Assignment Review Form

COMPSYS723-Assignment 2

This form must be completed to evaluate the overall performance of the assignment. Fill **one form for each group**.

Fill in the following information:

Team/Group Number : **13**

Name of the Student 1: **Connor Dunn**

Name of the Student 2: **Lachlan Chan**

Please fill in the following table to indicate the number of hours spent by each student.

|  |  |  |
| --- | --- | --- |
| **Task** | **Student 1** | **Student 2** |
| Planning | **2** | **2** |
| Understanding the assignment | **1** | **1** |
| Understanding Esterel | **7** | **6** |
| Programming | **4** | **5** |
| Debugging | **2** | **3** |
| Documentation and Reporting | **6** | **4** |
| Cropping Images | **2** | **0** |
| **Total number of hours** | **24** | **21** |

**Any further comments on the assignment:**

**How the cruise controller assignment can be improved?**

* More help on using the VM
* An actual full assignment folder – the makefile was only accessible through the lab folder
* Quite often, the VM bugs out or doesn’t you do somethings like accessing chrome wasn't possible after a while, pushing commits to github
* The makefile would sometimes bug out as well and we would have to redownload the lab1 folder and retry to make the project.

More comprehensive test vectors and test guidance, very little was talked about regarding testing the esterel code, especially the “vector.in” and “vector.out” files. I think if there were more test ticks which attempted to exploit edge cases, we could have created a truly robust system. It was also just quite difficult to navigate the tests, was there a program which inputs .in vectors, tests the code, and outputs an .out vector? If so, I did not know about it. If not, that’s fine, it was just tedious...

**What do you think of the Esterel language?**

It's okay but it's kind of difficult to get the semantics right at first, especially when trying to compile the code, but you get used to it after a while or once you know what you can or cannot do.

I think it’s cool, i like being able to design real-time systems with true determinism. I also like how the language properly describes FSMs. A new language is difficult to learn in a short period of time but with a strong background in designing and understanding FSMs it came quite naturally. Having to use the VM sucked though.

**What do you think of the formal verification?**

It is a lot more manual and tedious to validate since most of it must be done very slowly and by hand, but very interesting and we now understand why it is such a slow and painstaking process.

It is quite strict; an implementation we were playing with failed the formal verification due to a causality error. However, such an error could not exist as the scenario depended on several signals not being present which formed a loop as the program had no state to settle in, the signals, however, couldn’t all not be present. Also didn’t really know how to use it, was “make code.xev” enough? Or did we have to do more to truly verify it?