Software Development and Testing Support for the Avionics Systems Telemetry Tool Suite

Antonio José Negrón-Martínez

Major: Computer Engineering

Taylor Walter Thomas

Major: Applied Math and Computer Science; Information Communication Technologies

NASA Kennedy Space Center

2018 Fall Session

Date: 16 NOV 2018

Software Development and Testing Support for the Avionics Systems Telemetry Tool Suite

Antonio José Negrón-Martínez[1] and Taylor Walter Thomas[2]
John F. Kennedy Space Center, Kennedy Space Center, FL, 32899

Abstract

The Customer Avionics Interface Development and Analysis (CAIDA) team helps to provide modeling and simulation software for the verification of the Launch Control System (LCS). With a new iteration of telemetry tools being developed, extensive work must be done to ensure features are implemented in an efficient manner. The authors worked to develop new functionalities in the telemetry tools, update documentation, and perform various tests on the CAIDA Advanced Telemetry Tool (CATT). This was accomplished with Python through built-in library frameworks. In addition, work needed to be performed to set up a training document for new engineers and interns joining the team in the future. The outcome of this internship was the completion of several new features, unit and functional tests on CATT, thorough documentation, and a developer's guide to programming under CAIDA.

Nomenclature

CAIDA	=	Customer Avionics Interface Development and Analysis
CATT	=	CAIDA Advanced Telemetry Tools
GFAST	=	Ground Flight Application Software Team
ICPS	=	Interim Cryogenic Propulsion Stage
IDE	=	Integrated Development Environment
KSC	=	Kennedy Space Center
MPCV	=	Multi-Purpose Crew Vehicle
NASA	=	National Aeronautics and Space Administration
NE-XA	=	Exploration Avionics Branch
SLS	=	Space Launch System
UML	=	Unified Modeling Language

[1] Fall Intern, Avionics Branch, NASA Kennedy Space Center, Inter-American University of Puerto Rico, Bayamon,
[2] Fall Intern, Avionics Branch, NASA Kennedy Space Center, University of Wisconsin – Stout.

I. Introduction

Throughout the 2018 fall term, Antonio Negrón and Taylor Thomas worked at the National Aeronautics and Space Administration (NASA) Kennedy Space Center (KSC) in the Exploration Avionics Branch (NE-XA) with the CAIDA team. The work performed was on the CATT project, with Mr. Dean Orr as the supervisor, and with Mr. Walter Wehner as Antonio's mentor and Ms. Jill Giles as Taylor's mentor.

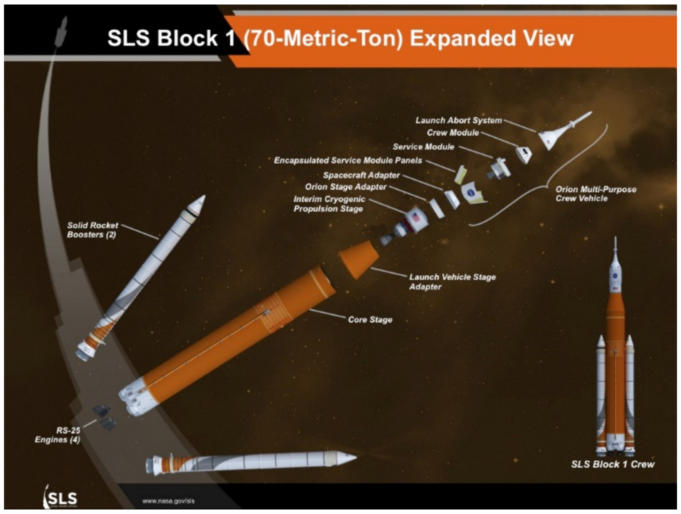
SLS Block 1 (70-Metric-Ton) Expanded View
Launch Abort System
Crew Module
Service Module
Encapsulated Service Module Panels
Spacecraft Adapter
Orion Stage Adapter
Interim Cryogenic Propulsion Stage
Orion Multi-Purpose Crew Vehicle
Solid Rocket Boosters (2)
Launch Vehicle Stage Adapter
Core Stage
RS-25 Engines (4)
SLS Block 1 Crew
SLS
www.nasa.gov/sls

Figure 1 - SLS Expanded View

The CATT project is intended to be used by groups such as the Ground Flight Application Software Team (GFAST) and testing groups wishing to verify vehicle and spacecraft processing requirements of the LCS. This includes processing telemetry and command data from the Space Launch System (SLS), Orion Multi-Purpose Crew Vehicle (MPCV), and Interim Cryogenic Propulsion Stage (ICPS). The CATT project emulates the telemetry from the SLS and MPCV through different communication protocols that allow GFAST to properly test their software before testing it against the SLS, MPCV, and ICPS hardware.

The tasks and responsibilities available were composed of different steps in the software development life cycle. These tasks are listed below:

- Build CATT software design documentation
- Build developer's guide
- Update sequence diagrams
- Update Unified Modeling Language (UML) diagrams
- Noise/mask feature development
- Multi-Measurement Sequencer feature development
- Unit testing CATT
- Integration testing CATT
- Functional testing CATT
- Verification testing CATT
- Document features (Noise/Multi-Measurement Sequencer)
- Code review CATT
- Telemetry Inspection engine

To successfully accomplish the available tasks, a set of objectives were set and sought out in this internship. These objectives can be seen below:

- To create Software Design Documentation for the CATT project
- To update Sequence and UML Diagrams for the CATT project
- To develop, document and test software for the CATT project
- To provide software development support to the CAIDA team
- To provide software testing support to the CAIDA team

To accomplish the objectives in this internship, some tools needed to be utilized to complete the tasks efficiently and effectively. The tools used are as follows:

- Specialized Integrated Development Environment (IDE) used to develop software, test software and maintain connection with the servers.
- Hosted hypervisor where the CATT is run in an isolated virtual machine.
- Version Control software used to maintain and review code for the CATT project.
- UML design software used to create and update CATT diagrams.

II. Methodologies

A. Agile Development

During the internship, the agile methodology was used for its flexibility and adaptability to accomplish tasks in a reasonable amount of time. While no specific branch of agile was used (i.e. SCRUM), much of the work performed followed the fundamental ideologies. This gave the ability to break down tasks into fully independent features that could be completed and tested in a reasonable amount of time. Another reason for the agile methodology was to allow a feedback based development with the mentor. This allowed for an iterative implementation where the software would be planned, designed, developed, tested and finally peer reviewed, as well as reviewed by the mentor. This sprint cycle would be iterated through each project, providing a fast development cycle with constructive feedback backing up the final product. Additionally, development plans allowed flaws and issues to be caught early before major design changes. This led to a minimized number of errors and fewer changes needing to be made in the final stages of development.

B. Documentation

Before the internship, two independent telemetry tools were built to generate and modify telemetry streams. As the internship started, the two tools were combined into a single project called CATT. The first goal was to ensure the documentation of these tools were combined to reflect the new design of CATT. As the initial documents had passed a 90% review, merging them did not require much rewording. This allowed the final document to quickly pass reviews with minor edits and changes being needed to fit the given list of requirements. Besides the wording, the images and diagrams also needed to be updated as the code layout had changed. The sequence diagrams that were used to describe how commands flow through the program were partly designed using PowerPoint. These diagrams were then transitioned into the UML design software to improve reusability and readability. Additionally, CATT commands that were not defined on the existing diagrams were also detailed. Along with the sequence diagrams, the UML designs were also updated for the class structures. While the code for the telemetry engines had remained mostly unchanged, it was the connections between them that had been updated to incorporate their functionality. This made for several UML tweaks and the addition of new class diagrams to describe these operations. Once completed, work was shifted towards the CAIDA developer's guide.

With the CAIDA team's progress, there was additional information that needed to be covered for those new to the team and projects. One of the documentation goals was to write a development guide that can cover all information needed for team members to get up and running with CATT. Topics covered by this guide included proper Python style guidelines, installation instructions for software and tools, and guides on getting set up to begin coding. While initial construction has begun, with several iterations completed, a finalized version will not be released until early December 2018.

Lastly, the current commenting structure used throughout the CATT project was redesigned. Previously there was not a unified style to describe the code structures across the classes. To ensure readability without obtrusiveness, the Napoleon commenting style was adapted for docstrings. This allowed the team to remove scattered comments and move vital information to the beginning of classes in a readable format. With a consistent docstring style, automated documentation tools can be used to their full potential to reduce the amount of manual work needed when searching for key sections of code.

C. Software Development

The main tasks assigned during the internship involved developing software for the CATT project that could satisfy the software requirements written by CAIDA. The software development was done completely in Python, using a Unix-based main operating system. To make software improvements and create new features, the IDE was used as the editor to provide a reliable and consistent work environment. Additionally, version control software was used to manage the different changes being incorporated to the project.

Throughout the internship, there were three key software-based projects that Antonio and Taylor were assigned:

- Noise/Mask feature
- Multi-Measurement Sequencer feature
- Inspection engine

The noise/mask feature was developed to improve CATT's emulated data production. The way the feature behaves is by taking the generated emulated data and performing an exclusive disjunction between the emulated data and an error pattern provided by the user. The error pattern should be provided from the difference in expected results to make the output as realistic as possible. The development of this feature adds effectiveness to testing for the CAIDA customers who use the tool to emulate the hardware by giving them a more precise way to understand how their tests behave under different circumstances and scenarios that can be provided by the user.

The multi-measurement sequencer feature is a series of updates to improve the capabilities of the telemetry modification tool within CATT. The first improvement is for the data overwrite functionality that is applied to streams of telemetry data. Previously, just the generation engine could overwrite strips of data since this was easy to implement - it had access to all the data being produced. If this was directly implemented into the modification engine, only a single segment of data would be modified rather than any location where the specified data showed up in the telemetry stream. Now that the modification engine has a custom overwrite functionality built in, a user can overwrite sections of data being streamed that will constantly apply to all new data flowing through it. With this implemented, teams performing tests can now easily modify entire sections of data to test certain scenarios and easily revert back to the original data.

The inspection engine is a new tool developed for CATT whose goal is to inspect specified data at any point in a telemetry stream. It was originally branched off of the modification engine code base, which was modified to analyze data points. Now that major updates are being done to CATT, the inspection engine needs to be brought online so it can now be upgraded as the main CATT tool is updated. In addition, new features to expand the limits of what data can be verified will be implemented. This will allow the team to implement automated tests for most of CATT's features.

D. Testing

Testing software is a core aspect of the agile process where the software developers make sure the product is working as designed. The software is put through the different scenarios that could potentially occur during the product's use, ensuring the software is robust and as error-free as possible.

This internship required that the selected individuals would perform different kinds of testing on the software. The testing methods performed on the project were the following:

- Unit testing
- Integration testing
- Functional testing
- Verification & Validation testing

III. Results

Through the work performed during the internship, various tasks were completed and delivered as requested. These tasks provided a positive outcome for the CATT project overall. The work consisted primarily of documentation, software development, and testing.

The first major document finished was the Software Design Document, which has continued into 90% review for final evaluations. It is the culmination of both telemetry engine designs, which were combined into one design document to be used with further versions of the CATT project. This document also includes a new series of sequence diagrams and UML diagrams that more thoroughly describe the code and how it works. The other major documentation project involves the developer's guide document, which is still in its initial rough draft stage. Most of the components have been outlined and described, but it still requires several more iterative reviews before it can be used by the entire CAIDA team.

From the noise/mask generation feature, a new functionality was created with the capability to produce noise over the data being transmitted from CATT. The implementation of this feature required a new method that could take the data to be transmitted as the input, pass it to a specialized method that could parse and apply the mask over the data payload, and return the newly generated set of data. This new payload would then override the data package for transmission and continue with the CATT process.

Additionally, for this new feature to work, parts of the CATT project were restructured to adhere to the new functionality the project would produce. The major change in the CATT source code was to recreate a set of functions that were used to parse data from files that could be used to simulate specific environments and scenarios. These functions were modified and restructured to create a new parent class that had the main aspects of acquiring and parsing the scenario file. Two child classes were created to perform specific capabilities required by the different modules in the project. The addition of this feature provides users a way to produce more realistic data from the CATT and a way to test software more effectively and obtain more useful results from the use of the feature.

With the work done on the multi-measurement sequencer, new functionalities have been added to the telemetry modification tool. The first big update is allowing the modifier to have access to overwrite bytes of information in the telemetry streams. If a tester needs to change the values in a section of data, they can modify them on the fly by calling this new overwrite command. However, a tester may want to remove previous changes so a cancel operation was added as well to return the data back to its normal state.

On the testing phase of the project, the different types of tests mentioned in the methodology were performed on the software. From these tests, bugs were found and problems in the code were fixed that could otherwise compromise the outcome from the project.

For the noise feature, unit tests were developed to test each new piece of software and the modifications that accompanied the new functionality. The tests were able to find that the method that performed the override over the payload was actually overflowing and overriding some data from the data packet outside the payload. This would have gone unnoticed if the tests were not performed, because the data packets are very long and difficult to read in the span of time that they are produced. The tests discovered the bugs that could have potentially caused errors and inconsistencies when testing, which means that it would delay development of software using the CATT.

On the multi-measurement sequencer, testing was performed through unit tests. The unit tests uncovered some issues with indentations, redundancies and unchecked cases. The issues were found to propagate throughout the software because some of the cases require a specific return, which was given incorrectly or not given at all. These errors in the feature were fixed and the ones found in the CATT were brought up to the supervisors of the project, so that these issues could be fixed promptly, to avoid further development with unstable code.

Furthermore, in the testing phase of the internship, a new task was assigned which required verifying and validating old unit tests that had been developed for the current production version of the project. These tests were found to be inadequate for the next version of CATT because most would not pass due to changes done to the source code and improvements made to CATT's reliability. After the tests were found to not work, they had to be modified and fixed to adapt to the new version of the project. These tests consisted in evaluating the general functionalities of the CATT project, so getting them operational was of great importance.

IV. Conclusion

The work performed positively impacted the CAIDA team as many projects were finished ahead of schedule. With updates being pushed out, the CAIDA clients will have further additional features available to more accurately fit their needs. Along with that, the documentation work will vastly improve the time it takes for new members to familiarize themselves with the project. Following this, there are several more tasks planned to take place.

The Inspection Engine was a tool under development during the summer by a previous intern. With new automated tests, the inspection engine needs to be brought back online to the latest version of code. As most of the system has been developed, the code transition will be fairly quick. The goal would be to add additional features, giving it improved dexterity when inspecting incoming telemetry streams. Another task would be continuing development of the multi-measurement sequencer. There are roughly twelve more features needed for its completion so these could be implemented in the future.

From the work we performed, we obtained a solid grasp of the entire development cycle from documentation to testing. We were able to follow our projects through the process as they were beginning to be implemented into the final code base. Not only were we able to finish the initial objectives, but also we were able to go above and beyond to complete future tasks needing to be done.

V. Acknowledgments

We would like to express our deepest thanks to our mentors, Walter Wehner and Jill Giles, for the incredible assistance they provided, their guidance, and the accomplishments they helped us achieve. Special thanks go out to our supervisors Dean Orr and Glenn Perez for giving us the privilege to work as part of the System Avionics branch, and always ensuring we were comfortable with our work. We are also appreciative of those on the team that helped us through the process of assimilating to the branch such as Curtis Williams, Matt Roscoe, Haroon Khan, Matt Harper, and Myphi Tran.

Additionally, thanks are given to Jill Giles and Jamie Szafran for all their help and guidance through the internship process. Thanks also goes out to the Education Office on getting everything set up for an amazing internship experience. We extend our gratitude to USRA and NASA for being able to provide these opportunities through sponsoring our internship here at the Kennedy Space Center.

VI. References

[1]Ilieva, S., et al. “Analyses of an Agile Methodology Implementation.” *Proceedings. 30th Euromicro Conference, 2004.*, 2004, doi:10.1109/eurmic.2004.1333387.

[2]Kleijnen, Jack P.C. “Verification and Validation of Simulation Models.” *European Journal of Operational Research*, vol. 82, no. 1, 1995, pp. 145–162., doi:10.1016/0377-2217(94)00016-6.

[3]United States, Congress, Department of Defense, et al. “Agile Software Development.” *Agile Software Development*, Data and Analysis Center for Software, 2003, pp. 1–63.