WENTWORTH
Institute of Technology

Launch Control System Software Development Intern

Tom Plano
Kennedy Space Center
Major: Computer Science
Spring Session
Date: 4-03-2017

Spaceport Command and Control System Automation Testing

Tom Plano
Wentworth Institute of Technology, Boston Mass, 02115

The goal of automated testing is to create and maintain a cohesive infrastructure of robust tests that could be run independently on a software package in its entirety. To that end, the Spaceport Command and Control System (SCCS) project at the National Aeronautics and Space Administration's (NASA) Kennedy Space Center (KSC) has brought in a large group of interns to work side-by-side with full time employees to do just this work. Thus, our job is to implement the tests that will put SCCS through its paces.

Nomenclature

GUI:	Graphical User Interface
KSC:	Kennedy Space Center
LCC:	Launch Control Center
NASA:	National Aeronautics and Space Administration
OCR:	Optical Character Recognition
Scripting Language:	High level, interpreted programming language
SCCS:	Spaceport Command and Control System
QC:	Quality Control

I. Introduction

The goal of this internship was to continue the development and expansion of the extensive array of automated tests that would be used to test the GUIs within SCCS software. The express purpose of this test suite is to free up the resources and time of those individuals who would otherwise be required to manually test each and every aspect of the user interface and user experience before each and every release of SCCS. This was a form of both active testing of the new features that were added and also a means of regression testing to check that no previously implemented features that had been successfully developed were now faulty. Automating these predominantly manual, image based tests also allows the development teams to consistently run the same tests over and over again without human error.

The major pieces of the test suite that we used to develop our tests was an automated test framework built on top of the Python programing language as a parser for generic files in which we would place keywords and directives. With this, we can automate any and all test procedures that we could be asked to work on.

II. Objective

The objective of this internship was to automate a given set of test steps in such a way that the testing of the SCCS software no longer needs to be done by hand by a QC engineer. These test cases come to us in the form of spreadsheets and our deliverable is a runnable script that would do precisely the same execution and verification of those steps that a QC engineer would do.

When we, the interns, could not work on our automated testing development in the LCC because of scheduling, we busied ourselves with other projects that were assigned or suggested by one of our technical mentors. One such project was research into software-based character recognition for the purpose of more easily reading user interfaces and text on-screen. This tool, should we integrate it into the testing suite proper, would do scans of the display and return a text document that we could use to map between actual images on that display and a memory location that would allow for direct calls to the display from our test scrips. Another such project was to develop a set of scripts or tools that would allow for the parallelization of multiple tests to be run across a network and to be coordinated by a central server.

III. Approach

With the goals outlined above, my approach was very straightforward and can be broken down into a few simple parts. As our first goal was to get very familiar with the automated testing framework, that is where I spent a majority of my time at first. Following that, I found time to exercise my preexisting knowledge of a scripting programming language. Finally, there were discussions toward the end of the internship about future plans of the automated testing framework, so that is also addressed.

A. Training/Familiarization

The first few days of the internship were all about training and learning the ropes of being on center and an employee of NASA. After that began the real work. Very quickly we were thrown right into the mix and given assignments to try and complete them as best as we could. For the first few weeks this was a struggle for me as the tools were new and it was an unfamiliar development environment. However, after completing my first few test cases, I got into a flow and was feeling confident in my abilities.

B. Automated Testing for the main SCCS GUI

Automation of the testing for the main SCCS GUI was done in a general-purpose automation framework. The tests are written in the form of “Label” followed by “Keyword”, where keywords are essentially macros that map high level strings down to operating system level commands to interface with GUIs and move the mouse. In this way, the testing framework is able to obfuscate the particulars of test execution while still providing a robust set of rules that can return with certainty whether or not a test step has passed. The test steps that we are to automate come to us in form of spreadsheets. These are the exact manual tests that an engineer would have to run if it were not for our work of automating. Most of my work consisted of determining where, if applicable, I could use preexisting keywords to accomplish the goal of fast, effective automation and where, when necessary, a new keyword would have to be created to serve the purposes of some new class of test or a new type of test step. The simplest type of test is one in which no new keywords had to be created, but it was more common that at least one new keyword would have to be created per test case. This constituted the bulk of my work this semester, alongside the maintenance and upgrading of previously written tests; there were that may have run correctly on a previous build of the SCCS program but now either needed to test new features or just had new test steps.

C. Scripting

One of the main projects that I worked on other than creating and maintaining test scripts was to create a test script manager and parallelizer. This was a full project that I volunteered for in addition to the work I was doing on creating test scripts. This process started out with an extensive design meeting, where myself, Jason Kapusta (a software architect), and other members of the team discussed ideas and outlined requirements for exactly what my software tool should be doing. Those requirements were as follows:

1. Be able to utilize independent multiple workstations on the development set
2. Sequence test execution at the test step level
3. Collate results from all stations into one report
4. Be highly maintainable for those who come after me
5. Be implemented in the language of our automation framework
6. Control multiple work stations simultaneously
7. Control multiple work stations in sequence
8. Pass data between work stations
9. Handle failure states on a workstation such that the rest of the test does not crash
10. Allow for the parallelization of multi-user processes

With these design goals in mind, I was able to get right to work on the tool development. The requirements listed above are in no particular order and as such I was able to develop first those requirements that I saw as most difficult or integral to the main functionality of the tool. As my first goal, I focused on requirements 1, 2 and 5. As previously stated, our automation framework is built on top of a scripting language and so an extension that would correctly interface with that program would also have to be written in that same language. I began by white boarding a design solution that would accommodate those requirements.

My design involves a single control node, written in that same scripting language, that would connect to remote boxes and initiate specially written automated testing scripts that contain control flow arguments. This control node also initializes a type of lock mechanism know as a semaphore for each running process in a shared network file where all processes and the control node can read and write from. My solution revolves around this lock file being polled by each testing process as well the controller. The controller is fed a sequence file that rewrites the lock file on a state change initiated by the test scripts rewriting the locks file when they complete a step. The controller will then rewrite the lock file with the next step in the sequence file. With these simple things implemented, requirements 1, 2 and 5 all are met. The other requirements will be met some time after the creation of this document.

D. Future Developments

In its current state, the SCCS automated testing framework is progressing nicely. More tests than ever are being run independently of human interaction and entire swaths of the test suite are being run over night. However, there is always more work to do when it comes to creating and maintaining a robust set of tests. To that end, plans for the future of the project are in the works. It has become clear that the future of our software package is in the creation of customized control processes that will manage and allocate test processes running on multiple, independent workstations across the network. In fact, these design and development meetings have already begun, but as their implementation will be completed after the termination of my internship, it only seems fitting to include it as future work.

IV. Conclusion

Since the beginning of this internship, a great amount of progress has been made on the development of our automated testing suite. There has been an almost 60% increase in the number of completely automated test steps over the beginning of the semesters. Additionally, significant progress has been made in the form of scripts, tools and utilities that will be running alongside the main automation scripts. That's not to say that the whole process was without its difficulties along the way, though. There were times when our normal development set was disabled or otherwise unavailable. At those times, it was down to our abilities and training to continue the development process without being able to work on a live set. As our completion numbers will attest, however, this was not a great detriment to our productivity, a fact that we are all very proud of. Knowing that our tests and tools will be helping to send humans to Mars and beyond has been a large motivation and really pushed to work hard on all our projects and it really is exciting to be part of something so big.

V. Acknowledgements

This internship has been such a transformative experience for me. Moving so far from home, working in such a prestigious institution, and managing new types of work with a vastly different work environment have been a challenge for sure. There was so much support from all sides. First of all, I would like to thank Caylyne Shelton, Jamie Szafran, and Oscar Brooks for their continuing support and assistance whenever I needed help or guidance. Even for the most simple or trivial things they were always available and supremely helpful. I would also like to thank Jason Kapusta for his thoughtful and technical comments on code as well as offering me the opportunity to stretch myself beyond the specific bounds of my role in some of the side projects he suggested. Finally, I would like to thank the team of both interns and full time employees with whom I found myself working on a regular basis - Sam Bridges, Susan Pemble, Michael Backus, Abraham Glasser, Andrew Hwang, Mark Rodriguez, and Meriel Stein all made for a wonderful team and made it fun to come to work every day.