System Architecture

Discussion

Architectural Decisions Taken:

The system was split into three subsystems for our implementation: the Business Layer, the Data Layer and the User Interface Layer.

We implemented a range of Model View Controller (MVC) design patterns in order to help our system be more flexible and easier to maintain. MVC separates business logic from the user interface. In our implementation we separated the user interface layer, the data layer and the business layer. By using MVC it allowed for code reuse within the system and it reduced coupling between the layers.

We also included a range of design patterns including the factory method which was implemented to help create proper subclasses of products so the system can easily determine which product was chosen by the customer at runtime.

UML Workbench

We chose Gliffy and Creately as our main UML workbenches due to the fact that they produce clear, easy to understand diagrams and they were also free to use. Each of the workbenches also provided some templates which aided us in creating our own diagrams for this implementation. We found it easier to divide our work between 2 different workbenches in order to avail of the strengths that each workbench possessed.

For example we found Creately much easier to use for the large analysis class diagram and state charts due to the fact that larger diagrams could be created easier and were much easier to read and navigate. We then used Gliffy for creating other diagrams, including the state diagrams and the sequence diagram.

Implementation language

We chose the Java programming language for our implementation. We chose Java as it is one of the most portable programming languages available, which means that our system is available to the many architectures which support the Java Virtual Machine (JVM). Java has a JVM for many of the modern architectures which means that our system will be available to many more users than if we used the C++ programming language.

Our program also does not rely on speed which is another reason why we did not use C++. The added security, through Java’s garbage collector and the fact that all members of the team were comfortable programming in Java is what helped us make our decision.