**Team Analysis**

for

Ferret Army Chess

Version 1.0 approved

Prepared by A. Maxwell

Ferret Army

February 2018

In Project 1 for CS 3398.264 Ferret Army (FA) undertook the recreation of the classic game of chess using the software design method known as *waterfall process.* This process included ridged guidelines, a methodical development process, and a plethora of documentation to generate while creating FA’s version of chess. While the *waterfall process* itself is known to be beneficial for large or critical system software these benefits did not translated to our project 1 software called *Ferret Army Chess* (FAC).

As the sole preparer of this document I could spend several pages disparaging the *waterfall process*; however, in order to gain a clearer picture from the entire team I created a survey asking every member of the team to answer the following questions:

1. *What was your favorite part of the Waterfall process?*

While it took some coaxing, the general sentiment is that while it’s hard to find a favorite part of the *waterfall process* we each enjoyed planning, strategizing, and agonizing over every detail of the chess logic. You really don’t gain appreciation for software you have access to until you have to program it yourself. The group also collectively mentioned enjoying having to collect user requirements, translate these into a design, and then implement the design we came up with. I know for some of us it was the first time getting to complete each process entirely from scratch without strict guidelines top-to-bottom given by a professor.

1. *What was your least favorite part of the Waterfall process?*

This question garnered the most responses, and the general sentiment from the group was that the entire *waterfall process* was our least favorite part of the *waterfall process*. From too much documentation to the general feeling of constantly being held back by the process. Not being able to code until documentation was completed left several of our group members feeling like a horse chomping at an imaginary bit that’s never released. Often times this meant our programmers had to either conform to what the documentation said regardless of its usability, or those writing documentation had to continually go back and conform the documentation to what code was being generated.

1. *What was your biggest challenge during project 1?*

Again, this question garnered a large amount of responses. One of the under-riding sentiments from our group was that, in addition to the constraints of the *waterfall process,* the inability to meet daily (or a couple times a week) in person led to confusion over the design process. This confusion would ultimately cause the programming process to come to a grinding halt until we’d debated the design of a component. These debates would only transpire during in-person meetings, so what ended up happening quite often is one person would dive in and code a bunch only to have to refactor it later due to it not meshing with preexisting code written by another group member.

1. *What would you have done differently?*

The overarching response to this question was everyone’s desire to adopt an agile model with an appointed scrum master each week to keep us on task and pointed towards the same goal. Often times during our weekly meetings we would debate trivial things, or we would argue over the design of a single component for the entire meeting without coming to a resolution that was implementable. This led to frustration by all parties involved and caused development to stall quite frequently.

1. *Would You Use Waterfall Again?*

This question was asked on a rating scale of *Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree.* With the responses to the aforementioned questions it’s no surprise that most in the group stated they would *Disagree* or *Strongly Disagree* with using *waterfall* again. Although there were some highlights project 1 with the use of *waterfall* left a rather foul taste on the teams proverbial “coding taste buds”.

1. *Does Waterfall prompt teamwork?*

Since one of the seeming cornerstones of the agile method is tightknit team work through intensive planning, daily meetings, and a retrospective after the completion of a project I thought it prudent to see how the *waterfall method* measured up in our groups eyes. This question too was asked on a rating scale of *Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree.* Surprisingly, even with weekly meetings and a decent amount of discussion via team chat, the responses to the level of team work promoted in the *waterfall method* was rather low with most responding neutrally.

Overall while our group learned a lot through project 1 and the *waterfall* process I know we are all echoing “On with project 2, on with agile!” It will be interesting to see in an agile method what changes are made to facilitate more teamwork, better use of our meeting times, and how an agile development style will lend to better code production.