Reflection Report on Measuring Microstructure Changes During Thermal Treatment

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

The following document provides a comprehensive overview of a mechatronics engineering project that involves both software development and hardware integration. The project underwent several changes and iterations throughout its development cycle, including modifications to the software requirements specification (SRS), hazard analysis, design, and design documentation based on user feedback. The document also covers the process of iterative design and the various decisions made during the design phase. In addition, it examines the economic considerations associated with the project, including development costs and potential return on investment. Finally, the document concludes with an evaluation of the project management process, highlighting both successful strategies and areas for improvement in future projects.

1 Changes in Response to Feedback

In this section, we will discuss the changes that were made to our project in response to feedback received from various sources throughout the course of the project. We received feedback from the instructor, teaching assistants, other teams, and our project supervisor who also acted as the user tester for our project. His feedback was particularly important as he was the primary source of feedback from the end-user perspective. In this section, we will highlight the changes made in three specific areas: SRS and Hazard Analysis, Design and Design Documentation, and VnV Plan and Report. Through this reflection, we aim to demonstrate our team's ability to take feedback seriously and use it to improve our project.

1.1 SRS and Hazard Analysis

In terms of changes made to SRS, we addressed the feedback given by the instructor and teaching assistants as well as feedback from students in the form of GitHub issues. Some of the changes made due to that were that we defined the terms "conductivity" and "resistivity" and we used them in a more consistent fashion throughout the project, we made improvements on the document's organization and updated the Context Diagram, fixed some formatting errors, provided clearer traceability between requirements, and revised requirements and use cases.

With respect to the Hazard Analysis document, it was generally a well-written document, yet we still followed the same feedback sources and addressed the following: We fixed some spelling and grammar issues as well as missing formatting such as a List of Tables, added more details to Critical Assumptions, and other minor fixes.

1.2 Design and Design Documentation

The system design document had a few formatting mistakes that we addressed such as converting the timeline into a LaTex Table, adding descriptions to diagrams, and replacing .png figures with .pdf versions.

The module guide had a similar issue with the .png figures, we also replaced those with .pdf versions. On another note, we needed a revision to the module design itself for consistency between different pieces of documentation as well as between documentation and implementation.

Similarly, the module interface specification document needed equivalent changes to the module design for the same reasons as mentioned in the module guide section above.

1.3 VnV Plan and Report

The VnV Plan and Report were the most revised documents overall. Briefly, here are some of the changes included: revising system tests for functional and non-functional requirements, adding unit tests, revising user experience surveys, revising the unit test plan, updating relevant documentation, and updating traceability matrices.

2 Design Iteration (LO11)

Over the course of the project, substantial changes were made to the system. To explain the different design iterations across the project, we will walk through each of the demos.

For the POC demo, we did not yet have the hardware (current source, nanovoltmeter, and multimeter) connected to the system. This was due to some technical issues we ran into with the lab computer while trying to update the OS from Windows XP to Windows 10, which caused the computer to crash completely. With a limited time frame, we decided instead to build a demo GUI which displayed fake data to simulate an experiment being run.

In the time between the POC demo and Rev 0 demo, we implemented the first iteration of real-time data streaming. We solved the issue with the lab computer, by installing Windows 10 on a new hard drive. After this, we connected the hardware devices to the system. The physical connection between the hardware and the lab computer was already set up prior to beginning the project, through the use of a PCI-GPIB card installed on the computer and GPIB cables connecting the hardware to the card. We implemented the software connection to the hardware through the use of the National Instruments NI-488.2 drivers and ".DLL" libraries. By the time of the Rev 0 demo, we had implemented control of the current source output and nanovoltmeter integration time, as well as

the display of real-time measurements on the GUI screen. However, the system did not include the graphical or tabular output of real-time measurements and did not yet include temperature measurements.

In the time between the Rev 0 demo and the Final Demo, major improvements were made to the GUI. The graph was implemented to display real-time measurements. Advanced graph features were also added, such as zooming, scrolling, and selecting which values to display. The table was created to display real-time measurements in an easy-to-read format, including filters for each value. Changes were made to the layout, such as rearranging the location of the graph, table, and hardware controls. In addition to the GUI updates, the multimeter was also connected to the system to be used as a temperature sensor. Finally, the remote access module was implemented, allowing the user to start/stop an experiment and turn the current output on/off.

3 Design Decisions (LO12)

The primary limitations of our design came from the hardware devices. While the current source and nanovoltmeter were fairly easy to work with by reading through the documentation, the multimeter used as a temperature sensor introduced limitations. The multimeter is older than the other two devices and does not support SCPI commands (the language used to program the other two devices). Because of this, we could not implement a control for the user to set the integration time of the temperature sensor; instead, this must be done manually on the device. In addition to this, the multimeter cannot make measurements as quickly as the nanovoltmeter, leading to throttling of the data output to the GUI application. We are satisfied with the overall acquisition rate (how many data points per second) of the application, as it is still an improvement from the previous implementation. However, if a newer device had been used for the temperature sensor, we likely could have achieved even faster speeds.

Another hardware limitation was introduced by the PCI-GPIB card. Installing the drivers from National Instruments posed no issue, however, we quickly discovered that the ".DLL" libraries used to implement the connection to the card in software are not compatible with a UWP application. Unfortunately, at this time in the project, we had been developing the GUI - separately from the hardware - as a UWP application. We originally chose UWP because we wanted to create a multi-platform application. But due to this incompatibility, we were forced to migrate our application from UWP to WPF, an older Windows framework for GUIs. This limited our application to being installed on Windows machines only and also prevented us from using some of the newer GUI controls available in UWP, such as the toggle switch.

Some assumptions were made by the team regarding the difficulty in connecting the GUI to real-time data input. Developing a static GUI is an easy task, but

once there is a real-time component, the coding becomes more complex. None of the team members had previous experience developing a GUI using C and WPF, so the design process consisted of much research and trial and error. The function for reading in data from the hardware must run in a continuous loop but also be interruptable (in our case, by the "Stop Capture" button click). To implement this in C, the async operator must be used to create a "worker thread" separate from the main UI thread. The real-time data streaming runs on a worker thread while the UI thread control button clicks and other user interactions such as entering text.

4 Economic Considerations (LO23)

There was never any plan for marketing or selling our application since it was designed specifically to be used by the project supervisor, Dr. Zurob, or any of his research assistants who would be performing thermal treatment experiments. In this way, the project was somewhat easier as we only had to focus on satisfying the requirements of a single client.

That being said, the application does not necessarily need to be limited to only one client. Any researcher using the 4-point probe method to calculate the resistivity of a thermally treated sample material could find value in a similar application. If we were to modify the scope of the project to include this consideration, some major changes would need to be implemented. The application was designed to connect to the specific hardware devices which were provided at the beginning of the project, namely the Keithley 6220 Current Source, Keithley 2182A Nanovoltmeter, and HP 3478A Multimeter. Each device (excluding the multimeter) is designed with its own unique set of SCPI commands, which the software uses to communicate with the hardware. This makes it difficult to design the application to be modular or "plug-and-play", such that the user could connect any GPIB-compatible device to the system. One way to do this may be to create a "database" internal to the system which contains a list of SCPI commands for a number of hardware devices. There would need to be a function for checking which devices are connected to the system (device ID can be queried with an SCPI command), and then fetching the correct SCPI commands as needed from the database to control the device. This change would allow any user with an appropriate hardware setup for the experiment to use the application. Even so, this may be too niche of an application for it to be viable for selling.

5 Reflection on Project Management (LO24)

5.1 How Does Your Project Management Compare to Your Development Plan

For the most part, we followed our Development Plan throughout the course of the project. We held weekly meetings on Teams to either discuss the progress of the current deliverable or plan for the next deliverable. However, we did not always meet with Dr. Zurob on a bi-weekly basis but instead only scheduled meetings with him when it was necessary to receive feedback on a certain part of the project or to ask for more detailed information on requirements. The team member roles set out in the Development Plan were accurate to each team member's role during the project. Regarding the workflow plan, we had planned to use GitHub Milestones to outline high-level goals but ultimately never used this. We did use GitHub Issues to track sections of Documentation that had to be completed. Each team member would create an issue based on what section they were assigned, label it as "documentation", and create a separate branch to work on. After completing the section, the team member would create a pull request to be reviewed by another team member. After review, the branch was merged and the associated issue was closed.

Some technology mentioned in the Development Plan was ultimately not used in the project. We stated that we would use both Teams and Discord for team communication, but only ended up using Teams as this was the most simple method. In the Technology section, we stated that we would use JavaScript, HTML, and CSS in the Electron framework for developing the GUI application. However, we decided instead to use C in the WPF framework to create a native Windows application.

5.2 What Went Well?

Team meetings and communication went well throughout the project. We stuck to our weekly meeting plan and divided work among the group effectively. Using GitHub Issues to track sections of documentation was also effective for managing workflow. With regards to technology, using Teams was a good choice for meetings and written communication between the group.

5.3 What Went Wrong?

We did not meet with Dr. Zurob as frequently as initially planned. More frequent meetings would have kept our project more closely aligned with Dr. Zurob's expectations and requirements, mitigating the need to retroactively make changes to documentation or add features to the application last minute.

Regarding the technology, the change from developing in JavaScript and Electron to C and WPF presented some issues as none of the team members had prior experience with the latter.

5.4 What Would you Do Differently Next Time?

For our next project, one major change to make would be to start developing the application as early as possible. Our thought process throughout this project was to follow the timeline of the deliverables and assign work accordingly. However, this led to us only beginning to the develop application a few weeks before the POC demo. In hindsight, this was a mistake as it did not leave us with enough time to work through setbacks. Ultimately, this caused our team to have a pretty poor Rev 0 Demo, as several key features of the application were not implemented yet.