Test Report: System Verification and Validation Plan for MISEG

Ao Dong

December 17, 2019

1 Revision History

Date	Version	Notes
Dec 17	1.0	Initial Draft

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

For the other symbols, abbreviations and acronym, see SRS Documentation at https://github.com/Ao99/MIA/blob/master/docs/SRS/SRS.pdf

Contents

List of Tables

List of Figures

This document is a report on the results of a testing suite for MISEG. Detailed descriptions of the tests executed can be found in SystVnVPlan Dong (2019c).

3 Functional Requirements Evaluation

4 Nonfunctional Requirements Evaluation

4.1 Usability

Questions	Sets of Answers	Answers
Are there installation instructions?	{yes,no}	yes
Are the installation instructions linear?	{yes, no, N/A}	yes
Is there something in place to automate the installation?	{yes*, no}	no
Is there a means given to validate the installation?	{yes*, no}	no
How many steps were involved in the installation?	$\mathbb{N}$	5
How many software packages need to be installed?	$\mathbb{N}$	3
Run uninstall, if available. Any obvious problems?	{yes*, no, n/a}	n/a
Overall Impression	{1 .. 10}	6

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

4.2 Correctness and Verifiability

Questions	Sets of Answers	Answers
Are external libraries used?	{yes*, no, unclear}	yes: Java library dcm4che.
Does the community have confidence in this library?	{yes, no, unclear}	yes
Any reference to the requirements specifications of the program?	{yes*, no, unclear}	yes: SRS Dong (2019b)
What tools or techniques are used to build confidence of correctness?	string	JUnit, PMD, SystVnVPlan Dong (2019c), UnitVnVPlan **?**
(I) If there is a getting started tutorial, is the output as expected?	{yes, no*, n/a}	yes
Overall impression?	{1 .. 10}	10

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

4.3 Robustness

Questions	Sets of Answers	Answers
(I) Does the software handle garbage input reasonably?	{yes, no*}	yes
(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}	n/a
Overall impression?	{1 .. 10}	8

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

4.4 Usability

[I didn't have a tester to do the new user test, so some results are faked. —Author] Percentages of improvements in indicators which are the less the better (such as time and number of misoperations) are measured by the following equation:

$$\text{percentage of improvement} = \frac{\text{first-time result} - \text{second-time result}}{\text{first-time result}} \times 100\%$$

Percentages of improvements in indicators which are the more the better (such as success rate) are measured by the following equation:

$$\text{percentage of improvement} = \frac{\text{second-time result} - \text{first-time result}}{\text{first-time result}} \times 100\%$$

Test ID	Question/test detail	Set of Answers	Answers
(I)Learnability: new users	Time to completion	Seconds	60
	Number of misoperations	$\mathbb{N}$	10
	Success rate	Percentage	80%
(I)Memorability second-time users	Time to completion	Seconds	40
	Percentage of improvement	Percentage	33.3%
	Number of misoperations	$\mathbb{N}$	8
	Percentage of improvement	Percentage	20%
	Success rate	Percentage	90%
	Percentage of improvement	Percentage	12.5%
(I)Efficiency: proficient users	Time to completion	Seconds	30
	Number of misoperations	$\mathbb{N}$	2
(I)Satisfaction: every user	Do the operations fit to human nature and your intuition?	{yes, no*}	yes
	Does it support your language?	{yes, no*}	yes
	Can you understand the descriptions easily	{yes, no*}	yes

Does it give a clear explanation when an error occurs?	{yes, no*}	yes
Have you noticed any hot keys?	{yes*, no}	no
Do you think any hot key need to be added?	{yes*, no}	no
Do you think undo or redo function is missing during any step?	{yes*, no}	yes: there can be undo and redo for input functions
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}	yes: a record of steps can be added
Overall satisfaction	{1 .. 10}	8

Table 4: Usability Grade Sheet

4.5 Maintainability

Questions	Sets of Answers	Answers
Is there a history of multiple versions of the software?	{yes, no, unclear}	no
Is there any information on how code is reviewed, or how to contribute?	{yes*, no}	yes: SystVnVPlan Dong (2019c)
Is there a change log?	{yes, no}	yes
What is the maintenance type?	{corrective, adaptive, perfective, unclear}	corrective
What issue tracking tool is employed?	{Trac, JIRA, Redmine, e-mail, discussion board, SourceForge, Git, none, unclear}	Git
Are the majority of identified bugs fixed?	{yes, no*, unclear}	yes
Which version control system is in use?	{svn, cvs, git, github, unclear}	github
Is there evidence that maintainability was considered in the design?	{yes*, no}	yes: SRS Dong (2019b)
Are there code clones?	{yes*, no, unclear}	yes: on GitHub
Overall impression?	{1 .. 10}	7

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

4.6

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

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

4.7 Understandability

Questions	Set of Answers	Answers
Consistent indentation and formatting style?	{yes, no, n/a}	yes
Explicit identification of a coding standard?	{yes*, no, n/a}	no
Are the code identifiers consistent, distinctive, and meaningful?	{yes, no*, n/a}	yes
Are constants (other than 0 and 1) hard-coded into the program?	{yes, no*, n/a}	no: there is a Constants module
Comments are clear, indicate what is being done, not how?	{yes, no*, n/a}	yes

Is the name/URL of any algorithms used mentioned?	{yes, no*, n/a}	yes
Parameters are in the same order for all functions?	{yes, no*, n/a}	yes
Is code modularized?	{yes, no*, n/a}	yes
Descriptive names of source code files?	{yes, no*, n/a}	yes
Is a design document provided?	{yes*, no, n/a}	yes: MIS Dong (2019a)
Overall impression?	{1 .. 10}	10

Table 7: Understandability Grade Sheet Smith et al. (2018)

5 Comparison to Existing Implementation

This section will not be appropriate for every project.

6 Unit Testing

7 Changes Due to Testing

8 Automated Testing

9 Trace to Requirements

10 Trace to Modules

11 Code Coverage Metrics

References

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

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

Ao Dong. System verification and validation plan for miseg. 2019c. URL https://github.com/Ao99/MISEG/blob/master/docs/VnVPlan/SystVnVPlan/SystVnVPlan.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.