Test Report: Diagnose

Andrea Clemeno

December 15, 2020

1 Revision History

Date	Version	Notes
December 10	1.0	Initial pdf
December 12	1.1	Notes
December 15	2.0	Final Documentation of VnV Report

2 Symbols, Abbreviations and Acronyms

The table that follows summarizes the symbols used in this document along with their descriptions. The symbols are listed in alphabetical order. In addition, all symbols, abbreviations, and acronyms recorded in the SRS and VnV plan for Diagnose apply to this document. Documents can be found in Clemeno (2020) and Clemeno (2020a) respectively.

symbol	**description**
SRS	Software Requirements Specification
T	Test
VnV	Verification and Validation

Table 1: Table of Symbols, Abbreviations and Acronyms

Contents

List of Tables

List of Figures

This document will present the results and findings for the tests outlined in the VnV plan of Diagnose (Clemeno, 2020a). The evaluation of functional and nonfunctional requirements with system testing methods can be found in sections 3 and 4 respectively. In addition to system testing results, unit testing for some modules of Diagnose in section 6. The validation of the Diagnose software will be compared to existing data for viral load elimination from Stafford et al. (2000) in section 5. Moreover, changes to the code with respect to these evaluations will be mentioned in section 7. Lastly, the automated testing evaluations will be mentioned in section 8.

3 Functional Requirements Evaluation

3.1 Input Verification

These input verification tests were completed using pytest and the methods mentioned in the VnV plan of Diagnose.Documentation of test results can be found on the github repository: andreamclemeno/CAS741-Diagnose/test.

Test	Test Description	Verification Complete
T-1	Expected Input	✓
T-2	Invalid Input: N_o	✓
T-3	Invalid Input: N_t	✓
T-4	Invalid Inputs: N_o, N_t	✓
T-5	Invalid Input: t_t	✓
T-6	Invalid Input: t_d	✓
T-7	Invalid Inputs: t_t, t_p	✓

Table 2: Functional Requirements Evaluation: Input Verification

3.2 Output Correctness

These output correctness was verified using pytest and the methods mentioned in the VnV plan of Diagnose.Documentation of test results can be found on the github repository: andreamclemeno/CAS741-Diagnose/test.

Test	Test Description	Verification Complete
T-8	Expected Input	✓

Table 3: Functional Requirements Evaluation: Output Correctness

4 Nonfunctional Requirements Evaluation

4.1 Understandability

A successful test is specified by the automatic verification of no error messages in the linting process. The static analysis run in the Spyder IDE found no error messages seen in table 4. Overall, the verification of understandability has been successful as seen in table 5. Documentation of test results can be found on the github repository: andreamclemeno/CAS741-Diagnose/test.

Module	Conventions	Refactoring	Warnings	Errors	Global Eval.
Control.py	8	0	0	0	7.14
Calculations.py	9	0	0	0	5.50
InputFormat.py	5	0	0	0	8.75
InputParameters.py	3	1	1	0	-15.00
InputConstraints.py	17	0	0	0	5.50
OutputFormat.py	5	0	0	0	6.88

Table 4: Linting Results

Test	Test Description	Verification Complete
T-9	Linting for Diagnose Modules	✓

Table 5: Non-functional Requirements Evaluation: Understandability

4.2 Reliability

A successful test is specified by the automatic verification of a total run time of less than 0.1 seconds. As seen in Figure 1, the run time is 0.007 seconds and passes the reliability test. Overall, the verification of reliability has been successful as seen in table 6. Documentation of test results can be found on the github repository: andreamclemeno/CAS741-Diagnose/test.

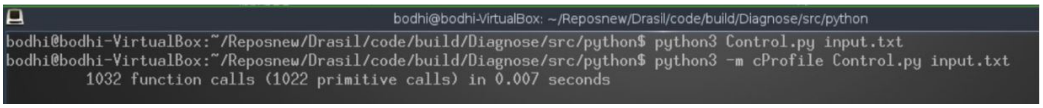

Figure 1: Reliability Test: Dynamic Analysis using cProfile

Test	Test Description	Verification Complete
T-10	Reliability	✓

Table 6: Non-functional Requirements Evaluation: Reliability

4.3 Maintainability

Testing for maintainability involves the manual verification of traceability from tests to requirements seen in table 14 and traceability from tests to modules seen in table 15. From these tables, it is seen that each requirement and module is tested. In addition, traceability matrices in the SRS and VnV document are completed and demonstrate traceability to ensure maintainability of the software (Clemeno, 2020b). Overall, the verification of maintainability has been successful as seen in table 7.

Test	Test Description	Verification Complete
T-11	Maintainability	✓

Table 7: Non-functional Requirements Evaluation: Maintainability

4.4 Portability

A successful test is specified by the completion of the make command. As seen in table 8, the make is completed for Windows OS and Linux OS, and passes the portability test. Overall, the verification of portability has been successful as seen in table 9.

OS	Make Completed
Windows OS	✓
Linux OS	✓

Table 8: Portability test: OS and Make Completion

Test	Test Description	Verification Complete
T-12	Portability	✓

Table 9: Non-functional Requirements Evaluation: Portability

4.5 Verifiability

Successful verification testing is indicated by the successful manual verification of tests T-1 to T-15. The verifiability of functional requirements is determined by the verification of functional requirement system testing seen in section 3 and unit testing seen in section 6. The verifiability of nonfunctional requirements is determined by the verification of non-functional requirement system testing seen in section 4. Overall, the verification of verifiability has been successful as seen in table 10.

Test	Test Description	Verification Complete
T-13	Verifiability of Functional Requirements	✓
T-14	Verifiability of Nonfunctional Requirements	✓

Table 10: Non-functional Requirements Evaluation: Verifiability

5 Comparison to Existing Implementation

The validation of Diagnose was completed by comparing the outputs of the software to several cases from scientific study called Modeling plasma virus concentration during primary HIV infection seen in The Journal of Theoretical Biology (Stafford et al., 2000). For more documentation of validation results can be found on the github repository: andreamclemeno/CAS741-Diagnose/test.

The validation was completed using patient 1-4 and 6-10 data from Stafford et al. (2000) seen in table 11. For simplicity, patient 1-4 and 6-10 will be referred to as patients 1-7.

Patient	N_o $(\frac{mol}{mL})$	N_t $(\frac{mol}{mL})$	t_t **(d)**	t_p **(d)**
1	94760000	70620000	2	8
2	28400000	21600000	1	9
3	148500000	70160000	5	7
4	239860000	33720000	5	8
5	242790000	220040000	3	7
6	35540000	14680000	4	11
7	962280000	783000000	1	5

Table 11: Software Validation: Patient 1-9 data

The data from table 11 is compared with data from the Diagnose software and the theoretical error is found and presented in table 12. Additionally, andreamclemeno/CAS741-Diagnose/test contains graphs for easier comparison similar to in figure 2.

Patient	**Strafford** N_p $(\frac{mol}{mL})$	**Diagnose** N_p $(\frac{mol}{mL})$	*error* **(d)**
1	144000	2.2094e-16	6.5174e+23
2	143000	1.4870e-07	9.6165e+15
3	564000	7.9327e-13	7.1097+21
4	340600	0.3354	1.0152e+10
5	1134300	4.8829e-85	2.3229e+94
6	100900	8.6790e-15	1.1625e+23
7	7158100	0.0282	2.5339e+11

Table 12: Software Validation: Viral Load from Stafford et al. (2000) vs Diagnose

6 Unit Testing

6.1 Calculations Module

Unit testing for Calculations.py was completed using unittest in the Spyder IDE. Exceptions were raised if the module was not behaving properly. For T-14, the test cases displayed the OK indicator classifying the test as successful as seen for testcase7b.txt in Figure 3. Overall, the unit test cases were successful as seen in table 13. Documentation of test results can be found on the github repository: andreamclemeno/CAS741-Diagnose/test.

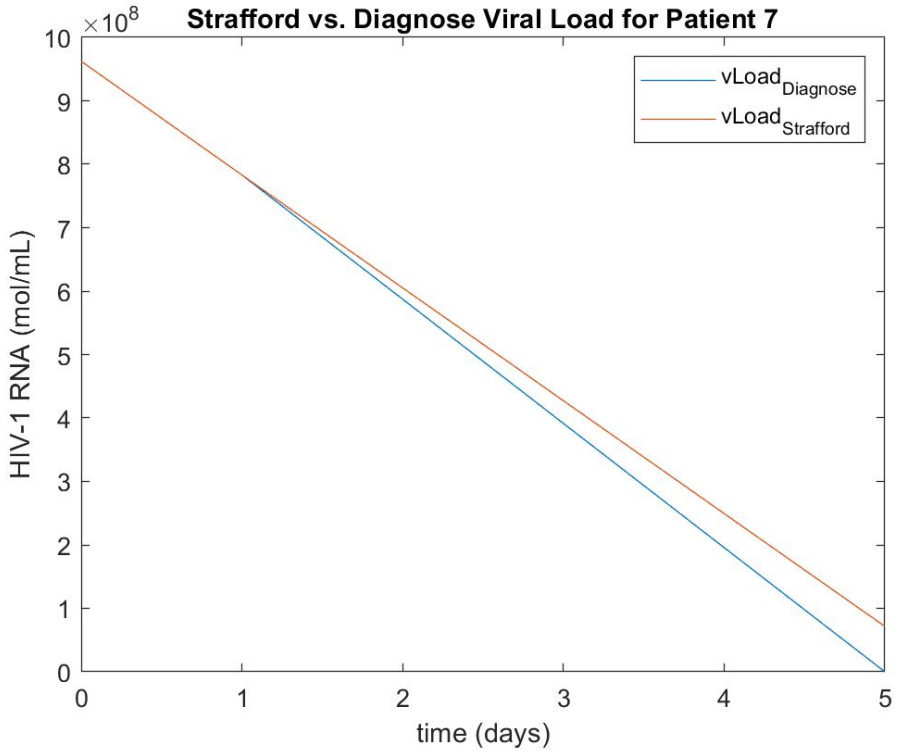

Figure 2: Viral Load from Stafford et al. (2000) vs Diagnose: Patient 7

Test	Test Description	Verification Complete
T-15	Test for Calculations.py	✓

Table 13: Unit testing Evaluation

7 Changes Due to Testing

No major changes were implemented due to testing. However, during the linting process, the variable of elimination rate was changed from λ to k as ASCII characters are did not follow coding standards.

Figure 3: Unit Test Example: Calculations.py, testcase7b.txt

8 Automated Testing

The automated testing tool used by Diagnose is Travis CI. The build can be found on the github repository: Documentation of test results can be found on the github repository: andreamclemeno/Drasil.

9 Trace to 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 14 shows which test cases are supporting which requirements.

	R1	R2	R3	R4	R5	NF1	NFR2	NFR3	NFR4	NFR5
T-1	X	X	X	X	X					
T-2	X	X								
T-3	X	X								
T-4	X	X								
T-5	X	X								
T-6	X	X								
T-7	X	X								
T-8	X	X	X	X	X					
T-9						X				
T-10						X				
T-11							X			
T-12								X		
T-13									X	
T-14										X
T-15			X							

Table 14: Traceability Matrix Showing the Connections Between Requirements and Tests

10 Trace to Modules

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 15 shows which test are supporting which modules.

	Calculations.py	Control.py	InputConstraints.py	InputFormat.py	InputParameter.py	OutputFormat.py
T-1	X	X	X	X	X	X
T-2						
T-3						
T-4						
T-5						
T-6						
T-7						
T-8	X	X	X	X	X	X
T-9						
T-10						
T-11						
T-12						
T-13						
T-14						
T-15			X			

Table 15: Traceability Matrix Showing the Connections Between Modules and Tests

References

A. Clemeno. Software requirements specification for diagnosis-aids: Medical diagnosis prediction tool for acquired immunodeficiency syndrome (aids). 2020.

Andrea Clemeno. Verification and validation plan, Nov 2020a. URL https://github.com/andreamclemeno/CAS741-Diagnose/blob/master/docs/VnVPlan/BlankProjectTemplate_docs_VnVPlan_VnVPlan.pdf.

Andrea Clemeno. Software requirements specification for diagnose.tex, December 2020b. URL https://github.com/andreamclemeno/Drasil/tree/master/code/stable/diagnose/SRS.

MAX A. Stafford, LAWRENCE COREY, YUNZHEN CAO, ERIC S. DAAR, DAVID D. HO, and ALAN S. PERELSON. Modeling plasma virus concentration during primary hiv infection. *Journal of Theoretical Biology*, 203(3):285 – 301, 2000. ISSN 0022-5193. doi: https://doi.org/10.1006/jtbi.2000.1076. URL http://www.sciencedirect.com/science/article/pii/S0022519300910762.