**

**ECE 4960: Computational and Software Engineering**

**Spring 2018**

**Programming Assignment 5: Code Evolution**

**1. Goals**

1. Learn how source code and regression testing can evolve from existing codes with persistent or revised structures
2. Top down designs for program specification and software quality assurance and quality control (QA/QC).
3. Extend one of your previous project to enhance the capability, interface, software engineering and testing coverage.

**2 Background preparation**

**2.1 Software quality management**

In the old days when the complex engineering system started around the 19th century, “quality” is roughly defined as “general goodness”, which is indeed more qualitative and ideological that cannot be easily measured or determined. Today, quality of any engineering system (as opposed to biological systems) including software is defined by reduction of variation, long-term value addition and conformance to specification. Alternatively, quality is the degree to which a set of inherent characteristics of a system fulfills published **requirements**.

**Quality assurance (QA)** consists of that “part of quality management focused on providing confidence that quality requirements *will* be fulfilled.” The confidence provided by quality assurance is twofold—internally to management and externally to customers. Quality assurance deals with the design and manufacturing process of the engineering system. **Quality control (QC)** is that “part of quality management focused on validating quality requirements.” Quality control deals with the black-box **testing** of the engineering system to the requirements. From this point of view, a product of the label of “QC” will have a specific test for each published requirement.

Most engineering systems start from the specification and requirement (top-down design or goal oriented), instead of what the present tool box can do (bottom-up capabilities or tool oriented). Surely the two approaches are more difficult to separate in the actual design process of a useful engineering system, as the final goal is to find a solution that can fulfill the requirements desirable to the end users.

Software quality assurance[[1]](#footnote-1) (SQA) consists of a means of monitoring the software engineering processes and methods used to ensure quality. The methods by which this process can be accomplished are many and varied, but a reasonable example can be found in ISO (International Standards Organization) 15504 (also termed as Software Process Improvement and Capability Determination, SPICE) and in Capability Maturity Model Integration (CMMI) as part of the Information Systems Audit and Control Association (ISACA) that deals with information technology (IT) governance. The most important part of SQA is how the software test suites can demonstrate the fulfillment of all requirements, which is often hugely complicated when interface to human activities is involved (such as Microsoft Office or Riot Games League of Legends).

**2.2 Evolution of your code and test suites**

We will take a practice for software evolution to a “product” that follows SQA practice and demonstrates QC. The process will consist of the following steps:

1. Propose a software that evolves from one or more of previous assignments. Within the proposal, you will need to illustrate your *implementation and testing strategy* that can fulfill your proposed requirements. Examples are (surely not as a restriction):
   1. A parser to validate the floating-pointing exception handling of part of the standard Gnu math.h as defined by ISO C89 and C99.
   2. A sparse matrix solver that contains both iterative and direct solutions with various matrix conditioning methods.
   3. Parameter optimization to extract circuit characteristics such as time constants or circuit parameters.
   4. A more comprehensive SPICE circuit simulator that includes more elements and DC solution.
   5. A 1D or 2D finite-element solver for the heat transfer equation with adaptive time stepping.
2. Create a Git Hub that inherits from the related previous projects. Your submitted Git Hub link will serve as a document for your development process (quality assurance) and progression of the test suites (quality control). This will be considered as part of the grading.
3. Create a report that contains two parts:
   1. An internal document for SQA that contains the implementation and testing strategy and the testing results. This is meant for internal software management.
   2. An external document to convince software adoption by end users. This is a practice, and you will not be graded by the number of downloads, although it is a popular measure in today’s IT world.

**3 Due dates and grading**

* This project can be done by groups of 2 students, but each student needs to know how to run the program.
* The proposal has an absolute due date before 5/1, but please iterate with me at your soonest possibility.
* All source codes, test suites and documentation should be submitted as a compressed file by email before 5/18 5pm. A Git link is required for the documented software and test suite evolution.

1. *IEEE Standard for Software Quality Assurance Processes*. 2014. [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1109/IEEESTD.2014.6835311](https://dx.doi.org/10.1109%2FIEEESTD.2014.6835311) [↑](#footnote-ref-1)