Dear Utah Valley University:

We have decided to keep most of our code from our previous program. Much of it is being reformatted, but our old code has many code blocks and functions that are easily reusable in our new code. There is no need to start from scratch if our old code can be reused. The requirements of the project have evolved, and so the base function of our system must evolve along with it.

Our old system was simple with few moving parts. We had two classes, the CPU and the GUI. The CPU handled most of the computation, and then the GUI displayed it all to the user. This system is far too simple to carry it over in its current form with the evolved format required in the new milestone.

We have decided to split up our code into several different classes and add new code to go along with it. Our current prototype will not be trashed, but it is definitely getting reused in our new prototype. Things such as the arithmetic within the switch case, and our handling of the operator and operand of the passed instructions won't be changed.

A good portion of how we handle the GUI won't be changing either. However, the strings and format of our GUI are changing to something more user friendly, and we're refactoring a lot of our code to conform to the MVC and Facade systems. We're pulling our strings out of our GUI (which is changing to the "view") class, and we're putting them into their own wrapper class to be pulled from. We're also separating the memory and making it it's own class so we can have more control over its manipulation. We're also separating the CPU itself from the model, which will run all of the functions of the code through the memory.

We feel this is the best way of handling things. It allows us to reuse code without "trashing" our old prototype, but also allows us the flexibility we'll need in order to meet the requirements of our client. Reusing code wherever possible is a good practice because it saves time and effort. If we can deliver new prototypes regularly to the client, we'll be able to complete the system at a much faster rate.

Completely trashing things without reusing code is a bad methodology to follow because it wastes time. If code is perfectly workable, it doesn't make any sense to get rid of it if you're just going to end up coding the same thing a second time.

I would be shocked if any other group entirely trashed their old prototype to start from ground zero with this milestone, and I'm glad my group is doing it this way. I'm not interested in doubling our group's work load. We're all very busy college students, so any time we can meet the requirements of an assignment with less work, the better off we'll be.

In conclusion, reusing our old code really is the best way to go about this. No matter what, as requirements change, our code will have to change with it, which is what we've done and will do going forward. But, if code can be reused it ought to be, and we have done so.

-After Innovative Education (IED)

Daniel Espinel *Erik Manley* *Santiago Rameriz* *Anthony Peterson*