**Condor Army – Team Analysis**

Team Members: Kevin Zarzana, Cullen Sturdivant, Sharon Trujillo Hernandez, Dustin Rowe and Michael McBride

**Software Requirements Specification**

Our team didn’t meet with many overwhelming problems during the SRS phase. Though, like any other phase, it did take significant communication in order to get on the same page. This is when our team made important decisions, such as what language we would be using; who would be responsible for orchestrating certain tasks, so as to ensure we could funnel intentions and work through one point to prevent portions of a task being done by two different members simultaneously; the overall idea of what our graphical user interface would look like. One member of the group made the suggestion that we use C# and Unity for the GUI and back end code, respectively, and while many of us were not familiar with the environment, we all agreed to the challenge. It turned out to be a significant opportunity to learn.

**Software Design Description**

Any issues that the team ran into seemed to stem from this portion of the waterfall method. We found that writing a software description independent from the code proved to be inefficient, at best.

While writing class diagrams, and attempting to organize a plan for the code seems straight forward on the surface, doing so without testing and implementing code proves to be rather difficult, as often there is some amount of trial and error when creating software (especially when many members of the group are new to the development environment and langrage). Because of the specific ways in which the backend code must work in conjunction with Unity, the design can be adhered to without the different pieces actually working together.

These problems are only amplified when the team is spread out. Add the fact that students are often taking many different courses, with different course loads and assignments, and it becomes difficult to find the same block of time on everyone’s schedule. However, many of these blocks of time and meeting sessions are necessary when attempting to implement certain design descriptions from the SDD. As a results of these difficulties, the backend code was written for a significant portion of the project, but it did not work with Unity. On a software development team where members are in closer proximity with each other, and can bounce ideas off of one another to ensure that the moving pieces will work together, the SDD implementation may prove to be slightly more practical.

While the issues with the SDD did not require us to go back to previous phases of the project in order to adjust requirements or design patterns, it did prove to be costly toward the end of the project, as is discussed in the coding section of this report below.

**Coding**

This step in the Waterfall method demands constant communication among developers. Due to the GUI environment requiring the back end code to be integrated seamlessly, the team members who were more familiar with the GUI implementation were required to shoulder a significantly larger burden during the programming portion. Should a team be filled with members that are very well versed in the technology being used, this problem may not be as amplified as it was in this project.

The code may also have benefitted from a more abstracted approach, such as can be implemented in the sprints in the agile method. Working and combining smaller portions of the code may have made some of the problems we ran in to apparent at a much earlier phase of the project. Because the waterfall method requires the coding to happen at a much later phase, problems with implementation can leave the team scrambling to complete the project in the final minutes.

**Overview**

The project, overall, provided some very effective learning opportunities for the team. While the difficulties of Waterfall were apparent, this is a valuable lesson to learn about software engineering techniques. Additionally, the decisions that the team made early on allowed for growth as well. The importance of communication cannot be overstated, and the group learned just how important this skill is while working on any team, especially a team that requires so many to intricate pieces to fit together and work seamlessly. In the end, the project was a very helpful tool that demonstrated important parts of software engineering, and the Waterfall method.