UVSim Quality Control

Riley Giles, Emily Starks, Caiden Kehrer, Bradley Van Fleet

We at IED believe that our product is up to the qualifications and standards of UVU due to the great amount of consideration put into testing and planning throughout the construction of the project.

Firstly, our product has multiple unit tests prepared. These tests are run every night on the server the product is built on, to ensure that every portion of the product is polished and working functionally as a whole.

The first set of tests focus on the ALU. When run, these tests check to make sure that the ALU is properly doing all the basic required functions such as subtraction and multiplication. To be thorough, these tests work through possible scenarios in each required operation, such as if someone were to multiply all negative numbers or add a mixture of negative and positive numbers together. In the case of the ALU I feel that enough tests have been prepared to make certain that the ALU runs properly.

There are also integration tests that check to make sure that the IOBus, processor, and memory are all functioning. Each and every single one of these portions of the project are each tested on their own. Running these tests will allow us to see if there are any issues in any of those individual areas. Further tests are then done to makes sure that each of these interacts well with everything else. For example, the test to make sure the processor and the IOBus can function together as well as separately.

These tests will allow us to find finding any issues when these parts of the program integrate with one another. It will also allow us to find bugs in individual portions of the program, should they exist, so that we may eradicate them long before we deliver the final version.

Each of these tests are also unique. We do this in order to make sure we aren’t wasting testing time with redundant code, and to make sure we go through as many different possibilities as we can in order to feel confident when we say that the code is well-tested. Due to this, I believe that the tests created run through a multitude of possible issues and scenarios, and thus are sound as they stand.

IED is focused on quality and understands that UVU had some requirements set forth. The product follows the requirements UVU has declared, and the company has taken some extra steps not provided by UVU in order to ensure the product we deliver is up to the expectations of UVU’s fine establishment, and the standards of our own company.

First of all, one of the standards of UVU was in reference to cohesion and coupling. We have done everything we can in order to ensure that this project was designed with low coupling and high cohesion in mind, which is the ideal.

Low coupling was achieved in part by making sure each individual part of the program could run smoothly on their own. We wanted our program to run smoothly, even if one area of the code has a bug, it should affect as little of the rest of the code as possible.

We made sure that individual parts and functions of the program had error handling and would allow you to reenter the information or restart the program with ease, without affecting parts of the memory, and it should also allow the rest of the program to run smoothly, even if an error occurs in one sector of the project.

High cohesion was also implemented to the best of our ability. The program, the objects, and the functions, are all classed together and organized well. It is logically assembled so that the functions and objects that work together make as much sense as possible, thus making sure it all runs smoothly together.

On top of that, the abstraction should be very clear. Each function is named and called in, or close to, sequential order. Every module of the program is clearly connected in some way to the others, without relying too heavily on the code implemented in those portions.

We also did our best to make sure we accomplished the most that we could with the fewest modules that were necessary. As I’ve stated before, we do not want redundant code, nor do we want unnecessary code. This has been trimmed out and streamlined to the best of our knowledge, thus we feel quite comfortable in saying that our project has achieved a status of clarity, as well as high cohesion and low coupling.

The code should not be overly complex in it’s execution, though we used AND, XOR and OR gates in order to achieve the arithmetic sequences, we made sure that we used the simplest methods possible for each arithmetic function. The point of the project is to allow students to understand little machines and how they operate, and we wanted to make sure it would do that even on the most basic level.

Each programmer also as been instructed to watch one another, our programmers will look over and check each other’s work while running tests of their own. We wish to distill a feeling of quality on the micro and macro level this way.

Programmers also made very clear and concise comments throughout the program, this should allow anyone who takes a look into the program to understand why certain decisions were made when it came to our code and would allow others unfamiliar with our coding style to look over our code, understand it, and adjust as necessary.

Speaking of others looking at our program, we made sure that the GUI interface will interact well with the students. Most of the code is hidden from them, so that they cannot make changes on the coded level, however, the students who use this program should be able to access the project memory and add or take away information there to their liking. This should allow them to use the project and learn the subject at hand while also avoided the likelihood of learning minds tampering with the code and causing unforeseen errors.

We also made our program with the possibility of upgrades and patches in mind. In fact, while working on this project, UVU has added a few requirements as we were creating the project, and we had no problem implementing those requests because upgrading was kept in mind from the very start.

The finished product should reflect that just as much as the previous versions. It is important to us that, should any awful bugs occur, or ideas to improve the program come to light, it can easily be integrated into our code. This way, should UVU think of any other improvements it wishes to make to our device, it should be relatively easy, whether they use our company or another in order to make the design changes. We know that no project is ever perfect, but we have done what we could to ensure that the process of improvement never ceases with our project.

In conclusion, the project has been tested to a great degree, in every variation that we could think of. While that doesn’t guarantee the absence of errors, it does guarantee we have done everything in our power to make sure we deliver excellent, and working code. It should be clear coded, high cohesion and low coupling. It was well designed with upgrades and students in mind. Overall, we are satisfied and certain that we have created a quality product that UVU can easily use to teach the minds of tomorrow, and we feel our code is sound quality work.