Software Development Project

LOC Counter

Report By: Gabriel Stroe

Course: Software Engineering

Project Dates: September 26th 2016

Date Submitted: September 5th 2016

# Introduction:

The following is an analysis on the completion of projects 1 and 2. This analysis will look at each stage of development and look at why and how those processes were executed. For projects 1 and 2 a “V Model” development process was used.

# Planning Phase:

For Project 1, the planning phase mostly consisted of estimating the number of lines of code and the time that it would take me to program this assignment. In truth, I had not considered how long the debugging or the write up analysis of this project would take as my coding methods before have simply been to debug as I code. The LOC estimate was a simple guess as I had no real framework to base it on. I know that programs of this complexity usually have line counts in the 100s with comments so estimate of 100 or a little less seemed appropriate. As for the time analysis I had estimated that a program of this nature would take me about 2 hours to program and debug. Due to that, I had planned to work on it the night before and the day of. This was a very big mistake as I found the coding and write-up would take me a lot longer then estimated.

Project 2, I was able to handle these estimates a little bit better. Estimating the LOC was easier due to already having a base program to work with. I was able to use that base to estimate the new feature of the code a lot better. In terms of the time estimates, I estimated that the code would be about 3 hours to code and another hour to test the code. I decided that it was better to give a little bit more time then I had initially expected on the coding and debugging since I was defiantly short last time. With that I had decided that the weekend before the project is due should be enough time to finish it.

In the end even after allotting a larger amount of time to the second project, it seemed as that was not enough. With the second project I had a few larger bug issues that took much longer to debug than expected. It would seem that although my coding time seemed to be fairly accurate, the amount of time spent finding and fixing issues is significantly larger than I was expecting. In the future projects more time needs to be allotted to the testing phase of the projects. In addition, the write ups take a bit more time than expected due to the nature of discussing each phases as well as building a professional design chart.

# Requirements & Analysis:

For both projects I had felt that this phase work very well for me. I was able to create quick summaries in each project that described and listed out all of the requirements for this project. In the future when a more feature heavy project is given a spreadsheet with all of the features listed may be more appropriate, but since the feature count for these projects were fairly small; a short paragraph seems to be appropriate.

# Design Specification.

So this is where I had some larger faults in terms of the process order. For project one I had just made a paragraph describing the logic for the program. I should have created a diagram showing the logic at each stage of the function. This cost me a lot of time for coding as some of the smaller details needed to be designed still.

In project 2, I had a better design plan as I had outlined it on paper before coding the new function. My only issue I had found is that converting that paper design to digital for the write-up took a lot logger then estimated. In the end though, I had felt that this design strategy saved me time in the coding phase.

# Implementation:

For the implementation of both programs, I had simply decided to code the entire program at once before testing. It seems that due to the lack of very separable features in these projects this was the best method to take. If there had been more features that did not rely on other features I would have coded them independently and tested them independently. For example had these projects had a second optional parameter that modified each line of code so anything after a “;” got pushed to a line of its own, then that feature would be something I developed after the main program as it is separate and does not actually interact with the counting feature.

# Verification/Validation:

For both of these projects, I had kept the feature verification and overall validation as the same process. The reason for this has is due to the number of features both projects had. Given that the feature count is so small, to separate the phases seemed rather redundant. The input and printing features are very secondary and simple. They are needed to validate the main counting features, so the unit test case and the integration tests came out to be one of the same. For larger programs with greater features counts, these two phases will be separated out, as then the difference between unit tests and integration tests will be much more obvious.

That being said project 2 did have two separate unit cases that had to be both individually tested and integration tested. In my code one happens directly after another, so the integration test actually happen at the same time as the unit test. Looking back now this should have been separated out into two separate tests. It did however lead to me finding my first bug with the program which was an issue with the two files sharing the same buffer.

For each test an expected result was calculated. If the print statements should the correct lines numbers then the test was marked as passed. If even the smallest difference was shown, then it was flagged as a failed test. When all of the test cases passed then the program as a whole was in the passing state.

The test cases themselves where different types of code that contained all of the possible situations for lines of code. They mostly involved the C code given as an example, the base java file, and my final java file for this project. In the second project, using those last two proved to be very useful as it showed some logical errors and situations that I had not accounted for.

# Retrospective Analysis:

For the retrospect Analysis, I had looked at my estimates and compared them with the results, In the case of these two projects, this meant looking at the lines of code and the time spent on each phase. Overall I had learned that I don’t spend enough time in the debug phase of the project as that seemed to be the phase that was grossly out of proportion of the estimates.

Also I have learned that writing this report or analysis as I finish each phase would save me a lot more time. I normally go through each phase and just take small note sin my code then compile the notes into the final document. This seems to be inefficient as it would be better to write each part as you finish a phase. This is what I plan on doing in the next project.