Software Engineering Sprint 2 – Spring 2014

Julian Brackins – Scrum Master reporting on behalf of LaTex Samurai

Hand-off Assessment – Software Provided by Kernel Panic

Overview:

For Sprint 2 of the Software Engineering Spring 2014 Course, our team, LaTex Samurai, is tasked with carrying out the new software additions for the Automated Grading System using the software designed by the Kernel Panic team consisting of Anthony Morast, Ben Sherman, and James Tilma. The purpose of this document is to assess the quality of both the documentation (both in the code and the LaTex document) and the actual software itself before concluding whether or not the code is sufficient or will need to be reworked.

Documentation:

Code Documentation:

The documentation within the .cpp files is somewhat minimal. There are very few inline comments, which is especially unfortunate in the situations where the directory traversal code is difficult to follow without having information provided by whomever wrote the software. This is noticeable in the getTstCases, get_folders, and get_files functions.

The team utilized doxygen headers to explain the purpose of each function, However, upon further inspection, it appears that a few of these headers were copy-pasted from other functions but then never updated to explain the function it represents (please see software analysis portion to see where this becomes an issue for code evaluation). Overall, there are a few notable spelling errors in the comments as well.

LaTex Documentation:

The LaTex Documentation overviews the algorithms well. However, there is not a lot of actual documentation that I could find explaining the structure for running the program. I found a usage statement in their grade.cpp file that tells me to run:

./grade <filename>.cpp

However, running that command throws up a bunch of errors since the executable cannot find the .cpp file to be run. I was unable to find any documentation noting where the grade executable should be located in relation to the .cpp file in order to run the software as expected.

Several sections of the documentation have been left out.

Software:

I made an attempt to compile the grade executable both within the same directory as the .cpp file and in the directory one level above the .cpp file, with no success in either effort. Trying to execute the program throws up an incredibly long list of errors, usually ending in a core dump. Looking at their documentation further, I came across this portion of documentation:

** The test*
** cases are stored in <filename>.tst and the answer to that test case is*
** stored in <filename>.ans. This program searches the directory level*
** where the executable is stored and all folders below it for test cases.*
** The students executable's output is compared to the answers in the*
** <filename>.ans files.*

According to this, the test cases for each .cpp file are set up with the same exact name as the file being tested (which goes against the structure of the test case suite provided by the instructor). Furthermore, this implies each file can only have *one* test case (or perhaps one test case per subdirectory). There is insufficient inline code to deduce whether this is actually happening (implying the comment instructions are simply incorrect) or if the grade executable is actually searching for the wrong test cases ("wrong" being defined as different from the test case structure provided by the instructor).

I looked at their getTstCases() function with the assumption that this is where the test cases are being collected. There are only two inline comments that basically say that the test case is only added to the tstLocations vector (which, if it's worth mentioning, is a global variable apparently) if a corresponding .ans file exists for it. Looking at the doxygen header, it's simply a copy-paste of their main function header so there really isn't a clear explanation as to what is going on in the function without sifting through the code line by line.

Conclusion:
Taking into consideration the fact that the software submitted to us appears to not match the test cases provided on your class website, I've concluded that our team will have to rewrite a fair amount of code. Besides merely rewriting some of the code to fit into the test case specifications, a considerable amount of code documentation will have to be implemented to gather a better understanding of algorithm structure and software usage, since neither is particularly clear in the files we've initially received.

Julian Brackins
LaTex Samurai Scrum Master – Sprint 2