Command and Control Software Development

Michael Wallace
NASA Kennedy Space Center
Major: Computer Science & Engineering
Program: NIFS Spring Session
Date: 04 13 2018

Command and Control Software Development

Michael Wallace[1]
University of California, Merced, Merced, California, 95348

The future of the National Aeronautics and Space Administration (NASA) depends on its innovation and efficiency in the coming years. With ambitious goals to reach Mars and explore the vast universe, correct steps must be taken to ensure our space program reaches its destination safely. The interns in the Exploration Systems and Operations Division at the Kennedy Space Center (KSC) have been tasked with building command line tools to ease the process of managing and testing the data being produced a ground control system while its recording system is not in use. While working alongside full-time engineers, we were able to create multiple programs that reduce the cost and time it takes to test the subsystems that launch rockets to outer space.

Nomenclature

API	=	Application Programming Interface
CSV	=	Comma Separated Values
KSC	=	Kennedy Space Center
LCC	=	Launch Control Center
NASA	=	National Aeronautics and Space Administration
NE-XS	=	Engineering Directorate, Exploration Systems and Operations Division, Software Branch
SLS	=	Space Launch System

I. Introduction

NASA has always strived for achievements that have never been accomplished by mankind before. After reaching the moon with the Apollo missions, establishing a continuous human presence in space with the International Space Station (ISS), and exploring the cosmos with the Hubble Space Telescope, NASA's ability to advance our knowledge of the unknown is unquestioned. Today, the software that will control everything from the ground systems to the flight systems are currently being developed for the Orion capsule and the Space Launch System (SLS), which will serve as part of the next phase of space missions planned by NASA. The systems and their subsystems will launch the manned rockets which have the lofty goals of bringing humans to Mars, farther than any human has ever traveled. The journey to Mars takes years of careful planning and execution from all of its employees.

The interns that come to KSC's NE-X division during all seasons of the year also play a vital role in contributing to this effort through their projects, no matter how big or small those might be. By plotting out various tasks and properly planning projects, the quality of the results will increase thus leading to more reliable systems and more possibilities for the future. Knowing all of this, the task of creating a command line tool that would actually be used by project software developers was very rewarding. The ability to efficiently record and manage data from components of the system when the main recording system was not in use was what my team and I worked on during my time here at KSC.

[1] Command and Control Software Development Intern, NE-XS, Kennedy Space Center, and University of California, Merced.

II. Objectives

It is crucial that the systems supporting the command and control systems are fully functional and provide the user with viable data when being run. Our technical mentor, Jason Kapusta, assigned an important project to Craig Einstein, Zachary Geddes, Ricky Velasquez, and myself, which was to create two programs to assist the software developers in testing and analyzing the data being received from the system's components. The data being received are identifiers, and can be described as alphanumeric strings that identify the type of information that they represent. Examples of identifiers are measurements, commands, and various other types that all play a role in making the system function properly. Craig and Zachary were in charge of building a data generation tool, while Ricky and I were instructed to build a data recorder tool. The data generator tool would be able to generate and publish specific identifiers on demand as specified by the user or attached configuration file. This would ensure the messages were being properly passed between the systems and is crucial for testing. The data recorder tool would subscribe to the system and receive all of the data being published. This data would be in the form of identifiers and would be stored for testing purposes either manually or by an automated test. This was an interesting task, mainly because we were building the program from scratch and had to brainstorm different strategies on how to bring our final product to life. The idea of our tool was that it would be an option for recording data that was propagating through the system even when the main recording systems were unavailable or offline. Our goal was to create and automate the tool so that it parsed a configuration file for its arguments and would store the received identifiers into their own respective output files. This would make the ability to navigate and troubleshoot the information easier for the developer. We were aiming to have the produced output format be CSV and suitable to be imported to a spreadsheet.

III. Approach

When the tool launches, it is supposed to subscribe to the system that is publishing data and record it for future testing and troubleshooting. Ricky and I decided to approach the data recorder project by simulating its functionality before implementing the actual system's components. We made text files to mimic the data flow that happens when the system is subscribed to, and then stored the information from those files to their own queues. We used multithreading to convey data from the text files to the queues and from the queues to their output files in order to manually manage how the data was flowing, as well as to prevent any identifiers from being missed or overwritten throughout the process. The next step was to apply the API and remove the text files entirely. This would connect it to the system and receive the real identifiers instead of the information from text files. Once this is finished, the project will be nearly complete. Some additional features that could be added would be to designate a specific directory to send the information to instead of hard-coded output files. Another would be to add another key that allowed the user to quit the application such as typing "q" and pressing <enter>, instead of just doing <Ctrl-C>. As a final product, the tool should be able to read and record the data being published quickly and with all foreseen possible error conditions properly handled.

IV. Results

The command line tool being built provides a simple and efficient way of receiving and recording messages sent through the command and control system for testing. As of now, we are finished with the data recorder tool. The basic functionality of the program is complete. By the end of the 16-week internship term, the program will be fully functional for everyday use by any system developer. The API function calls that subscribe to the system and receive the identifiers into queues are fairly straightforward, and have been implemented in our main program. Once we feel more comfortable about the cleanliness of our code, it will then undergo further unit testing before being submitted for an official code review. Additional features will also be added with the remaining time left before the completion of the internship. Overall, this is a useful tool that can be used to fix and find issues within the system's components while the main system is offline.

V. Conclusion

The tools that my team built provide efficient methods that can read and record information from the system without user input. Testing can take up a large amount of time, so having an automated test tool for developers to use is important to keep the testing process well-organized. This project has taught me many lessons on how to successfully work on an impactful project, such as being a good team player and a leader. The performance of this tool aims to directly assist the project software developers with their testing of the system, which is why this internship has been so special. Some additional features can also be added in the future to improve the tool itself. NASA's innovation has kept them relevant for the last six decades. Whether it be the shuttle missions that took humans to Earth's Moon or discovering new galaxies through high-tech telescopes, they have always used science and mathematics to make breakthroughs in modern technology. Our roles as interns are invaluable as we provide a fresh and ambitious approach to the projects we're a part of.

Acknowledgments

I would like to personally thank Jason Kapusta, our technical mentor, for offering excellent feedback for any and all questions throughout the 16-week term. Also, Jamie Szafran, Caylyne Shelton, and Andrew Davis for providing an astounding work environment that allowed all of the interns to work efficiently and grow as a professional at their own pace. I am truly grateful for this incredible opportunity, and have learned countless new skills that I will keep with me for the rest of my life. I am extremely thankful to have had fellow intern Craig Einstein on my team this fall semester. He has played a huge role in the development of my ability to use C and Emacs as a means of accomplishing tasks efficiently within a project. The ability to work on a project that includes many key concepts covered in school while still contributing to the future of NASA's space program still amazes me.