Realistic Data Generation and Publication for Launch Control Systems

Gregory Hubbard
Kennedy Space Center
Spaceport Command and Control Software Intern

12 - 02 - 2019

Realistic Data Generation and Publication for Launch Control Systems

Gregory W. Hubbard[1]

Old Dominion University, Norfolk, VA, 23529

Nomenclature

CSV	= Comma Separated Values
EM-1	= Exploration Mission 1
GDB	= GNU Debugger
IDE	= Integrated Development Environment
KSC	= Kennedy Space Center
LCC	= Launch Control Center
LCS	= Launch Control System
SCCS	= Spaceport Command and Control System
SLS	– Space Launch System

NASA's Launch Control Center or LCC requires an extensive infrastructure to manage a successful launch. Within the LCC, the Launch Control System (LCS) contains the Class A, human-rated, mission critical support software. Before deployment in the LCC, software must be thoroughly inspected. To accomplish this goal, automated testing and software simulations are employed. Automated testing is used to validate the software using realistic operator commands. Data for simulated operator commands and related information can be generated using software that simulates actual real-time commands and telemetry. Using both command end-items and received telemetry, this generation tool will allow LCS to quickly verify procedures before operational use, thus protecting valuable assets.

Introduction

This paper will discuss my internship developing software for launch control systems over the Fall 2019 semester. In particular, I will describe the architecture in place, my assigned tasks, and their impact contribution. Outside the technical basis, my internship also exposed me to numerous universal software engineering practices. Finally, I will remark on aspects of this internship, which can only be described as unique and particular to NASA Kennedy Space Center (KSC).

Overall, this is a historical and exciting time to work at NASA Kennedy Space Center. We are on the forefront of the next major human space flight program. In preparation for this paradigm shift, NASA Kennedy Space Center is retrofitting hardware to support the newly announced Artemis program. This program's heritage lays in the foundation of NASA's prior human space flight programs: in particular, Mercury, Gemini, Apollo, Space Shuttle, and Space Station.

Kennedy Space Center stands as a geographical testament to the legacy of the United States Space Program. Since 1968, KSC has been the primary launch center in North America. In memorable history, Kennedy Space Center exclusively launched all 135 Space Shuttle missions. Now KSC looks to shift its objective towards lunar monopolization and human-based Martian exploration.

This new direction will begin with Artemis-1. Initially coined as Explorer Mission 1 (EM-1), Artemis will operate as a functional test of the Space Launch System (SLS) rocket, Orion capsule, and related ground systems. Given successful feedback, Artemis-1 will lead to annual manned lunar missions. Each sequential launch will develop lunar infrastructure towards a permanent American presence on the moon. These foundations will bring humanity closer and closer to “boots on Mars” with a projected Martian landing by 2030.

An operation of such scale requires the support of all ten NASA centers across the United States. With the program already well on its way, testing procedures are in development to secure the initial (Artemis-1) launch.

Objectives

From a top level, the core requirement of my internship was to assist in the engineering of Class A, safety-critical, human-rated spaceflight software. This development included participation in the complete software development lifecycle with full-time engineers in adherence to NASA’s agile development processes.

One portion of this developmental initiative was to automate redundancies across the Launch Control Center (LCC). Launch Control software requires meticulous testing before approved use. Test automation software increases effectiveness and time efficiency for NASA engineers. Due to the size and scale of a space launch operation, manual testing would be timely and prone to human error. With a majority of tests under automation, engineers can focus their skills on other critical aspects of the mission. Once the respective subsystems are verified, the system can be used to launch the vehicle.

Developing automated tests is a large project and requires many dependencies to simulate a pseudo launch operation. A portion of this real life operation includes the rapid generation and transmissions of data for test use. In order to ensure tests are as realistic as possible, a simulation tool must be used to produce accurate data. Over my 16 weeks at NASA Kennedy Space Center, I primarily focused on adding features to this data generation tool.

Analysis

The data generation tool is intended to simulate data akin to actual commands sent by the launch control team in the LCC. This tool allows for simple feedback between test communication components for automated testing. From a high level approach, it is a helper tool that simulates the middleware message bus.

This message bus allows for publication and subscription of specific items that are sent over the network. The network combines multiple frameworks, such as data generation, to interact in realistic automated tests. This message bus is intended to process three main kinds of information:

Measurements

Data information is accessible and transmitted with specific content. Data, along with associated values are sent out to numerous listeners.

Triggered Data

Much like data, this data is sent to multiple listeners. It has no active state and corresponds to a triggered process.

Commands

Command information is sent to a specific end item. A typical command would return data specifically to the requester.

Information created for automated tests can be further validated and recorded using this tool. A file can be modified to change input and output data. This allows for added use case functionality.

Task

My work on the generation tool allowed me to familiarize myself with the new topics and frameworks. Through a combination of NASA’s employee training courses, instruction from mentors, and independent study, I was able to adequately adapt for my assignments.

One feature I directly contributed was the implementation of standardized measurement timestamps. This amendment to data processing ensures consistency across multiple machines. All measurement information parsed is tokenized into main CSV components as outlined in figure 1 below.

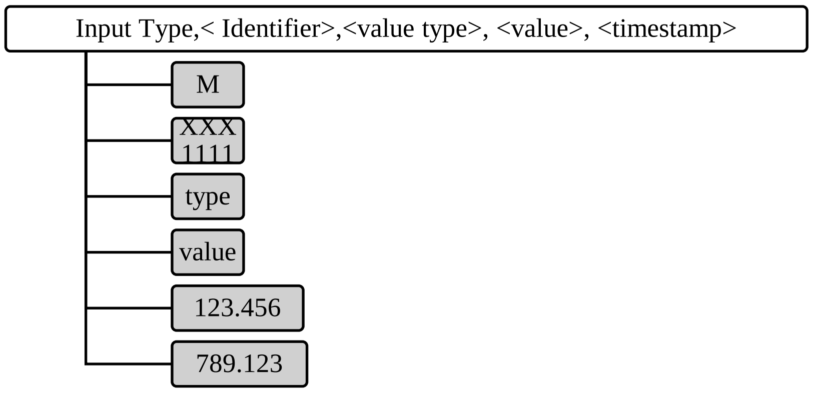

Fig 1. Example tokenization of data

The timestamp is further broken down into a source time and acquisition time in seconds. Through standardization, users can confidently specify times or rely on their machine's system clock.

Similar to sending measurements, the generation tool can also send identifier messages. These messages contain identifier metadata. The purpose of this metadata is to specify values that are out of range and need action. My task was to expand on the system for parsed alarms.

In addition to these primary goals, I implemented unit testing in order to follow NASA's software engineering standards. Using agile practices, development was regularly supervised with peer reviews and configuration management.

Conclusion

The data generation tool allows for clean and simple testing of LCS components. Generation of realistic data guarantees increased coverage for automated tests. My work added features to increase usability and functionality of this tool. I am confident that capabilities added by my code will streamline the software testing process and will contribute to NASA's ambitious launch schedule. Through working with my team, attending weekly tag up meetings, and participating in code reviews, I experienced NASA's software development lifecycle. In summary, I learned a tremendous amount about the development of large scale engineering operations. I look forward to the nearing launch of SLS, and seeing that I contributed to its success.

Acknowledgements

I would like to thank my mentor, Jill Giles, as well as the two Software Architects, Jason Kapusta and Tony Ciavarella for their constant leadership and guidance. Additionally I would also like to thank Kennedy's Education team for their administrative help. Finally, I'd like to thank the team of software engineers I spent the past 16 weeks with. It has been a privilege to work alongside such remarkable individuals.