Software Development Project

LOC Counter

Report By: Gabriel Stroe

Course: Software Engineering

Project Dates: August 29th 2016

Date Submitted: September 12th 2016

[**General Comments**:

* Your report should stand on its own; delete all of ***MY*** blue comments in the square brackets (INCLUDING the brackets) and replace them with ***your own*** content. Each section should include a ***brief*** summary explanation (one or two sentences) as to the purpose and/or content of the section.
* One important goal of this class is to help you to become more effective in communicating complex and abstract ideas to others (both technical and non-technical).
* As you work through your project and record the associated documentation, try to keep in mind the thought “What if someone else were assigned to implement or modify my software?” Consider the following realistic scenarios:
  + Immediately after completing your project, you are promoted or assigned to a new project. Your software (THIS project) requires a change (enhancement, fix, new customer, etc.) “RIGHT AWAY!” There is no margin for error, and in your new role or assignment, you are too busy to make the change. How much difficulty would a new team member have (given your project documentation - requirements, design, and test - and source code) modifying your software (requirements, design, code) ***with no mistakes***? Could they repeat your testing to make sure that they did not break anything that used to work correctly? (Also keep in mind that this lucky person modifying your software might be **YOU** six months or two years from now... Did you leave yourself enough clues to remember what you were thinking way back when?)
  + You are managing a project, and you have the domain expertise to completely understand the customer’s requirements and expectations. However, you will be overseeing a team of developers as ***they*** develop the software. Will the development team understand your statement of requirements ***without misinterpretation***? Will an independent validation team be able to use your statement of requirements to develop test cases to validate the final product against the customer’s expectations?
  + You are a lead engineer on a team of developers. You understand the requirements, and you have a concept of how to decompose the software into “components” (whether functions or objects). You will need to distribute the development work among the team members to complete the project on schedule. How will you communicate High-Level Design (your concept of the program structure) to you team so that they can independently design, implement, and test their components? The High-Level Design includes the overall decomposition into components, the interfaces and interactions between the components, call/return hierarchy and interface structure (function prototype).

**General Suggestions**:

* In general, write your project report as if you were presenting it to somebody who has knowledge of the subject matter but was not involved with the project. Include diagrams, tables, etc. throughout your report as needed for the reader to understand your report.
* Label ALL figures, diagrams, tables, etc. throughout the report.
* The general flow of the report should be something like:
* Summarize project Purpose, Mission, and Vision
* Time and Size Estimates & Plan
* Requirements Analysis
* Design
* Implementation
* Testing
* Retrospective Analysis
* Appendices

THIS IS A GENERAL LAYOUT SUGGESTION, **NOT** A REQUIRED LAYOUT, FORMAT, ETC.

Also understand that the various sections of the template represent a GENERAL project/development workflow. Include the information that is specific to **YOUR** project/development workflow.]

# Introduction:

The goal of this project is to create a command line based program that takes a file as a parameter and counts the number of programing lines that exists inside of it, excluding any C++ style comments.

This program should help developers count their lines of code to better understand the current scale of their program and event use past programs to understand what kind of scale you could be looking at in the next project.

In the future this project would be adaptable enough to include in as an extension to other more complicated programs. Although for now, the simplicity of this as a command line will satisfy the requirements given to us.

# Estimates/Plans:

The overall estimations for the project is that it will take a total time of 4.5 hours and about 67 lines of code. This is based off a few educated guesses based on my previous projects in other courses. The exact breakdown of these estimates can be found in the Project Log Excel Workbook on the Project Estimates Worksheet.

The plan for this project is to spend is that I spend 9/9/2016 working on the design features and 9/11/2016 working on the code and testing for the project. Since this project is small then one day working on the coding and the testing should be ok.

# Requirements Analysis:

The requirements for this project are that the program is able to take a sample code and count the number of lines in that code. For this project, it will use a command line interface.

[Include or summarize (along with reference to appropriate appendices) any more detailed Requirements analysis as to *WHAT* the final product will do.

Consider Use Cases or tables/lists of requirements. Remember that “good” requirements are specific, unambiguous, and observable/verifiable (among other things). Well-defined requirements should almost automatically create their own test cases; if you are not sure how to test a requirement as stated, consider revising the statement of the requirement.]

# Design:

[Include or summarize (along with diagrams or references to the appendices containing them) any more detailed Design analysis as to *HOW* the various parts of the program will be integrated and will interface (“Architecture” or “High-Level Design”) and *HOW* the individual components of the program will work (“Detail Design”).

Include a Context Diagram (either in the Requirements or Design section) showing ALL system/software Inputs and Outputs, along with an indication of the source of the input and destination of the output (these will be devices, “Actors”, etc.)

A “High-Level” or “Architectural” design of the overall system/software shows all major “components” of the system/software (whether objects or functional blocks) and their relationships and/or interfaces (how they share information, call and control hierarchy (who calls who), etc. As a starting point, consider any “components” that you assumed during “decomposition” in the planning and estimating process; revise and refine these initial decomposition assumptions as necessary.

For each “component”, a “Low-Level” or “Detail” design should define all Inputs and Outputs of the component (like a Context Diagram for the component), all sub-components and their interactions and interfaces, and implementation details (algorithm, logic, states and transitions, equations, flow chart, structured language, etc.) on how the component transforms its inputs into its outputs and/or provides its intended behavior.]

# Implementation:

[Summarize details related to the implementation methodology (such as coding standards, code review, etc.)  
  
**DO NOT PASTE SOURCE CODE LISTINGS HERE**.]

# Testing:

[Include or summarize (along with reference to appropriate appendices) any more detailed Verification/Validation plans and outcomes. Consider here how will you demonstrate (to yourself and to your customer) that the system/software (individual components and the overall integrated system) functions as expected.

Document your test cases (what you tried; any special set-up and the inputs that you provided), expected results (how you expected the system/software to respond/behave) and observed results. For each test case, did the system/software respond/behave as expected (“PASS”) or not (“FAIL”)?

The “Test Log Template.xlsx” is provided to help with the test planning and logging.

Self-check: Could your testing be performed by or repeated and confirmed by an independent tester based on the information that you have provided?]

# Conclusion/Project Retrospective Analysis:

[This section contains closing remarks summarizing the key outcomes and observations from the project. Consider the data in the both the Time Log and Defect Log in this analysis. Include a summary of the estimated vs. actual Time and Size. Include a summary of “Things Gone Right” and “Things Gone Wrong.” Include a brief discussion of any “Lessons Learned” that can be applied in future projects.]

|  |  |  |  |
| --- | --- | --- | --- |
|  | Estimated: | Measured: | Analyze: %Error |
| Time | Hours | Hours | % |
| Size | LOC | LOC | % |

EXAMPLE: Actual Project Time and Program Size vs. Estimates

# Appendices:

[Include Time and Defect Logs, along with any detailed planning worksheets, project schedule, Requirements Analysis, Design Documentation, Testing Plans and Outcomes, etc. that are referred to elsewhere in the report.

The Time Log is essentially a journal with entries (single line entries are fine) that include:

* Date
* Development phase or activity
* Time duration
* Comments, if desired.

(The Process Dashboard contains all of the expected fields for Time Log entries.)

Evaluating the Time Log should allow you to analyze your workflow patterns for possible process improvement actions.

The details in the Defect Log are important for identifying areas that need improvement and effective ways to avoid future occurrences of the same problems. Each Defect Log entry should include:

* A brief description of the defect or the observed “misbehavior”
* Some classification of the type of defect (logic, syntax, requirement specification, etc.)
* The phase of the project in which the defect was introduced
* The phase of the project in which the defect was discovered
* The time that it took to diagnose, find, and fix the defect
* An indication of whether the defect was introduced in the process of resolving another defect (a “Change Defect”).

(The Process Dashboard contains all of the expected fields for Defect Log entries.)

If using Process Dashboard, do not include “screen shots” of the Time and Defect logs; these are often very difficult to read. **There is documentation provided by the instructor on how to export these logs to Excel format**. Include the Time and Defect logs either as Excel sheets or as Tables in Word pasted from Excel.]