial State: MISEG has been successfully installed on a virtual machine of fresh Windows 10 operating system

Input/Condition: operate the basic functions of the software

Output/Result: the degree of effectiveness and efficiency with which MISEGcan be operate on this platform

How test will be performed: Portability can be measured by the grade sheet in Table 5. It will be performed by the test team manually.

2. Portability on Mac system

   Type: Manual

   Initial State: MISEG has been successfully installed on a virtual machine of fresh macOS 10.14 operating system

   Input/Condition: operate the basic functions of the software

   Output/Result: the degree of effectiveness and efficiency with which MISEG can be operate on this platform

   How test will be performed: Portability can be measured by the grade sheet in Table 5. It will be performed by the test team manually.

3. Portability on Linux system

Questions	Sets of Answers
(I)What platforms is the software advertised to work on?	{Windows, Linux, macOS, Android, Other OS}
(I)Is there any compromise to functional or nonfunctional requirements by running on this platform?	{yes*, no}
Are special steps taken in the source code to handle portability?	{yes*, no, n/a}
Is portability explicitly identified as NOT being important?	{yes, no}
Convincing evidence that portability has been achieved?	{yes*, no}
Overall impression?	{1 .. 10}

Table 5: Portability Grade Sheet Smith et al. (2018)

Type: Manual

Initial State: MISEG has been successfully installed on a virtual machine of fresh Ubuntu Linux 18.04 operating system

Input/Condition: operate the basic functions of the software

Output/Result: the degree of effectiveness and efficiency with which MISEG can be operate on this platform

How test will be performed: Portability can be measured by the grade sheet in Table 5. It will be performed by the test team manually.

5.2.7 Understandability

Understandability measures the ease with which a new developer can understand the design and source code. Good understandability contributes to maintainability, and provides critical information for verifiability Smith et al. (2018).

Understandability Test

1. Code review

   Type: Manual

   Initial State: the development of MISEG is completed and the source code is accessible

   Input/Condition: review the source code

   Output/Result: the ease with which a new developer can understand the source code

   How test will be performed: After reading part or all of the source code, Understandability can be measured by the grade sheet in Table 6. It will be performed by the test team manually.

Questions	Set of Answers
Consistent indentation and formatting style?	{yes, no, n/a}
Explicit identification of a coding standard?	{yes*, no, n/a}
Are the code identifiers consistent, distinctive, and meaningful?	{yes, no*, n/a}
Are constants (other than 0 and 1) hard-coded into the program?	{yes, no*, n/a}
Comments are clear, indicate what is being done, not how?	{yes, no*, n/a}
Is the name/URL of any algorithms used mentioned?	{yes, no*, n/a}
Parameters are in the same order for all functions?	{yes, no*, n/a}
Is code modularized?	{yes, no*, n/a}
Descriptive names of source code files?	{yes, no*, n/a}
Is a design document provided?	{yes*, no, n/a}
Overall impression?	{1 .. 10}

Table 6: Understandability Grade Sheet Smith et al. (2018)

2. MG and MIS review

   Type: Manual

   Initial State: the development of MISEG is completed and the MG and MIS documents are accessible

   Input/Condition: review the MG and MIS documents

   Output/Result: the ease with which a new developer can understand the design

   How test will be performed: After reading the MG and MIS, Understandability can be measured by the grade sheet in Table 6. it will be performed by the test team manually.

[Good work with the nonfunctional requirements. I hope you are able to do at least some of these measurements. I think we would learn from collecting the data, and this could be helpful for your MEng project. —SS]

5.3 Traceability Between Test Cases and Requirements

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed,the items in the column of that component that are marked with an "X" may have to be modified as well. Table 7 shows the dependencies between the test cases and the requirements. [Include some text to introduce the table. —SS] [Added text similar to the SRS —Author]

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13
5.1.1	X								X				
5.1.2			X				X	X					
5.1.3					X								
5.1.4		X		X			X	X					
5.2.1						X							
5.2.2							X	X					
5.2.3									X				
5.2.4										X			
5.2.5											X		
5.2.6												X	
5.2.7													X

Table 7: Traceability Matrix showing the connections between requirements and tests

References

Isaac N. Bankman. Preface. In Isaac N. Bankman, editor, *Handbook of Medical Imaging*, Biomedical Engineering, pages xi – xii. Academic Press, San Diego, 2000. ISBN 978-0-12-077790-7. doi: https://doi.org/10.1016/B978-012077790-7/50001-1. URL http://www.sciencedirect.com/science/article/pii/B9780120777907500011.

Ao Dong. Module guide for miseg. 2019a. URL https://github.com/Ao99/MISEG/blob/master/docs/Design/MG/MG.pdf.

Ao Dong. Module interface specification for miseg. 2019b. URL https://github.com/Ao99/MISEG/blob/master/docs/Design/MIS/MIS.pdf.

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

Mohamad Forouzanfar, Nosratallah Forghani, and Mohammad Teshnehlab. Parameter optimization of improved fuzzy c-means clustering algorithm for brain mr image segmentation. *Engineering Applications of Artificial Intelligence*, 23(2):160 – 168, 2010. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2009.10.002. URL http://www.sciencedirect.com/science/article/pii/S095219760900150X.

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of software engineering*. Prentice Hall PTR, 1991.

IEEE. Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84, Dec 1990. doi: 10.1109/IEEESTD.1990.101064. URL https://ieeexplore.ieee.org/document/159342.

ISO/IEC. Iso/iec25010 - systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models. Standard, International Organization for Standardization, Mar 2011. URL https://pdfs.semanticscholar.org/57a5/b99eceff9da205e244337c9f4678b5b23d25.pdf.

Spencer Smith, Zheng Zeng, and Jacques Carette. Seismology software: state of the practice. *Journal of Seismology*, 22, 02 2018. doi: 10.1007/s10950-018-9731-3.

6 Appendix

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

6.2 Usability Grade Sheet

This grade sheet is used for test in Section 5.2.4

Test ID	Question/test detail	Anser/result
(I)Learnability: new users	Time to completion	Seconds
	Number of misoperations	$\mathbb{N}$
	Percentage of success	Percentage
(I)Memorability second-time users	Time to completion	Seconds
	Percentage of improvement	Percentage
	Number of misoperations	$\mathbb{N}$
	Percentage of improvement	Percentage
	Percentage of success	Percentage
	Percentage of improvement	Percentage
(I)Efficiency: proficient users	Time to completion	Seconds
	Number of misoperations	$\mathbb{N}$
(I)Satisfaction: every user	Do the operations fit to human nature and your intuition?	{yes, no*}
	Does it support your language?	{yes, no*}
	Can you understand the descriptions easily	{yes, no*}
	Does it give a clear explanation when an error occurs?	{yes, no*}
	Have you noticed any hot keys?	{yes*, no}
	Do you think any hot key need to be added?	{yes*, no}
	Do you think undo or redo function is missing during any step?	{yes*, no}
	Do you think any other function for convenience need to be added? Such as auto-fill, repeat and a record for all the steps.	{yes*, no}
	Overall satisfaction	{1 .. 10}

Table 8: Usability Grade Sheet