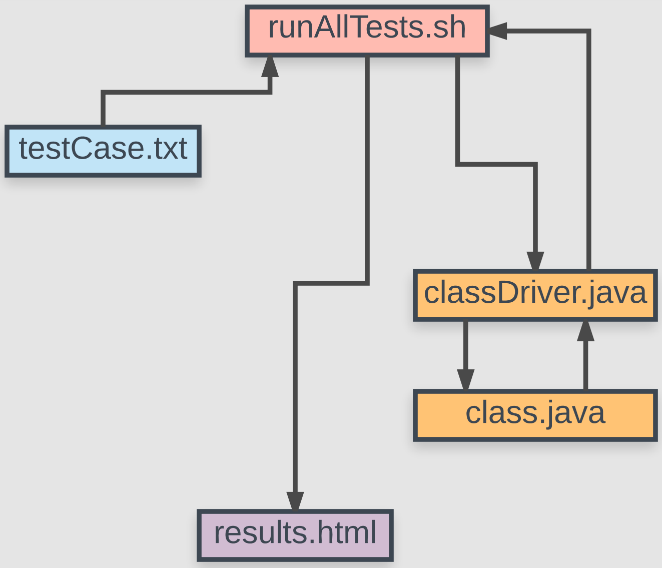
Automated Testing Framework for Sigmah

by Andrew Miller, Brian Steele, Noah Griffin
milleran@g.cofc.edu steelebm@g.cofc.edu griffinnr@g.cofc.edu

Introduction

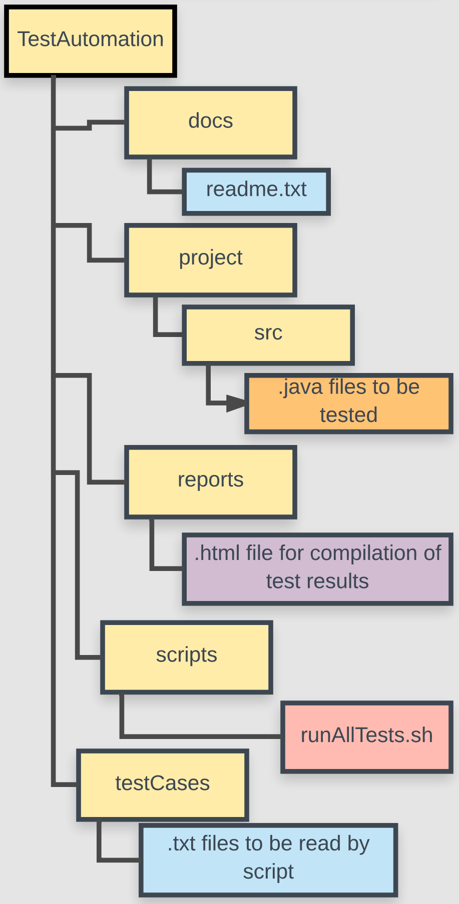
Large projects need tight coordination to be successful, and Sigmah hopes to bring success to international aid projects with their software. We chose to test a critical area of Sigmah's code: commonly used mathematical functions. Sigmah has specified certain mathematical standards by global declaration, and it is important that those globally declared methods be mathematically accurate. Using a fully automated testing process complete with 25 individual test cases, we determined that Sigmah's mathematical functions were indeed accurate (as predicted).



Pictured Above: runAllScripts.sh reads from testCase.txt files, and sends instructions to the relevant classDriver.java. classDriver.java sends instructions to the relevant class.java, collects results, and returns the results to runAllScripts.sh. runAllScripts.sh then reads the results, compares them to the expected results in testCase.txt, and stores the output in results.html. The cycle repeats for every testCase.txt in the testCases folder.

File System

Our testing suite had to be relative, requiring that existing in a specific location in a filesystem be unnecessary. To accomplish this, we used a uniform file structure provided by Dr. Jim Bowring with relative classpaths. The end result was a script that could run required files, analyze their output and record relevant results in an HTML document. Below is a diagram of the file system.



Experiences and Lessons Learned

Testing code is a cornerstone of software development. Being able to produce a clean and efficient testing suite is a very important skill for budding software developers. The road to cultivating testing experience is a dangerous one, however, with many lessons waiting to be taught.

The first lesson we learned as a group was to pay great attention to requirement specifications. We put a lot of effort into our second deliverable, but we "missed the boat" when it came to a few specific requirements. A second look at the project specifications left us with a clear goal for future deliverables.

Additionally, we were also taught the value of persistence. Before we had to demo our code live, we were at a full stop in development. We had a classpath error whose solution eluded our group and a professor that was asked for guidance. After a week of searching online for similar issues, our mistake was made obvious. A quick adjustment to the script resulted in a working automated testing suite with what were then just a few small test cases. That breakthrough energized our group and paved the way for success.

I suppose we reaped the most benefit from developing the project itself. As upper classmen, we were no strangers to collaborative projects. We were not aware of all the tools available to us, however, until we were encouraged to work on Sigmah and introduced to the bread and butter of project collaboration: GitHub. Our minor previous experience turned into veterancy as we pushed, pulled, and cloned our way to a well oganized repository. Having such clear communication and resource sharing promoted a highly professional environment, allowing us to spur ever onward toward our goals.