Design and Development of Functionally Operative and Visually Appealing Remote Firing Room Displays

Kristy Quaranto
NASA Kennedy Space Center
Major: B.S. Aerospace Engineering
KSC FO: Spring Session
April 9th, 2014

Table of Contents

I. Abstract 2
II. Introduction 3
III. Hypergolic Subsystem 4
IV. Training 4
V. Software Lifecycle 4
VI. Display Editor 5
VII. Completed Tasks 7
VIII. Beneficial Exposure 9
IX. Conclusion 9
X. Appendix A – Acronyms and Abbreviations 9
XI. Acknowledgements 10
XII. References 10

Design and Development of Functionally Operative and Visually Appealing Remote Firing Room Displays

Kristy Quaranto[1]
Embry-Riddle Aeronautical University, Daytona Beach, FL 32114
Kennedy Space Center, FL 32899

I. Abstract

This internship provided an opportunity for an intern to work with NASA's Ground Support Equipment (GSE) for the Spaceport Command and Control System (SCCS) at Kennedy Space Center as a remote display developer, under NASA mentor Kurt Leucht. The main focus was on creating remote displays for the hypergolic and high pressure helium subsystem team to help control the filling of the respective tanks.

As a remote display developer for the GSE hypergolic and high pressure helium subsystem team the intern was responsible for creating and testing graphical remote displays to be used in the Launch Control Center (LCC) on the Firing Room's computer monitors. To become more familiar with the subsystem, the individual attended multiple project meetings and acquired their specific requirements regarding what needed to be included in the remote displays. After receiving the requirements, the next step was to create a display that had both visual appeal and logical order using the Display Editor, on the Virtual Machine (VM). In doing so, all Compact Unique Identifiers (CUI), which are associated with specific components within the subsystem, will need to be included in each respective display for the system to run properly. Then, once the display was created it needed to be tested to ensure that the display runs as intended by using the Test Driver, also found on the VM. This Test Driver is a specific application that checks to make sure all the CUIs in the display are running properly and returning the correct form of information. After creating and locally testing the display it will need to go through further testing and evaluation before deemed suitable for actual use.

By the end of the semester long experience at NASA's Kennedy Space Center, the individual should have gained great knowledge and experience in various areas of display development and testing. They were able to demonstrate this new knowledge obtained by creating multiple successful remote displays that will one day be used by the hypergolic and high pressure helium subsystem team in one of the LCC's firing rooms to fill the new Orion spacecraft.

---

[1] Remote Display Developer for Firing Room Applications Intern, Spaceport Command and Control System, Kennedy Space Center, Embry-Riddle Aeronautical University, Daytona Beach

II. Introduction

With the close of the 30 year long Space Shuttle program, NASA has now committed their efforts to explore beyond Earth's orbit and eventually to Mars. The Orbiters were only equipped to reach low Earth orbit (LEO), this is where the International Space Station (ISS), Hubble Space Telescope, and many other human made satellites reside. The Orbiters' main purpose was to construct the ISS in orbit and, service and deploy the Hubble Space Telescope. Unfortunately, the Orbiter would not be able to withstand the harsh space atmosphere that lies outside of low Earth orbit. Therefore, once the ISS was completed the program would be retired, but NASA would respond with the announcement of their new spacecraft, Orion, that could travel millions of miles away from Earth.

Orion is considered a Multi-Purpose Crew Vehicle (MPCV); visually this capsule looks very similar to that of the capsule used during the Apollo program. However, it is also very different; taking what they knew about the Apollo program and what they have learned through the Shuttle program NASA was able to design the Orion MPCV to be one of the safest spacecraft to date. With the added Launch Abort system, if there was ever a problem with the rocket on the Launchpad and the crew needed to escape quickly, this system would be able to shoot the capsule and the astronauts up and away from the rocket base. The capsule is also encased when stacked on the Space Launch System (SLS) rocket, unlike the Shuttle which was exposed to possible debris collision during launch. Also, the Orion capsule will be able to carry four astronauts unlike the Apollo capsule which could only carry three [Orion Quick]. At Kennedy Space Center, one of the main focus' is to design and develop the Ground Support Equipment (GSE) that will service both the new Orion MPCV capsule and the SLS rocket. Many of the old Shuttle processing and servicing facilities at Kennedy Space Center will be converted to accommodate the new Orion MPCV and SLS rocket.

I accepted an internship opportunity to work with the Spaceport Command and Control System as a remote display developer. During my time at Kennedy Space Center, I developed remote displays to be used in the Firing Rooms to help with the Ground Support Equipment for the Hypergolic Subsystem team for servicing the Orion MPCV. These displays will be controlling potentially hazardous fluids, so it is crucial that the displays be simple and as easy to read as possible for the console operator.

III.Hypergolic Subsystem

I am grateful to have been paired up with my subsystem mentor Joey Parkerson, he took the time out of his busy schedule to educate me on not only the software side of operations but also how the hypergolic subsystem of the Orion Multi-Purpose Crew Vehicle (MPCV) is designed and how hypergolics actually work. For this I am forever grateful, because this new type of knowledge and resources is not provided in a classroom or even public setting. Unfortunately, I cannot share the specifics on Orion's hypergolic subsystem but, he was able to show me schematics and provide me with documents to review that pertained to the specific processes that will be used to service the Orion MPCV's fuel and oxidizer tanks. On the other hand, for those that do not know what the hypergolic subsystem is, it is a system that combines a liquid fuel and a liquid oxidizer which when mixed produces combustion. This type of system does not require a source of ignition, the mixing of the both the oxidizer and the fuel causes the ignition. The hypergolic type of combustion is used for the reaction control thrusters when the spacecraft is finally in space. Next time you watch a space movie or documentary look for the short bursts that help stabilize or move the spacecraft, as seen in Figure 1, which is the hypergolic subsystem firing the reaction control thrusters.

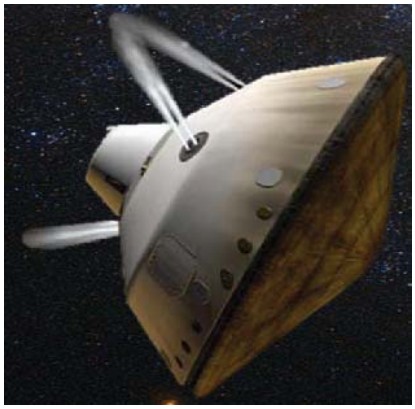

Figure 1: Example of the hypergolic subsystem, this is a computer rendered image of the capsule that housed the Mars Science Laboratory on its trip to Mars [Astronomy].

IV.Training

It took me and my other remote display development team members about three weeks to complete the required software training. The two main software programs that we were trained on were NASA's abstraction layer of a programming language for remote application development and NASA's Display Editor for remote display development. We had both an in-class type of setting along with added self-guided documents and PowerPoints for both software programs. After completing our training classes and documents we were required to submit a skills demonstration for each software program. For the abstraction layer we need to demonstrate a set of code that used the abstraction layers calls, commands, and functions to successfully perform a task with the respective CUI. For the Display Editor, we were required to create a display that had visual and logical order, and it also needed to include all four types of display symbols, which will be described in greater detail in section six.

V. Software Lifecycle

While going through training, one of the first required readings was on the software development lifecycle. I found that the software development life cycle can be a lengthy one,

but it is also very important because it shapes how the system will be developed and implemented. This life cycle is broken up into four main parts, the 30% Design Review, the 60% Design Review, the 90% Design Review, and Implementation. In preparation for the 30% Design review is where many of the basic elements are determined, and the software side of the System Requirements and Design Specification (SRDS) document is created. At the 30%, only the expected amount of hardware and software interaction is included, along with the estimated complexity of both local and remote software need, and an approximate number of both the local and remote displays that need to be created. During preparation for the 60% Design Review the SRDS becomes better defined with what software components will need and their specifications. A major component included in this review is the Software Test Plan (STP). At this point, the STP is a general outline of test procedures and expectations, the detailed test steps will be added in the next stage. As a source for double checking that all major elements are included, the documents will go through peer review before reaching the 60% Design Review. The preparation for the 90% Design Review is similar to that of the 60%, but this time the updates to the documents will be more refined and finalized. Again, the documents will go through a peer review before reaching the formal 90% Design Review. After the documents have gone through all three phases of Design Review the team will enter the Implementation phase. This is where they will be able to start writing application software and creating displays in accordance to the SRDS that was approved through the Design Review phases [ILOA Training].

After I completed my required software training I was placed to work with the hypergolic sub-system team. When I entered the group they were finishing up their 90% Design Review and transitioning into the Implementation phase. This was a good time to enter because I was able to see firsthand how the team operated for both the 90% Design Review and the Implementation phase. The majority of my internship experience was spent in the Implementation phase where I developed and edited many of the remote displays to be used in the Firing Room to service the Orion MPCV's hypergolic sub-system.

VI. Display Editor

NASA's Display Editor software was used to create the remote displays, and the editing workspace is called the Display Editor. The SRDS that is created through the software lifecycle, as stated above, is the main document that outlines all the necessary components and a conceptual design to follow. While creating the display I worked very closely with the SRDS document. Not only did I use the document to help me design the displays, I also used this time to proof read the document for my hypergolic sub-system mentor.

The Display Editor has a very simplistic style of editing capabilities; the two main types of components were drawing components and symbols. The drawing components, as seen in Figure 1, were predominantly used as visual components with no physical functionality. These components were used mainly to separate and organize related groups of symbols and text together. More importantly, since the hypergolic sub-system works with both a fuel and an

oxidizer, and the layout of each system is very similar so, the use of the color coded visual drawing components was very important in distinguishing between the two.

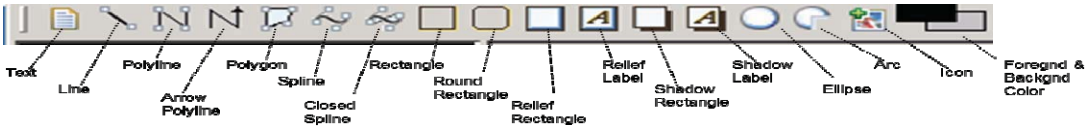

Figure 2: Drawing components tool bar in the Display Editor.

The symbol components are the heart and soul of the display, as seen in Figure 3 the symbols tool bar is comprised of four components: Text Measurements, Command Buttons, State Components, and Display Buttons. Three of the four of these symbols are tied to one or more Compact Unique Identifiers (CUI), which outlines specific limits, measurement types, commands, or run a script file.

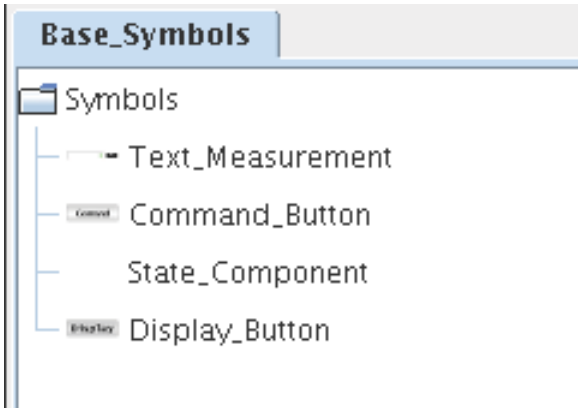

Figure 3: Symbols tool bar in the Display Editor

The text measurement symbol can display both numeric and enumerated measurements, such as; +1.00, 100.00, ON, OFF, OPEN, and CLOSE. Each text measurement symbol can only be tied to a single CUI.

Figure 4: Examples of text measurements.

Command buttons are used to issue commands such as turning something on or off, open or closed, or to primary or secondary. These command buttons can be tied to: an action, a status CUI, a script name, and at most 8 other arguments. The action argument is usually one word similar to what function you want the button to perform, such as start or stop. The status CUI argument is another CUI that controls the status of the command that button has performed, ensuring that the command has been sent successfully. The script name argument is the name of the script that the command button will find and then run through the code to complete the command. The other 8 arguments can be anything else that helps describe the button or what it is

attached to, for example its hardware number or other specifics that are necessary to run the code in the script file.

Figure 5: Examples of command buttons.

The state components are used to show the state of a certain element or grouping of the system. These symbols are highly important for ease of use and eliminating confusion for the console operator in the Firing Room. Having these state components the console operator can know exactly what is happening throughout the system at a glance. To provide uniformity throughout all the displays a standard color scheme for the images has been set, this also decreases the likelihood of confusion. These symbols must be tied to a set of images along with a single CUI.

Figure 6: Examples of state components.

Display Buttons are the only symbols that are not tied to a CUI; however, they do tie all the displays together. These buttons allow you to open another display from one that you already have open. Many of the hypergolic displays are related to one another and console operator will need to have multiple displays open to analyze a specific part of the system. Unfortunately, both the display and command buttons look similar to one another. This forced me to add a color coded header bar to the top of each display which is where I laid out all the display buttons needed for that display to avoid command/display button confusion.

Figure 7: Examples of display buttons.

VII. Completed Tasks

Below are a few summaries of some tasks I completed during my internship experience.

Once assigned to the hypergolic sub-system team I was given all the partially complete displays that a previous intern had made, along with the most recent copy of the Software Requirements and Design Specifications document (SRDS). For my first task, I was responsible

for naming or renaming all the display files with respect to the naming guidelines put forth by the hypergolics sub-system team.

My next task was to go through and confirm that all the symbols, such as text measurements and state components, had the correct CUIs associated with them as stated in the SRDS. This task was necessary due to the fact that since the last intern had worked on the displays some CUI names had changed and many more had been added to the database. The adding the new CUIs to the previously made displays would become one of the largest tasks for me to complete. It was my responsibility to go through all previously made displays and add all the symbols that had missing CUIs while the other intern was working with the displays. To complete this task, I again worked closely with the SRDS, especially the conceptual design and the charts of CUIs with their names and descriptions. Having most of the CUIs ingested into the database allowed me to be tasked with creating the remaining displays from scratch.

My first display creation task was to create two toxic vapor detection sensor displays, one solely devoted to the fuel and the other to the oxidizer. These displays will be used to detect if there is any leakage while the team is filling the respective tanks. This system is especially important due to the nature of these highly toxic potentially hazardous fluids.

My second display creation task was to construct a main menu display for the hypergolic sub-system console operator to use when first logging onto the Firing Room consoles. It needed to include all the display buttons that were linked to the most used displays. I was granted more freedom with this display, I was able to add more graphical design and pictures to this display, as it will only be used to open up the other displays. It does not have the strict guidelines for simplicity as the other displays. This was probably one of my favorite tasks because I had no limits and could truly show the software skills that I developed while working with the Display Editor.

I also created all nine Electronic Regulator Controller (ERC) displays that will be linked to their respective pressurization and purge panel display and integrated displays. Along with the ERC displays I also created the six flow meter displays, who will be attached to various displays via a display button. These displays were created using all four types of the symbols stated in section four along with the drawing components. Again, the SRDS was used immensely to create these displays.

Once many of the displays were mostly complete I submitted a build request to have the displays built into the configuration on the test set of consoles over in the Firing Room of the Launch Control Center (LCC). After, I received confirmation that the displays were built I was able to go over the Firing Room test consoles with my hypergolics mentor and perform some tests on the displays I created. We were able to see how the displays ran on the actual consoles, and more importantly against the running model of the system. Seeing the displays you spend

many hours working on actually controlling operations, even if it was only against the model for the time being, was a great experience.

VIII. Beneficial Exposure

Currently, I am working to complete my Bachelors of Science in Aerospace Engineering at Embry-Riddle Aeronautical University. This internship opportunity has exposed me to many new skills and hands on software experience. I will be able to bring these newly acquired skills back with me to my university to help further my education, and more importantly in my future career. Not only have I acquired technical skills, I have also become more confident in my business communication and documentation skills. Being at NASA's Kennedy Space Center, I have had the opportunity to gain real world industry experience, and see many projects development and implementation cycles in real time.

Since I was in about third grade it has been my dream to one day work for NASA and help create and develop spacecraft such as the Shuttle, and now the SLS-Orion. Being able to work with the Hypergolic subsystem and learn more about how their system works has been a great honor. Creating the subsystem's remote displays has showed me how both the hardware and software are linked together to support the end item, the Orion MPCV. It has been even more surreal to know that I have actually helped develop software that will be used in the Firing Rooms, relatively soon, to help service the actual space bound Orion capsule.

IX. Conclusion

In conclusion, the remote displays that support the hypergolic sub-system need to be simple, easily readable, but still contain all necessary information to control the subsystem from the remote location of the Firing Room. Being assigned as the sole remote display developer for the hypergolic sub-system team has provided me the experience to develop both my technical and leadership skills in a real workplace environment. I was able to show that I could balance the large responsibility along with completing tasks with due diligence. I am excited to continue to grow within the NASA family as I was offered to extend my stay with the hypergolic sub-system team as their remote display developer throughout the summer, as a NASA summer 2014 intern.

X. Appendix A – Acronyms and Abbreviations

Acronym/Abbreviation	Description
CM	Crew Module
CUI	Compact Unique Identifier
DC	Direct Current
GSE	Ground Support Equipment
HP GHe	High Pressure Gaseous Helium
ISS	International Space Station
LCC	Launch Control Center

LEO	**Low Earth Orbit**
MPCV	**Multi-Purpose Crew Vehicle**
MPPF	**Multi-Purpose Processing Facility**
SCCS	**Spaceport Command and Control System**
SLS	**Space Launch System**
SM	**Service Module**
SRDS	**System Requirements and Design Specifications**
STP	**Software Test Plan**
TCID	**Test Control Identifier Document**
VDC	**Volts DC**
VIP	**Vehicle Interface Panel**
VM	**Virtual Machine**
VPI	**Valve Position Indicator**

XI. Acknowledgements

I would first off like to thank NASA for the great opportunity I was given to experience how the aerospace industry works in real life, also thank you to all the NASA employees and contractors that have helped me complete my assigned tasks and taught me so many invaluable skills that I will continue to develop throughout my future career. I would especially like to thank my supervisor, Cheryle Mako for requesting interns and opening up this experience for college students like me. Also to my first mentor Kurt Leucht for his guidance and patience throughout the training process. Linda Crawford for all she has done to make us interns feel like part of the NASA family during our stay and to Greg for reviewing both my abstract and final report before publication. Also to, Bill Craig for taking the time out of his busy schedule to help me with any problems that I ran into. Finally, I would like to greatly thank Joey Parkerson for his time, guidance, help with the software tools used to complete my tasks, and all of the knowledge he has shared with me about the Hypergolic Subsystem.

XII. References

"NASA Facts: Orion Quick Facts", *NASA Lyndon B. Johnson Space Center,* Houston, Texas, 77068

NASA ILOA Core Team, "ILOA Training: The ILOA Software Development Lifecycle", NASA, September 6th, 2012.

Astronomy [website], URL http://www.astronomy.com//media/Images/News%20and%20Observing/News/2013/05/MSL_Orion.jpg [cited 07 April 2014].