Verification and Validation Report: Measuring Microstructure Changes During Thermal Treatment

Team #30, ReSprint
Edwin Do
Joseph Braun
Timothy Chen
Abdul Nour Seddiki
Tyler Magarelli

March 8, 2023

1 Revision History

Date	Developer	Change
Mar 8 2023	Edwin Do	Added usability test results
Mar 8 2023	Edwin Do	Added Traceability matrices
Mar 8, 2023	Joseph Braun	Added Sections 5, 7, 8
Mar 8, 2023	Joseph Braun	Added Reflection

2 Symbols, Abbreviations and Acronyms

symbol	description
T	Test

[symbols, abbreviations or acronyms – you can reference the SRS tables if needed —SS]

Contents

List of Tables

List of Figures

This document ... Section 11 Code Coverage Metrics is removed.

3 Functional Requirements Evaluation

4 Nonfunctional Requirements Evaluation

¡¡¡¡¡¡¡ HEAD In the section ======= In the section, the test results for the most important non functional requirements are shown. The team decided to focus testing primarily on Usability and Performance since they are key qualities to the success of this project. ¿¿¿¿¿¿¿ 01c88ed942368a0c67af8051fe6a8b782f30397e

4.1 Usability

The table below shows the results of our usability tests based on the tests in the V&V plan based on the requirements mentioned in the SRS. Each requirement can be traced to multiple unit tests and the usability survey used can be found in the Appendix.

Usability Tests				
Test Requirement	Related Unit Tests	Description	Expected Result	Result
NF-UT1	AF	Completing tasks without additional assistance.	User will be complete tasks successfully	PASS
NF-UT2	AX	Interact with interace to modify parameters.	User will be able to modify the parameters accurately and quickly.	PASS
NF-UT3	AL	Completing all tasks with limited number of mistakes	User will be able to complete all tasks with *MAX_MISTAKE*	PASS
NF-UT3	AL	Verifying if interacting with the application previously improves ease of use (learnability)	User will be able to complete the tasks more quickly and accurately the second time	PASS
NF-UT5	AS	Verifying how calculations are performed is hidden	User will not know how calculations are performed after doing the set of tasks	PASS
NF-UT6	AS	Verifying appropriate application size upon installation	User will install application onto computer and verify that the application size is less than or equal to *MAX_SIZE*	PASS

4.2 Performance

¡¡¡¡¡¡¡ HEAD ======= The following is the list of Non-functional tests performed on the application to evaluate the performance of the application with respect to the test requirement. Each test will be mapped to unit tests that are related to the corresponding requirement.

Performance Tests				
Test Requirement	Related Unit Tests	Description	Expected Result	Result
NF-PT1	-	Checking the minimum sampling rate of the application.	The sampling rate of the application will be equal or greater than *MIN_SAMPLE_RATE*	PASS
NF-PT2	-	Checking the time required for parameters to reflect in the application.	The parameters will reflect in the application by within *TIME_ACCEPTED*	PASS
NF-PT3	-	Checking the significant digits used for calculations and display in the application.	The significant digits seen and used in the application is accurate to *ACCEPTED_SIGFIG*.	PASS
NF-PT4	-	Checking the up-time of the application during and after usage.	The application will have a up-time equal to or more than *MIN_UPTIME* after the user completes a set of tasks.	PASS

¿¿¿¿¿¿¿

01c88ed942368a0c67af8051fe6a8b782f30397e

4.3 etc.

5 Comparison to Existing Implementation

Below is an image of the exisiting implementation's GUI.

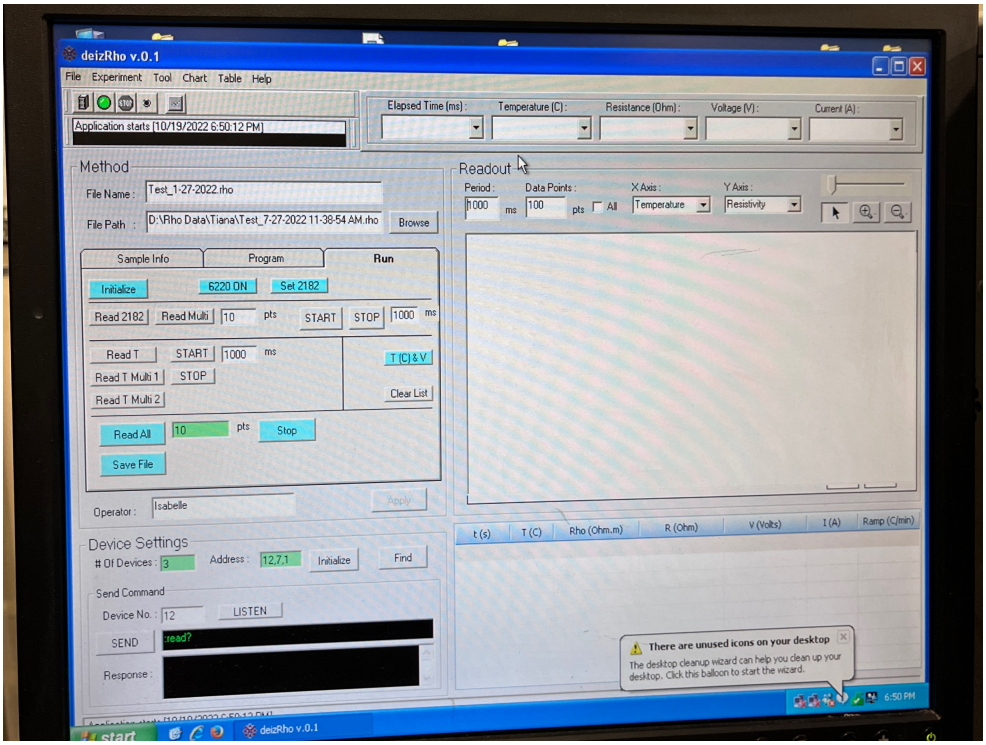

Figure 1: Previous software user interface design

The key elements of the existing implementation will also be included in our implementation. This is so that the application will be familiar to the user and as intuitive as possible. These elements are listed below:

- Method Panel: controls which device to read data from (temperature or voltage), and which file to save the data in
- Device Settings Panel: used to send SCPI commands directly to a device
- Readout: includes graphical output and listed output of relevant values (current, voltage, resistance, etc.)

The primary differences between the existing implementation and our implementaion are the appearance of the GUI and the option of remote access. The existing implementation was developed for Windows XP whereas our implementation is developed for Windows 10, which gives it an updated look. One of the stretch goals for this project is to enable remote access to the application to be able to monitor and stop experiments remotely. The existing implementation does not have this functionality.

6 Unit Testing

6.1 Calculation Module

Unit Tests for Calculation Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-C1	Testing the getResistance Method	From HardwareInput ADT: Voltage = 5, Current = 1	5.000	5.000	PASS
UT-C1	Testing the getResistance Method	From HardwareInput ADT: Voltage = 4, Current = 0.6	6.667	6.667	PASS
UT-C2	Testing the getResistance Method	From HardwareInput ADT: Voltage = -5, Current = 1	Invalid	Invalid	PASS
UT-C2	Testing the getResistance Method	From HardwareInput ADT: Voltage = -4, Current = 0.6	Invalid	Invalid	PASS
UT-C3	Testing the getResistivity Method	Resistance = 3, Area = 2.5, Length = 2	3.750	3.750	PASS
UT-C3	Testing the getResistivity Method	Resistance = 2, Area = 2.8, Length = 1	5.600	5.600	PASS
UT-C4	Testing the getResistivity Method	Resistance = A, Area = 2.8, Length = 0	Invalid	Invalid	PASS
UT-C4	Testing the getResistivity Method	Resistance = 1, Area = 0, Length = BC	Invalid	Invalid	PASS
UT-C5	Testing the calcResistance Method	Voltage = 2.6, Current = 2	1.300	1.300	PASS
UT-C5	Testing the calcResistance Method	Voltage = 3.6, Current = 2	1.800	1.800	PASS

Unit Tests for Calculation Module (Continued)					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-C6	Testing the calcResistance Method	Voltage = AB, Current = 0	Invalid	Invalid	PASS
UT-C6	Testing the calcResistance Method	Voltage = 0, Current = DR	Invalid	Invalid	PASS
UT-C7	Testing the calcResistivity Method	Resistance = 2, Area = 1.5, Length = 2	1.500	1.500	PASS
UT-C7	Testing the calcResistivity Method	Resistance = 1.8, Area = 1.3, Length = 1.5	1.560	1.560	PASS
UT-C8	Testing the calcResistivity Method	Resistance = AB, Area = 1, Length = 2	Invalid	Invalid	PASS
UT-C8	Testing the calcResistivity Method	Resistance = 2, Area = 2, Length = NT	Invalid	Invalid	PASS

6.2 User Input Validation Module

Unit Tests for User Input Validation Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-UI1	Testing the getUserInput Method	N/A	UserInput (SamplingRate: 60, SampleLength: 4, SampleWidth:2)	UserInput (SamplingRate: 60, SampleLength: 4, SampleWidth:2)	PASS
UT-UI2	Testing the validateFileData Method	FileName: Test, Date: 03/01/2023, Name: Test	TRUE	TRUE	PASS
UT-UI2	Testing the validateFileData Method	FileName: Output, Date: 03/01/2023, Name: John	TRUE	TRUE	PASS
UT-UI3	Testing the validateFileData Method	FileName: Output, Date: Someday, Name: Tester	FALSE	FALSE	PASS
UT-UI3	Testing the validateFileData Method	FileName: 0, Date: 03/01/2023, Name: 0	FALSE	FALSE	PASS

Unit Tests for User Input Validation Module (Continued)					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-UI4	Testing the validateSampleData Method	SamplingRate: 50, SampleLength:2, SampleWidth:5	TRUE	TRUE	PASS
UT-UI4	Testing the validateSampleData Method	SamplingRate: 60, SampleLength:1.5, SampleWidth:3	TRUE	TRUE	PASS
UT-UI5	Testing the validateSampleData Method	SamplingRate: -7, SampleLength:1.5, SampleWidth:3	FALSE	FALSE	PASS
UT-UI5	Testing the validateSampleData Method	SamplingRate: 60, SampleLength:-5, SampleWidth:3	FALSE	FALSE	PASS

6.3 Hardware Input Validation Module

Unit Tests for Hardware Input Validation Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-HI1	Testing the getHardwareInput Method	-	HardwareInput (Voltage: 5, Current: 3)	HardwareInput (Voltage: 5, Current: 3)	PASS
UT-HI2	Testing the validatePa-rameters Method	Voltage: 3.0 , Time: 5:31 PM , Current: 1.0	TRUE	TRUE	PASS
UT-HI2	Testing the validate-FileData Method	Voltage: 3.2 , Time: 5:45 PM , Current: 1.2	TRUE	TRUE	PASS
UT-HI3	Testing the validate-FileData Method	Voltage: 10 , Time: 5:48 PM , Current: NY	FALSE	FALSE	PASS
UT-HI3	Testing the validate-FileData Method	Voltage: YT , Time: 5:51 PM , Current: 0.5	FALSE	FALSE	PASS

6.4 FileOutput Module

Unit Tests for FileOutput Module					
Unit Test ID	Description	Input	Expected Output	Output	Result
UT-FO1	Testing the WriteUserInput Method	No Input	Invalid	Invalid	PASS
UT-FO1	Testing the WriteUserInput Method	Empty UserInput ADT	Invalid	Invalid	PASS
UT-FO2	Testing the WriteUserInput Method	UserInput ADT	None	None	PASS
UT-FO3	Testing the WriteSampleOutput Method	No Input	Invalid	Invalid	PASS
UT-FO4	Testing the WriteSampleOutput Method	HardwareInput ADT	File updated	File updated	PASS
UT-FO5	Testing the WriteSampleOutput Method	HardwareInput ADT	Invalid	Invalid	PASS
UT-FO6	Testing the WriteUserInput Method	UserInput ADT	Invalid	Invalid	PASS

7 Changes Due to Testing

Based on the feedback from our Rev 0 Demo, we found that our application failed some requirements tests. The major failures were the lack of two main features: the graphical output (which still did not display real data as of Rev 0 Demo) and the ability to save output data to a chosen file. While we did not need to "test" our application to know this, we still consider these to be failed tests, as our application failed to meet requirements set out by the project supervisor. These two features have been implemented for the final demo.

Test ID	Failure Observed	Change(s) Made
ST6	Graph only displays dummy data	Bug with displaying real-time data was fixed
ST7	There is no method to output data to a file	File system browser and writing to output file was added

8 Automated Testing

We achieved automated unit testing through the use of the NUnit testing framework in Visual Studio. NUnit is one of the most popular test frameworks used for running tests on a .NET project.

NUnit tests are setup by first creating a new project file in Visual Studio and adding it to the solution file for your project (in our case, the application). In the new project file, a new class is created. Each unit test we want to carry out is written as a method of the test class. Since the project file for the test class is inlcuded in the same solution file as our application, we are able to call the test methods from our main application to run the tests.

9 Trace to Requirements

Traceability Matrix to Non Functional Requirements			
Requirement Type	Requirement(SRS)	Test Requirement	Related Unit Tests
Non Functional	NFR-U1	NF-UT1	U
Non Functional	NFR-U2	NF-UT2	U
Non Functional	NFR-U3	NF-UT3	U
Non Functional	NFR-U4	NF-UT4	U
Non Functional	NFR-U5	NF-UT5	U
Non Functional	NFR-U6	NF-UT6	U

Traceability Matrix to Functional Requirements			
Requirement Type	Requirement(SRS)	Test Requirement	Related Unit Tests
Functional	FR1	FR-T1	U
Functional	FR2	FR-T2	U
Functional	FR3	FR-T3	U
Functional	FR4	FR-T4	U
Functional	FR5	FR-T5	U
Functional	FR6	FR-T6	U

¡¡¡¡¡¡¡ HEAD

======= The table below shows the traceability between each functional requirement from the SRS, and the test requirement in this report.

Table 1: Requirements Traceability

System Test	**Requirement**	**Plan**
ST1	FR1	FR-T1
ST2	FR2	FR-T2
ST3	FR3	FR-T3
ST4	FR4	FR-T4
ST5	FR5	FR-T5
ST6	FR6	FR-T6
ST7	FR7	FR-T7

¿¿¿¿¿¿¿ 01c88ed942368a0c67af8051fe6a8b782f30397e

10 Trace to Modules

References

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)

There were several differences between what we had planned for VnV and what we ended up carrying out. When completing the initial revision of VnV plan, we had not yet completed the Design documents (MG, MIS, System Design) and so we did not have a plan for unit/module testing, only for system testing. Our initial VnV plan included three main sections: SRS Verification, Design Verification, and Implementation Verification.

For SRS Verification, we planned to meet every two weeks to discuss potential updates to the SRS doc. While we continued to meet frequently (weekly or bi-weekly), our team focussed instead on the next deliverable rather than revisiting old documents during meetings. Because of this, our SRS was not continually being updated during the project. This change occurred mainly due to time constraints, as we faced some technical issues leading up to the first demo which took priority over other tasks. In hindsight, frequently revisiting and updating the SRS certainly would have created less work for us in the long run. For future projects, though we can't predict any exact technical issues that would set us back on time, we should be able to anticipate issues arising which cause delays. We should have a plan to stay on schedule despite such issues.

For Design Verification, we planned to use the MIS checklist to ensure that requirements in the SRS are met and hazards in the Hazard Analysis are covered. Our team followed the MIS checklist when testing. We also planned to use feedback from the course instructor, teaching assistants, classmates, and our project supervisor. There were not many changes made in this section of the plan, except that our team decided to focus primarily on feedback from Dr. Zurob, as he is the end-user of the application.

For Implementation Verification, we planned to used GitHub issues and pull requests to maintain our code base. Any pull request made to the main branch requires at least two other team members to review and approve. We did not make any major changes to our Implementation Verification plan.