CASE WESTERN RESERVE UNIVERSITY EST. 1826

think beyond the possible℠

Spaceport Command and Control System Software Development Intern

Andrew Hwang
Kennedy Space Center
Major: Computer Science
Spring Session
Date: 19 04 2017

Spaceport Command and Control System Automation Testing

Andrew T. Hwang[1]
Case Western Reserve University, Cleveland, Ohio, 44106

The Spaceport Command and Control System (SCCS) is the National Aeronautics and Space Administration's (NASA) launch control system for the Orion capsule and Space Launch System, the next generation manned rocket currently in development. This large system requires high quality testing that will properly measure the capabilities of the system. Automating the test procedures would save the project time and money. Therefore, the Electrical Engineering Division at Kennedy Space Center (KSC) has recruited interns for the past two years to work alongside full-time engineers to develop these automated tests, as well as innovate upon the current automation process.

Nomenclature

CSV	= Comma Separated Values
GUI	= Graphical User Interface
KSC	= Kennedy Space Center
LCC	= Launch Control Center
NASA	= National Aeronautics and Space Administration
OCR	= Optical Character Recognition
MD5	= Message Digest algorithm that produces a 128-bit hash value.
SCCS	= Spaceport Command and Control System

I. Introduction

The importance of this internship is to make the software development process more optimized by reducing the amount of manual testing for software to be used for future Human Spaceflight Program launches by developing automated software testing. This will allow the software engineers, who previously spent large amounts of time working on these tests, to spend their skills and time on other aspects of the project. In addition, the automated tests can help avoid human errors that would possibly appear after hours of rigorous testing. Once these automated test cases pass high-quality verification and contribute to a successful launch, we can be assured that the automation will run the same way for future testing.

The major pieces of software that we were using to develop our automated tests include an automated testing framework and an automation library. The automated testing framework has tabular test data syntax and utilizes keywords to operate automation features. The automation library contains functionality to automate anything that appears on a desired screen with the use of image recognition software to detect and control GUI components.

The interns who worked on the automated testing project were Michael Backus, Abraham Glasser, Thomas Plano, Mark Rodriguez, Meriel Stein, and myself.

II. Objectives

The main objective of this project was to automate the testing of given sections of a system GUI that would use streamed simulated data from the testing servers to produce measurements, plots, statuses, etc. to the GUI. There are spreadsheets of test procedures and each section contains step-by-step test cases that would normally be followed by engineers.

When we, the interns, could not work on our automated testing development in the LCC because of scheduling or because a system reprovisioning was in progress, we would either attempt to "blind code" the automated test procedures to the best of our abilities or work on side projects that were individually assigned to the interns. My alternative assignment was to develop a script to setup a user's remote accessibility. My second alternative assignment

[1] Spaceport Command and Control System Software Development Intern, NE-XS, Kennedy Space Center, Case Western Reserve University

was to develop a script to update the password of multiple extra pseudo users, which are for the use of the automation team.

III. Approach

My approach to effectively tackle the assignments that were given to all the interns is to first properly familiarize myself with the objectives and tools that would be utilized by researching into each topic. Then, I'd proceed to run tests, learning from trial and error until I have a solid understanding of the most optimal use of the tools and software I am using.

A. Training/Familiarization

a. Automated Testing

I have been working with the automated testing framework for about 7 months now, but some changes were made about two weeks ago to the file organization of the resource files. All of the general keywords normally used for our test cases had been separated and categorized based on to what the keywords are commonly applied. This change took a little adjustment for the team, but in the long term it will provide a more efficient process of searching for desired keywords.

Another new addition was the introduction of sprint meetings to help keep track of which tests were assigned to who and see how much the automation team has progressed. The meetings are held every other week, where we reflect on what we have done well, what we could improve upon, and what we should keep in mind for the future.

b. Scripting

The script is written with a command language that is commonly used in operating system terminals. I have had past shell scripts assigned to me by my project lead, so I have a good understanding of programing the new scripts I was assigned. There were a few times I had to research for a functionality of the command language to perform a task that I could not program with my base knowledge. Thankfully, I had the help of my fellow interns when I needed a different perspective of how I should approach the issue I was facing.

B. Automated Testing for SCCS Dashboard

For the primary objective of this internship, we were tasked with the responsibility to automate the testing process for the SCCS dashboard. To know the testing procedure for the test case that needs to be automated, we were given spreadsheets with each test step in a row with the operator action and expected responses.

Step #	Operator Action	Expected Response
12	Launch display, "Some Display"	"Some Display" display appears
13	Execute the command script "Some Script", to Load data	Values on "Some Display" are changing

Table 1: Example of a couple of Test Steps from a Test Procedure Spreadsheet.

After looking at the existing test steps, it seemed to make the most sense to initially run the test steps manually, as if to mirror the process of a testing engineer. Then, once understanding the whole process of the test step, the next step was to check for any suitable keywords that were found in the libraries, and thus starts the trial and error process. Once a selection of keywords and a working algorithm were established, we were ready to do a first run of the automated test step. If the test failed, the first step to take would be to take a look at the produced report file and read through the specifics of the errors. If the test passed, the next step would be to verify that the test passes for all possible edge cases.

a. Technical Specifics

While writing test steps that will automate a GUI, there are buttons, text fields, editable value fields, and dropdown menus that need to be considered. This requires precise selection of those desired areas on the screen, so the process the automation team takes is to either produce an image of the text we desire to interact with or take a capture of the image with which we wish to interact. We are able to perform these tasks with the use of keywords that compare images from an image library to either the entire screen or a selected area of the screen; this is all possible with the use of the automation library's image comparison capabilities.

b. Test File Formatting

The automated testing framework has a tabular-style syntax, which means the functionality of a line of code must have the appropriate number of tabs for the line to function as intended. The header section contains either paths to custom resources or the names of libraries being used; in the automated testing team's case, the automation library and common function file would be included in the header. The data section contains any data values strictly created for the current automated testing framework. The body section holds the tests that are being run, but requires a title for each procedure, and there is no limit to how many procedures that can be made; however, all of the procedures must run sequentially based off where it was written in the file. The function section can include any number of functions and does not run sequentially, but runs in whatever orientation the function was called by a procedure or another function, and these functions may only be used by the current automated testing framework file or any other file that resources it. The resources and body section are required for all test files; the data and function sections can be left empty if the data values and functions being used are from a resourced library or file.

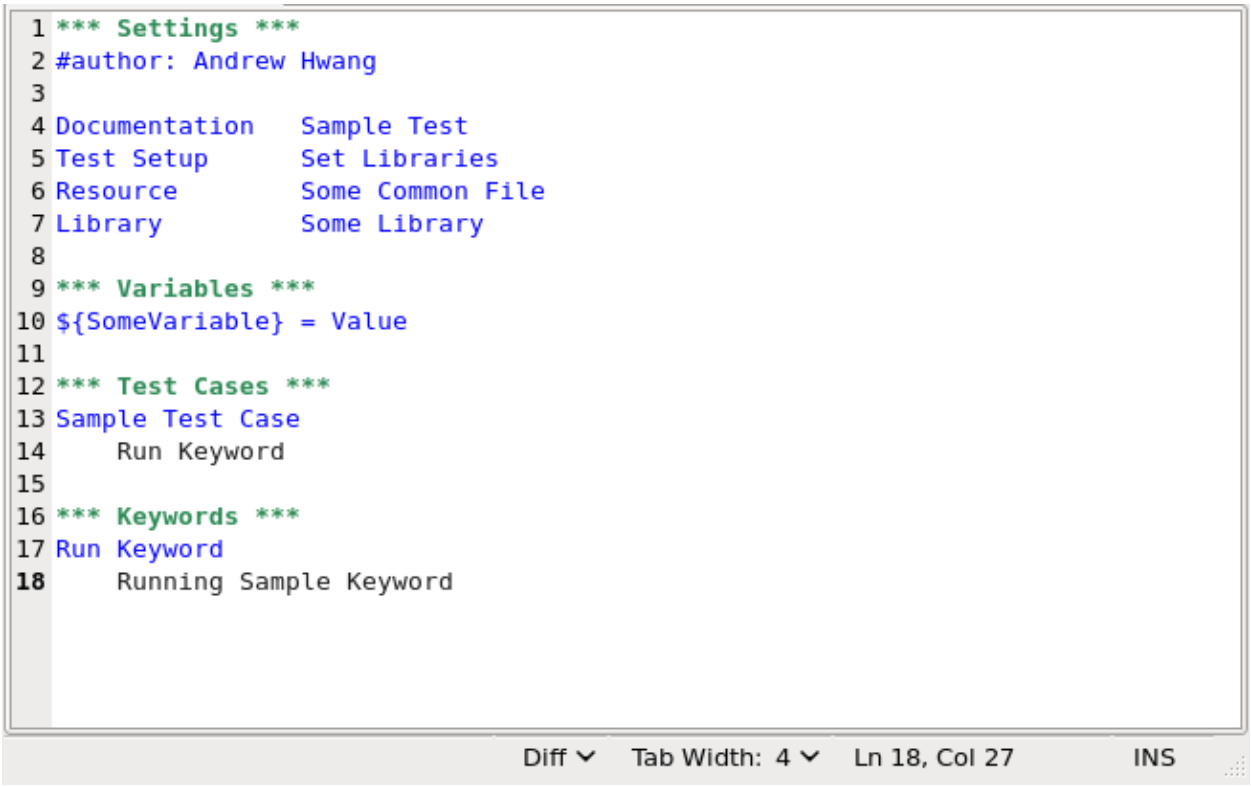

Figure 1: Format of Sample Automated Test File.

C. Script

The Project Lead of the Automated Testing Team, Jason Kapusta, assigned additional projects to a few of the interns. The first project assigned was a script that would setup the current user's remote accessibility. The second script will modify the authentication for a group of pseudo users, which are used to for multi-user automated tests for the automation testing teams.

D. Future Developments

As of now, the automated tests rely on the image recognition capabilities and the images that we produced and captured, but we are getting closer to using a more effective method of OCR. This will only require the automated test development to take screen shots of the entire screen, which is far more automatable than specifically capturing and creating images whenever something on the screen is updated. This will also make the automated test cases more adaptable to a change in the screen's environment.

The Setup Script and Updater Script will help with automating these common setups for every user account and make a user's setup faster and simpler. The scripts are very modular, which means they can be adjusted easily for a more specific or general application.

IV. Conclusion

Since a majority of development and testing for the automated test cases for the GUI in question has been done at the LCC, a large amount of progress has been made. As of this writing, about 60% of all of automated testing has been implemented, but as the GUI continues to be updated, some future edits will have to be done to make the automated test cases compatible with the latest version of it. At times the consoles at the LCC would not be receiving simulated data, which is required to run tests accurately; this would result in temporary restrictions on development of automated test cases. On the days that this happened, the next prioritized task would be to work on any additional projects that were assigned. There are still more automated test cases to finish, but even once these objectives have been met, it's best to remember that all of our efforts are contributing to the mission to get mankind to Mars.

Acknowledgements

I have a many great people to thank for all of the accomplishments I was able to achieve throughout this internship. I would like to first thank Caylyne Shelton for continuously offering her most gracious generosity, assistance, and time, even if the help needed was minor assistance for simple project concerns or asking about programming advice and assistance. Next, I would like to thank Oscar Brooks for always thinking of the interns' safety and interests first. Additionally, I would like to thank Jamie Szafran for her great assistance and honest advice for any general questions I would have. Also, I would like to thank Jason Kapusta for always being patient with the interns and answering every question that would be thrown at him with a smile. Lastly, I would like to thank all of the automated testing interns and full-time engineers: Michael Backus, Raymond Bridges, Abraham Glasser, Susan Pemble, Thomas Plano, Mark Rodriguez, and Meriel Stein, for all working so well together as a team with great communication, patience, and support.

References

Spaceport Command and Control System Automation Testing, A. Hwang, 2016