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

Taylor Walter Thomas

Major: Applied Math and Computer Science; Information Communication Technologies

NASA Kennedy Space Center

2019 Spring Session

Date: 11 APR 2019

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

Taylor Walter Thomas[1]
John F. Kennedy Space Center, Kennedy Space Center, FL, 32899

Abstract

The Customer Avionics Interface Development and Analysis (CAIDA) team provides modeling and simulation software for the verification of the Launch Control System (LCS). In late 2015, CAIDA began development of the Data Exchange Message (DEM) Generator (DEMGen) and development of the DEM Modifier (DEMMod) in early 2017. DEMGen is able to mimic the telemetry streams normally sent by different simulators allowing increased user control and more telemetry instances. DEMMod takes in a telemetry stream and allows the modification of entire packets or simple measurement values. Together, these tools provide the capabilities to simulate and test complex launch scenarios to ensure LCS is fully prepared for any anomalous behavior during an actual launch. In early 2018, CAIDA began a new project called the CAIDA Advanced Telemetry Tool (CATT), which combined the code and documentation of DEMGen and DEMMod. CATT now allows the integration of new tools into a singular program providing compact and easy access. In the fall of 2018, the author worked with another intern, Antonio Negrón, to develop several new CATT features along with proper documentation and unit testing. In the following spring, the author continued that work by integrating the features into CATT, solidifying the documentation, and expanding the testing to cover more cases. The outcome of both semesters includes an inspection engine for automated packet analysis, a packet-masking feature, and a multi-measurement sequencing feature.

Nomenclature

ADTE	=	Avionics Development and Test Environment
CAIDA	=	Customer Avionics Interface Development and Analysis
CATT	=	CAIDA Advanced Telemetry Tool
DEM	=	Data Exchange Message
DEMGen	=	DEM Generator
DEMMod	=	DEM Modifier
GFAST	=	Ground Flight Application Software Team
ICPS	=	Interim Cryogenic Propulsion Stage
IDE	=	Integrated Development Environment
KATS	=	Kennedy Avionics Test Set
KSC	=	Kennedy Space Center
LCC	=	Launch Control Center
LCS	=	Launch Control System
LX	=	Exploration Ground Systems Directorate

[1] Spring Intern, Avionics Branch, NASA Kennedy Space Center, University of Wisconsin – Stout.

MPCV	=	Multi-Purpose Crew Vehicle
MSFC	=	Marshall Space Flight Center
NASA	=	National Aeronautics and Space Administration
NE	=	Engineering Directorate
O&C	=	Operations and Checkout
OPR	=	Office of Primary Responsibility
PCM	=	Pulse Code Modulation
PCMMU	=	PCM Master Unit
SHADE	=	SLS High fidelity All Digital Emulator
SLS	=	Space Launch System
SOCRRATES	=	Software-Only CEV Risk Reduction Analysis and Test Engineering Simulator
SSPF	=	Space Station Processing Facility
TOSC	=	Test and Operations Support Contract
ULA	=	United Launch Alliance
UML	=	Unified Modeling Language
USRA	=	University Space Research Association
VMC	=	Vehicle Management Computer

I. Introduction

During the fall of 2018, Antonio Negrón and Taylor Thomas interned at the National Aeronautics and Space Administrations (NASA) Kennedy Space Center (KSC). They joined the Customer Avionics Interface Design and Analysis (CAIDA) team located in the Exploration Avionics Branch. Dean Orr was the project supervisor of both interns in the fall and Taylor's supervisor in the spring; Jill Giles, from the Software Branch, was Taylor's mentor in the fall; Walter Wehner was Antonio's mentor during the fall and Taylor's mentor during the spring. The work performed by Antonio and Taylor led to the further enhancement of the CAIDA Advanced Telemetry Tool (CATT), which serves as CAIDA's new primary utility.

II. History of CAIDA

In 2011, Lockheed was constructing the Orion capsule in the Operations and Checkout (O&C) building at KSC, while NASA was building the Space Launch System (SLS) vehicle at the Marshall Space Flight Center (MSFC). Arnold Postell, the Avionics Division Chief at KSC, foresaw a need to start developing flight simulators for the Orion spacecraft and the SLS. With Orion and SLS unavailable for on-location testing, the emulators would provide KSC engineers a means of preparing for the development and testing of ground systems hardware and software. This would reduce the time required for checkout at the different flight centers.

The original plan was to have physical hardware flight emulators. Although these would most accurately emulate the processing and timing of flight hardware, limited funding would ultimately require a software-only solution for flight emulation. The vision for CAIDA would be to provide a capability for customer avionics platforms to interface with multiple systems whether that be the Launch Control System (LCS), agency centers, or vendor labs and facilities. It would also include the addition of the Avionics Development and Test Environment (ADTE) lab, which was reserved for avionics related research, small project development for undefined missions, hardware and software prototyping, and hands-on learning for new engineers. These labs would help remove the reliance on simulator and emulator deliverables, thus saving significantly on equipment travel costs and reducing technical risks for the development of the LCS.

The team started the CAIDA lab within the Space Station Processing Facility (SSPF) (Figure 1). It started as an engineering lab under the Engineering Directorate (NE) and drew inspiration from the Kennedy Avionics Test Set (KATS), which fed Pulse Code Modulation (PCM) telemetry from the Orbiter into a PCM Master Unit (PCMMU) effectively providing emulation. Through the funding CAIDA received, the team was able to acquire a Test Bench with the Orion Vehicle Management Computer (VMC), which would become the foundation for CAIDA's testing equipment.

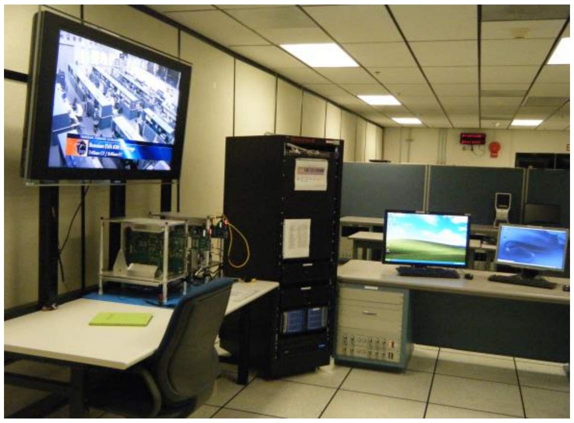
Figure 1 - CAIDA Lab in SSPF with Honeywell Test Bench and Orion VMC

When CAIDA's size and capabilities started expanding, the team moved the lab into the Launch Control Center (LCC) where it resides to this day. Over the years, CAIDA used the Test Bench along with the Software-Only CEV Risk Reduction Analysis and Test Engineering Simulator (SOCRRATES) by Lockheed to decommutate and send telemetry to the LCS.

At that time, CAIDA was fully capable of supporting performance testing for the Orion Multi-Purpose Crew Vehicle (MPCV) Gateway; however, MSFC noted that CAIDA would not be able to meet the SLS Gateway's requirements set by the SLS High fidelity All Digital Emulators (SHADE). This was resolved through the development of a nine-line script that would eventually become a manual telemetry-testing tool and form the future of CAIDA's software development.

In early 2016, CAIDA's new tool was growing increasingly popular among testers, which required the team to spend more time manually operating the tool. Soon, the small team became overwhelmed with requests and needed a solution to alleviate this work. The developers decided the best course of action was to automate the tool, and in March 2016, CAIDA released the Data Exchange Message (DEM) Generator (DEMGen). With DEMGen, testers could now generate streams of telemetry normally produced by the SLS and MPCV simulators. These telemetry streams contained packets of information, called DEMs, which consisted of measurement readings needed for SLS such as pressure and temperature. Now, testers had full control of the telemetry, greatly improving the number of concurrent telemetry instances available to them. With this change, CAIDA had become an official subsystem of the Exploration Ground Systems Directorate (LX) and thus came changes to CAIDA's structure.

As the CAIDA developers improved DEMGen over the next several years, the need arose for integrated simulations. CAIDA needed physics models provided by the emulators in addition to the ability to override measurement values. The developers proposed the idea to build a unified code base to include these new features into DEMGen, but due to the time sensitivity of the requested delivery, the developers released a standalone product in April 2017 called the DEM Modifier (DEMMod). This tool pulls in DEMGen's generated telemetry streams, and then modifies the individual packets or measurement values in real-time. With both tools, testers could now design and test SLS launch scenarios, ensuring the LCS is fully prepared for any anomalous behavior that may occur during a launch. In addition, these tools became highly utilized by groups such as the Ground Flight Application Software Team (GFAST) to test and verify the vehicle and spacecraft processing requirements of the LCS.

Near the start of 2018, LX requested that CAIDA provide a similar tool to DEMGen that would support generating telemetry streams for the United Launch Alliances (ULA) Interim Cryogenic Propulsion Stage (ICPS) simulator. With this, the case for a single, combined tool for ease of maintenance and expandability was apparent. CAIDA then delivered a multi-segment plan to develop this tool within the timeframe needed. In June 2018, the plan was accepted and development began on the new CATT project.

III. Objectives

In the fall of 2018, Antonio and Taylor joined CAIDA as interns and assisted in preparing CATT for its 2.0 engineering release, scheduled for that November. In addition, they assisted with the development of several new features for the 2.2 release. CATT 2.0, composed of both DEMGen and DEMMod, would server as the base CATT release. In March of 2019, CATT 2.1 would release with the addition of the ICPS telemetry generation tool. During spring of 2019, the features developed by Antonio and Taylor were integrated into CATT 2.2 while CATT 2.3's goal was to streamline code and to implement common features across the tools CATT contained.

In the fall of 2018, several tasks were assigned to Antonio and Taylor. These tasks included:

- Creating documentation for CATT by merging existing DEMGen and DEMMod design documentation
- Writing a generic developer's guide for future interns and employees
- Updating sequence diagrams and Unified Modeling Language (UML) diagrams
- Developing the noise-masking feature
- Developing the multi-measurement sequencing feature
- Providing basic unit and functional tests for developed features
- Writing initial code documentation for developed features
- Rebuilding the telemetry inspection engine

In the spring of 2019, Taylor continued work on CATT. His tasks included:

- Updating the aforementioned features to the latest engineering release of CATT
- Writing integration tests and addressing more edge cases for each feature
- Integrating each feature into the full release of CATT to prevent code depreciation
- Writing official design documentation for each feature and design new diagrams
- Continuing development of the aforementioned developer's guide for future interns and employees

Specific tools and software were required to accomplish Antonio and Taylor's objectives effectively and efficiently. The equipment needed includes:

- Integrated Development Environment (IDE) to develop and test software while maintaining a connection with the servers
- Hypervisor to host CATT on an isolated virtual machine
- Version control software to maintain and review code for CATT
- UML design software to create and update CATT diagrams

IV. Methodologies

A. Agile Development

During Antonio and Taylor's internships, CAIDA was using the agile methodology. Agile's flexibility and adaptability enabled task completion in a reasonable amount of time. While no specific framework was used, the structure of the work performed followed fundamental agile ideologies. By using agile, it was possible to break down tasks into fully independent features that could be more easily tested and implemented. Agile also allowed a feedback-based development environment, which facilitated an iterative software development lifecycle. This was broken down into several peer-reviewed steps including planning, designing, developing, and testing. Each project was composed of multiple sprints during development, providing recurring, constructive feedback for each feature. This iterative process increased the likelihood of eliminating errors and addressing issues early on in the development cycle, which lead to minimal errors and less last-minute changes.

B. Documentation

In the fall of 2018, the first engineering release of CATT was still underway with one of the larger tasks being a new software design document. Since CATT will have the capabilities of both DEMGen and DEMMod, their respective design documents can provide the foundation. With DEMGen and DEMMod documentation passing a 90% design review previously, merging them together to form CATT's design document required minimal rewording. The final document then only needed minor editing and changes to fit the new list of requirements for the upcoming design review. In addition to the wording, the diagrams also required updates as merging the code had changed the basic layout of the software. Originally, previous interns constructed the sequence and UML diagrams using PowerPoint, so Antonio and Taylor transferred them to a new tool allowing more design options and faster delivery of the new changes. They implemented several aesthetic improvements to the sequence diagrams while constructing several new ones for unique CATT commands that did not exist previously. The UML diagrams experienced the most changes as CATT contained new functions and structures to wrap DEMGen and DEMMod into this single tool. In addition, several images of how the telemetry engines connected required remapping. After the CATT design document was finished, CAIDA sent it through several design reviews. In the meantime, this allowed Antonio and Taylor to begin work on the CAIDA developer's guide.

With the CAIDA team's progress, there was a need for several documents to help new team members to get up to speed with the project. The first document was an introductory guide that would cover everything needed to begin developing software for the team. Topics covered include a CATT overview, installation guides for the software and tools, and examples on how to connect and use the CAIDA equipment. The second document explained coding guidelines to improve and optimize anything written for CATT. This ensured code would be more succinct and readable by the team as a whole. The third document listed dozens of resources on tutorials and notes for those who may be newer to the language used. Finally, the fourth document contained software libraries found through the writing of the previous three documents. The document further explains how these libraries might assist with testing or improving efficiencies if utilized in CATT. While Antonio and Taylor built these four documents in the fall, they will remain as a consistently updated resource as more team members add and improve them. In the spring of 2019, Taylor added an additional guide solely for interns, which outlines information to assist them while on the CAIDA team and while at their internship. The goal would be to pass the document from intern to intern, recording new information about the position to better help interns in their first few weeks of their internship.

When CAIDA was originally developing CATT, there was no succinct commenting structure for the software. To prepare for the engineering release in November, Antonio and Taylor applied the approved commenting style to the functions and classes, providing more details into the overall code functionality. This also allowed the team to remove scattered in-line comments and move important information to the beginning of the classes in a readable format. By December, the developers implemented an automated documentation tool to generate comments for created functions and classes.

In the spring, Taylor expanded on the documentation of the features built in the fall to fit the requirements needed for the new version of the CATT design document. With CATT now including software to generate ICPS telemetry streams, it changed how the fundamentals of the code operated. This meant that both the sequence and UML diagrams needed reworking and relabeling. Taylor also built several internal wiki guides to give general explanations of how each feature worked, with a focus on how the commands traversed the code.

C. Software Development

The primary tasks assigned during the fall internship, involved developing software for CATT that could satisfy the CAIDA-specified requirements. Antonio and Taylor used a Unix-based operating system on virtualized hardware to write the source and tests. All CAIDA developers used the same IDE to provide a consistent work environment. This assisted in resolving any occurrences of IDE specific issues. Additionally, the team used version control software to compartmentalize development, handle merging of complete features, and provide a means of retrieving older versions of the software.

In the fall, Walter assigned Antonio and Taylor three software-based projects. These were the:

1. Noise-masking feature
2. Multi-measurement sequencing feature
3. Telemetry inspection engine tool

In the spring, Taylor finalized these features and tools before fully integrating the projects into CATT.

The noise-masking feature allows DEMMod to stream telemetry that matches realistic measurement conditions. For example, some measurements, when obtained by DEMMod, appear to have a constant value. However, there could exist a seemingly insignificant decimal place in the measurement that rapidly fluctuates between several values. The new feature would now factor in that fractional quantity when outputting the telemetry stream, better simulating realistic LCS conditions. The feature is a modification to DEMMod that allows it to perform an exclusive disjunction between a user-provided generated telemetry stream and error pattern. The error pattern would contain small, fractional differences, which allows the modified stream to have small deviations in the appropriate values. The development of this feature makes testing more effective by allowing CAIDA customers a more precise understanding of how their tests behave under different scenarios.

The multi-measurement sequencing feature is made of a series of updates to improve the capabilities of the DEMMod within CATT. CAIDA only required the first few updates to be finished by the CATT 2.2 release date. The first update adds the ability to overwrite entire segments of packets from incoming streams of telemetry data by checking for a user specified packet ID. Previously, only the DEMGen within CATT could overwrite large segments of packets. Since DEMGen was generating the telemetry, it implicitly had direct control of the data. However, DEMMod is constantly receiving new packets of data, so the overwrite function would only apply to a single packet that matched the specified ID, and not every packet that came afterwards. This new feature would store the user’s overwrite command in a list, and continually run that command when a matching packet streams through DEMMod. With this implementation, users can now easily modify entire packet segments using DEMMod and quickly revert changes by removing commands from the user command list. The second update gave DEMMod the ability to load in scenario files, which would change the default starting values to a different configuration. This update became obsolete by the aforementioned noise-masking feature whose scenario loading capability became far superior. The third update allowed a user to increment or decrement measurement patterns found in certain packets, which only required minimal changes to implement.

The telemetry inspection engine is a tool currently in development for CATT that inspects and analyzes specific packets or segments of packets in any given telemetry stream. CAIDA originally built the tool as a standalone extension to DEMMod’s code base, independent of CATT. With CATT requiring more testing, and with the inspection engine depreciating, CAIDA decided to implement this tool into CATT. The tool would assist in those remaining CATT tests, especially since developers will be adding new and distinct data types to the software. The ultimate goal for the inspection engine would be to assist the team in automating tests for CATT features.

Although Antonio and Taylor developed the three projects according to the outlined requirements in the fall of 2018, CAIDA did not integrate the new features until after the CATT 2.1 release. During the spring of 2019, Taylor updated the source code of the projects from the fall to match the style conventions of the 2.1 release. The original functionality in the projects remained the same for the most part, but most files needed to point to new locations in 2.1, as CAIDA developers and refactored and modified most classes and functions. In addition, since the ICPS telemetry generator was a new tool within CATT, Taylor needed to ensure the new features would not create new issues with this tool.

D. Testing

Testing software is a core component of the agile process in which software developers ensure that their product works as designed. Developers present different scenarios and possible use cases that could potentially occur during the product’s use to ensure that it is as robust and error-free as possible.

These internships required the interns to perform different testing on CATT. Some of the testing methods needed included:

- Unit testing

- Integration testing
- Functional testing
- Verification and Validation testing

During the fall semester, Antonio and Taylor focused on unit testing. By checking each feature for nominal and edge cases, they were able to ensure the programs worked as expected which eased the process for feature integration. There were initially about 10 unit tests developed for each feature. Since Antonio and Taylor developed the inspection engine towards the end of the fall semester, CAIDA will not build unit tests until after full integration. The tests used a language specific testing library that aids the creation and manipulation of small, individual tests. To ensure all three projects were operating as intended, Antonio and Taylor continued testing by writing several functional tests.

Before the spring internship started, the structure for building tests had changed, so Taylor worked on modifying the aforementioned tests when he returned in the spring. Taylor also wrote 10 integration tests for each of the three features to verify their functionality in the context of the CATT 2.1 release. The developers will leave the functional tests written in the fall as is until the CATT 2.2 release is further along in the development cycle.

V. Results

Through the work performed during both internships, Taylor and Antonio successfully completed their tasks and projects and delivered them as requested. Overall, their work provided a positive outcome for the CAIDA team and the continued development of CATT. The tasks and projects worked on consisted of documentation, testing, and software development.

The first documentation project finished was the CATT Software Design Document, which passed through the 90% design review. Antonio and Taylor combined the technical documents from DEMGen and DEMMod into a singular design for CATT. The new document included new and updated sequence and UML diagrams that more accurately described the CATT code. The second documentation project was the developer's guide, which was started early into the fall internship. It is still in the rough draft phase and developers will improve on it as time goes on. In addition, Taylor developed an intern specific guide in the spring, which is in a similar state. Both of these will require several more iterative reviews as there may still be continuity and grammar issues.

From the noise-masking feature came the functionality to apply user-generated noise over data transmitted from DEMMod within CATT. The implementation of this feature required a new method that could use streamed data as its input, pass it to a specialized method that would parse and apply the noise mask over the data payload, and then return the newly generated set of data. This new masked-payload would serve as the data package for transmission. The largest change in the source code was recreating a set of functions users used to parse data from files to simulate different scenarios. These functions were now contained within a new parent class that had the main aspects of acquiring and parsing the scenarios. The two child classes created performed specific capabilities required by the generation and modification tools. The addition of noise-masking provided users a way to produce realistic measurement data from CATT and obtain more accurate and useful results for their respective hardware and software.

The multi-measurement sequencing feature introduced a new set of updates to the DEMMod tool within CATT. The largest update involved duplicating the overwrite capability from DEMGen, and putting it in DEMMod. Once the code was in place, Taylor and Antonio made changes that allowed the overwrite commands to be accumulated in a list. Then, as DEMMod receives packets, DEMMod would check the list and see if any incoming packets required a data overwrite. In addition, a feature to remove overwrites from the list was also created in case the user no longer needed to edit certain packets. Afterwards, Taylor and Antonio made several other slight modifications such as tweaking how the scenario parsing worked.

For the telemetry inspection engine, Taylor and Antonio were able to salvage some of the original code, and thus most of the work performed centered on making the tool operational while adding additional improvements. After slowly merging six months of changes into the tool, the inspection engine could now operate within CATT. Its initial capabilities included watching and reporting information about specific values within certain packets. During the last few weeks of the internship, Taylor and Antonio added the ability to watch for entire packets. This allowed a user to gain information on the rate of a packet, as well as how often the packet was becoming stale. To check for a stale packet, the user would specify the maximum time it should take for a packet with the same unique ID to show up. After adding these packet IDs to a watch list, the inspection engine would record every time a packet arrival exceeds this maximum time. Then, by sending another command, the user can see how many times packets with that unique ID became stale within a given amount of time. For example, one might look for packets with the ID of 1234 and set the maximum stale time to 100 milliseconds. After several minutes, the user could execute the command to check for stale packets and see that within the past several minutes, packets with the ID 1234 have become stale numerous times. This feature will better allow a user to help troubleshoot possible connection issues, and give future developers a branching off point to add additional information and abilities to this tool.

As for the testing phase of these projects, Taylor and Antonio performed the different types of tests mentioned in the methodology on the software. From the test results, they were able to fix issues found in the code that might otherwise compromise the outcome of each project. Due to the time constraints of the internship, Taylor and Antonio were unable to develop unit tests for the inspection engine; however, the developers plan to build the unit tests when CATT receives the inspection engine.

For the noise-masking feature, unit tests were able to find that the method performing the payload override for a packet was actually overflowing and overriding other data within that packet. Without these tests, the issue would have gone unnoticed as the project progressed and could have caused a delay in the development of CATT. It is difficult to notice these issues off hand as the data packets are typically long and hard to read in the span of time that DEMMod produces them.

On the multi-measurement sequencing feature, unit tests uncovered several pre-existing issues with indentations, redundancies, and unchecked cases. Some of the cases were expecting very specific returns, but the unit tests received incorrect data or no data at all, which helped find issues that propagated throughout the software. Taylor and Antonio fixed the errors found in the multi-measurement feature, while they brought up the errors found in CATT itself to the supervisors of the project. This allowed the developers to fix those issues promptly to avoid further development with possibly unstable code.

In addition to testing the new features, CAIDA tasked Taylor and Antonio with verifying and validating old unit tests. CAIDA found these tests to be inadequate for the CATT 2.0 engineering release as developers programmed them before DEMGen and DEMMod merged. After showing the tests did not work, Taylor and Antonio modified and fixed these 20 to 30 unit tests for CATT. The tests now consisted of evaluating the general functionalities of the CATT project, so getting them operational was of great importance. While more tests were still required, this served as a solid unit-testing base for the next set of engineering releases.

In the following spring, Taylor continued the testing needed to ensure the features would remain stable as he slowly integrated them into the CATT 2.2 release. Taylor developed new integration tests based on an improved, generic testing format that developers had come up with, allowing easy duplication and review. Initially it took some trial and error, as integration testing had not been done by the developers yet for DEMGen and DEMMod, so ensuring the tests were operating successfully was a challenge. After the integration tests for the multi-measurement sequencing feature passed, Taylor was able to duplicate the tests for the remaining features reducing overall development time. With the integration-testing baseline established, testing future features for DEMGen and DEMMod will be a much smoother process. As time permits, Taylor will continue the development of specific unit tests and apply the similar format that was established. Since unit testing for specific features will not be a huge concern until the next engineering release, the incoming summer intern will likely complete the unit tests.

VI. Conclusion

Overall, the work performed during the fall and spring internships positively affected the CAIDA team as many projects were finished on time, with a few being ahead of schedule. With Taylor and Antonio finishing these projects, the CAIDA clients will now have these features available to better suit their needs. In addition, the documentation completed will greatly improve the time required for new team members and interns to familiarize themselves with CATT. Following this spring internship, CAIDA has several tasks planned for the incoming interns.

While Taylor and Antonio brought the inspection engine back online with new features to improve its effectiveness, several more updates remain. Since this tool is in the early stages of development, it needs more work done before CAIDA is able to utilize the tool for automation purposes. In addition, the multi-measurement sequencing feature will have additional components that will need to be developed. CAIDA planned for it to have twelve features in total, with Taylor and Antonio completing the first several. As CATT development proceeds, the new interns will develop and integrate these next set of features.

From our work on this project, we obtained a solid grasp of the entire development cycle from documentation to testing. We were able to follow our projects through the process of development as we integrated them into the latest engineering release. It was a rewarding feeling to know that we completed all of our expected objectives, while also going beyond that to begin planned tasks for the future as needed.

VII. Acknowledgments

I would like to express my deepest thanks to my mentors during both internships, Walter Wehner and Jill Giles, for the incredible assistance they provided, their guidance, and the accomplishments they helped me achieve. Special thanks goes out to my supervisors Dean Orr and Glenn Perez for giving me the amazing privilege to work under the Exploration Avionics Branch, and by always ensuring I was comfortable with my work. I am also appreciative of those on the CAIDA team that helped me through the process of acclimating to the team and provided insight into the incredible history we are building upon which includes Curtis Williams, Kevin Grant, Tyler Ensey, and Myphi Tran. I could not have been able to tell the amazing journey of how CAIDA came to be without the knowledge and memories that Arnold Postell and Brian Luther shared. They helped me tell a story that will live on as the team continues to grow and reach new goals. In addition, I am grateful for the Test and Operations Support Contract (TOSC) team members Mat Roscoe, Haroon Khan, and Matt Harper who helped me hone my programming talents and showed me how to develop better code.

Additionally, I thank Jill Giles and Jaime Szafran for all their work in guiding me through the internship process. Thanks goes out to the Education Office, including Gwen Gamble and Kathleen Wilcox, for laying the groundwork for this internship to run as smoothly as it did. I also extend my gratitude towards the University Space Research Association (USRA) and the National Aeronautics and Space Administration (NASA) for being able to provide these opportunities and experiences by sponsoring my internship here at KSC.

VIII. 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.

[4]Wehner, W., "CAIDA Software User Guide" K0000328602-GEN, NASA, KSC, 2019 (unpublished).

[5]Wehner, W., "CAIDA DEMGen Software Design Description" K0000328603-GEN, NASA, KSC, 2018 (unpublished).

[6]Wehner, W., "CAIDA DEMMod Software Design Description" K0000356820-SPC, NASA, KSC, 2018 (unpublished).