**CS 396**

**Shariq Jamil, Austin Sanders**

**Assignment 3**

**Modeling Using Classes and State Machines**

**Due: Monday, May 5 2014**

**Modeling and Implementation Approach**

           Our approach for this project was iterative for the most part.  We began with a small skeleton model and progressively added functionality as we discovered what was missing.  We began with implementing the core state machines that the program required (i.e. plane, runway, personnel) and progressively added layers of functionality (i.e. emergency). We tested each function we implemented using a driver class to ensure reliability. Eventually we worked our way towards interacting state machines.  During this time, we referenced Umple documentation heavily to utilize any built-in functions. We implemented complex state machines, but they did not transfer over correctly to the Java code, so we commented them out.

**Self-Criticism**

We started working on the project late and were not able to implement all the functionality that we had planned on adding to the program. This was due to the fact that we were busy with projects from other courses. Having started late and dealing with Umple’s issues (deleting parts of code, incorrectly rendered diagrams, and features such as nested state machines not working) led to us getting frustrated and having little patience to want to work around issues. These errors in Umple also drove us away from thinking “out of the box” and experimenting with different features we wanted to add. Mainly, we focused on dissecting the project requirements and implemented everything the project required at minimum.

**Suggestions for Enhancement**

Due to the relatively small size of this project compared to that of a model of an actual airport, there are numerous enhancements that could be made.  One thing in particular that could be added to enhance our system is a random event system.  Currently, the creation of planes, runway malfunctions, emergencies, etc. must be triggered manually (as can be seen in the driver class).  By adding a random event system, the software could be more realistic and “autonomous” in a sense.  However, this draws away from the modeling paradigm and would require more imperative coding than anything else.  Another feature that could be implemented to enhance the system is introducing functionality for the customer class.  This would include booking flights, boarding planes, and checking in luggage.  We originally included a “flight” class as an intermediary between customers and planes, but we had to choose between complicating our system and straying away from the goal of the project or simply having no functionality for a flight at all.  Due to time constraints, we chose this to be an enhancement rather than an implemented feature.  Also, we chose to implement our model as a single airport instead of an interacting system of airports.  We could easily expand this functionality to include multiple airports by creating multiple air traffic control stations and adding an airport class.