**Design Justification Group Assignment 4**

With functionality, the new updates administered to the vending machines will include an ability to pay through debit card, a graphical user interface, and a configuration panel for technicians. It will also include a locking and unlocking mechanism, which are designed to enhance safety protocols for technicians.

Our updates are facilitated through a number of classes that communicate amongst one another, thereby communicating with the respective logic class. The communication is carried between parts of the software, which are then administered, and will not rely heavily on the hardware. The lessened communicational burden between software and hardware allows for changeability of the program, while providing comprehensibility. When taking into account the advise that was provided by Mr. Client on keeping the software at optimal level of changeability without the burden of the aforementioned hardware, our design has been formulated according to efficiency. Hence, throughout the organization of the source code, each of the unique classes is observed to be communicating with one another.

This organization of the source is formulated in manner that would be easily understood by an individual with a Computer Science background. Organized in respective groups of major tasks, the source code has classes that all fall under the graphic interface, the configuration panel and so forth. Although these major tasks are then frequent to communicate with one another, its main task will remain acting as a closed system. Thus, making this process independent from the hardware, or the rest of the code for that matter. This allows for accessibility of new implementations in further updates. This manner of programing through closed systems would prove convenient, as it would not interfere with any recent changes to the software. When communicating with one another, via vending machine’s software and hardware, the two act akin to the likes of a TV communicating with a DVD. If in a case that an aspect of the DVD is flawed, the TV would not experience negative feedbacks of major bugs, but instead maintains its integrity by remaining as independent entities - despite the awareness of the other system’s presence.

In addressing changeability, the most realistic feature allowing this code to evolve would be the ways in which the user’s credit is incremented. Currently we are keeping track of the user credit in one variable, where the credit is incremented when a valid coin is inserted, or when a user pays with debit. With this method, which one variable is keeping track of the credit, we can add features that allows a user to partially pay coins, and pay the remainder with any other form of payment including debit. This method keeps the flexibility and evolve-ability of the code very high, also making it easy to implement users paying with various forms of payment, including Bitcoin. Thus this system can efficiently support any new currency that may be implemented at a later date; yet a majority of the original code will remain as it is. Once again the idea of closed systems applies here as the calculation of the user’s credit is done through a listener class, and will not take into account weather the user is entering in a coin or paying with debit, as long as there is sufficient funds, the pop will be delivered to the user.

If a feature allows accessibility to other forms of payments, the code would remain nearly stagnant, or within its intended originality. This will forward ease in building on top of pre-existing software. The structure present in the software is designed in respect to support alternative hardware for further future uses. This can be observed through the style utilized to code the methods, such as: event logs, valid coin inserts, buttons pressed, and so forth. In methods that are used to determine what buttons are pressed, or how much a pop-can costs, it is simplified using an array list that benefits hardware changes. This array list has proven to be effective when for example the number of buttons transforms within the hardware, or when the cost of the pop can is being updated. Thus, this will ensure that any buttons added to the vending machine hardware will fully be represented in the software. Finally, the update of the cost for a pop-can will be administered through the user-friendly technician panel.

As far as scalability goes, when on the topic of vending machines, the software we have implemented is unquestionably scalable when considering it can support new hardware’s with basic adjustments.

The structure of the vending machine software as a whole is set up as a group of interconnected closed systems allowing for high levels of changeability, and this is evident in all the examples we have discussed. Not only have we confirmed the validity of the immense levels of changeability but also the ease in understanding the software. Along with the comments to support your understanding, the code itself will not require a convoluted amount tracing throughout the system to follow along. Rather, you will only have to look at the each closed system, and then learn the ways in which it communicates.