Software Engineering Intern

Edgar Hernandez
NASA Kennedy Space Center
Major: Computer Science
NIFS Fall Session
11/10/2018

HYPERS Software Development

Edgar Hernandez[1]
University of California Santa Cruz, Santa Cruz, California, 95060

Nomenclature

API = Application Program Interface
CLI = Command Line Interface
CM = Crew Module
COTS = Commercial-off-the-shelf
CSV = Comma-Separated Values
GFAST = Ground and Flight Application Software Team
HTML = Hypertext Markup Language
HYPERS = Hypergolics Software
LCC = Launch Control Center
LCS = Launch Control System
KSC = Kennedy Space Center
NASA = National Aeronautics and Space Administration
PDF = Portable Document Format
SLS = Space Launch System
SM = Service Module
VIL = Vehicle Integration and Launch
VM = Virtual Machine
XLS = eXceL Spreadsheet
XML = Extensible Markup Language

I. Introduction

NASA has a long and decorated history of spaceflight innovation and achievements. The next great endeavor is NASA's Journey to Mars, which will be achieved with the Space Launch System (SLS) and Orion capsule. Developing and testing these systems is no easy feat. Commercial-off-the-shelf (COTS) tools do not always provide enough functionality for engineers to do their job efficiently, making internal custom-made tools is necessary to meet the expected launch date.

The purpose of this internship was to provide software support to the Storable Propellants and Hydraulic Systems Branch, specifically the Hypergolics Software (HYPERS) team. This included developing tools to parse unique measurements from the vehicle into the format specified by the HYPERS team. Displays were also created per requirements.

Another major component of this internship was to create an intuitive interactive offline graphing application. The current tool for plotting vehicle data does not have all the functionality and features that HYPERS would like. By inputting a vehicle data file, the application plots the data based on the time range and components the user would like to view. After the graph is generated, the user is able to zoom in, pan horizontally, add comments, hover over data points, and take a snapshot of the current state of the graph. These additional features will help engineers quickly investigate the relationship between vehicle components through data visualization.

II. Objectives

As SLS and the Orion capsule continue development, it is necessary to create remote Launch Control System (LCS) displays for subsystem development and Vehicle Integration and Launch (VIL) operations. Thankfully, the creation of these displays can be automated with an internal NASA tool. The caveat to this method is that the input file for this tool consists of unique identifiers in a particular scheme that is not readily available without excess amounts of tedious work. This is a frustrating limitation that can affect many phases of operations. These databases do provide

other exporting formats such as Extensible Markup Language (XML) and Portable Document Format (PDF) files. Extracting the necessary unique identifiers can also be automated using a modified version of the XML to Comma-Separated Values (CSV) file converter I created during my summer internship.

Data visualization is vital to understanding how components affect and interact with each other. One example of this is investigations after a vehicle mishap. Vehicle components generate large amounts of data. A single component recording data at a rate of 100 samples per second records 6,000 data points in one minute, 360,000 data points in one hour, 8,640,000 data points in one day, and 25,920,000 data points in three days. To effectively understand the story behind data, it is necessary to only view the essential components and time range. Additionally, investigating data becomes much easier when many features are supported. Required features for this graphing application include:

- Dynamic plotting of datasets on different y-axis, adapting to data type and number of components
- Zooming into a specific time range and have all plots zoom in as well
- Commenting on data points
- Providing printer friendly snapshots of current state of the graph
- Ability to hove the mouse over a data point and display the y-value at that x-value
    - More importantly, having the option to hover over a data point and display all y-values at that x-value

III. Approach

A. Preparation

Since I created this application from scratch, there was quite a bit of preparation. After understanding the requirements, my mentor and I researched different technologies to pick the right tools. First, we needed a way to parse vehicle data directly downloaded from NASA databases. Pandas is a Python library that is optimized for data handling and analysis. It provides special data structures and efficient operations for manipulating the data structures.

As for the actual graphing of data, there are many graphing libraries that each have their benefits and limitations. Listed below are some of the most popular graphing libraries we researched and their pros and cons:

Gnuplot:
Pros:

- Incredibly simple and easy to use
- Great for creating lots of quick graphs

Cons:

- Appearance is difficult to customize
- Limited interactive abilities

Matplotlib:
Pros:

- Fast and painless installation
- Many online documents and resources

Cons:

- Difficult to create interactive plots

Plotly:
Pros:

- Highly customizable and highly interactive
- Beautiful representation of data
- Active online developer community

Cons:

- Browser based
- Much higher learning curve than Gnuplot and Matplotlib

Considering all this, the Plotly Application Program Interface (API) open source library was chosen since it is possible to create highly interactive graphs that can fulfil the requirements for the desired features.

As for HYPERS Software Support, my preparation was fairly minimal because of previous experience received during my summer internship. For display creation, I already had access to the tools needed to create those displays and had already created plenty of displays in the summer. I had also created a data parsing tool in the summer that converted XML database files to CSV files which helped tremendously when doing similar tasks in the fall.

B. HYPERS Software Support

NASA has an in-house tool for automating the remote LCS display generation process. This takes input files formatted as unique identifier names line by line. The database that contains the identifier names and additional information does not have a readily available solution for providing a file in such a format, but can export XML and PDF file formats for requirement data which contain the identifiers for which we would like to create displays. By modifying an XML-to-CSV tool I created during the summer, we gained the ability to, given an XML file, produce a file that contains identifier names line by line. Although this was possible, many of the reports contained fewer than 5 identifiers, so in some cases it made more sense to simply copy and paste identifier names. This parsing tool is still important for operations because situations arise in which data must be extracted from long reports in specific formats that must be generated manually.

Additionally, other LCS displays were created manually using a display editor on a Unix-based Virtual Machine (VM). These displays were created manually, since the tool allows users to create displays with high fidelity compared to the automated display tool. Manually creating displays is usually reserved for important displays that provide an overview of the system and links to other subsystems displays.

C. Interactive Offline Graphing Application

The tool is launched by executing a Python script. The Command Line Interface (CLI) waits for the user to enter an Excel Spreadsheet (XLS) file name containing the vehicle data. The purpose of the script is to clean the data in the XLS file and transfer this to a text file which will be easier for the code to process. This data cleaning involves deleting blank rows, blank columns and eliminating unnecessary whitespaces all performed by the application's library. This cleaned version of the data is saved into a new text file and a Hypertext Markup Language (HTML) file is opened in the user's browser. The user then selects the new text file and the software begins storing the text file's data.

Now that the data is modified such that it is much easier to collect, all the time data points are recorded and all vehicle component data points are recorded into appropriate arrays. After this, a base layout is generated for the graph. Finally, the subplots and vehicle component data are entered into the graph. This outputs an interactive graph in the browser.

In order to fulfil the requirements, I implemented two custom Plotly buttons, 'Toggle all hover info' and 'Download image'. When 'Toggle all hover info' is ON, hover information is shown for all y-values corresponding to the cursors position. When the 'Download image' button is pressed, a printer friendly version of the graph is downloaded. This include a white background and all text is converted to black.

Figure 1 is an example graph. At the top right is the toolbar, from left to right 'Toggle all hover info', 'Download plot', 'Pan', 'Zoom', 'Zoom in', 'Zoom out', 'Autoscale', 'Reset Axis', 'Show closest data on hover', and 'Compare hover data'. 'Toggle all hover info' is demonstrated in **Figure 1** on the right side and shows all hover info for time 09/15/2018-183233.295. The grey boxes are comments which can be edited, deleted, or moved. If 'Toggle all hover info' is ON and a comment is added, that comment is added to all y values at that x value as demonstrated in the middle of the graph.

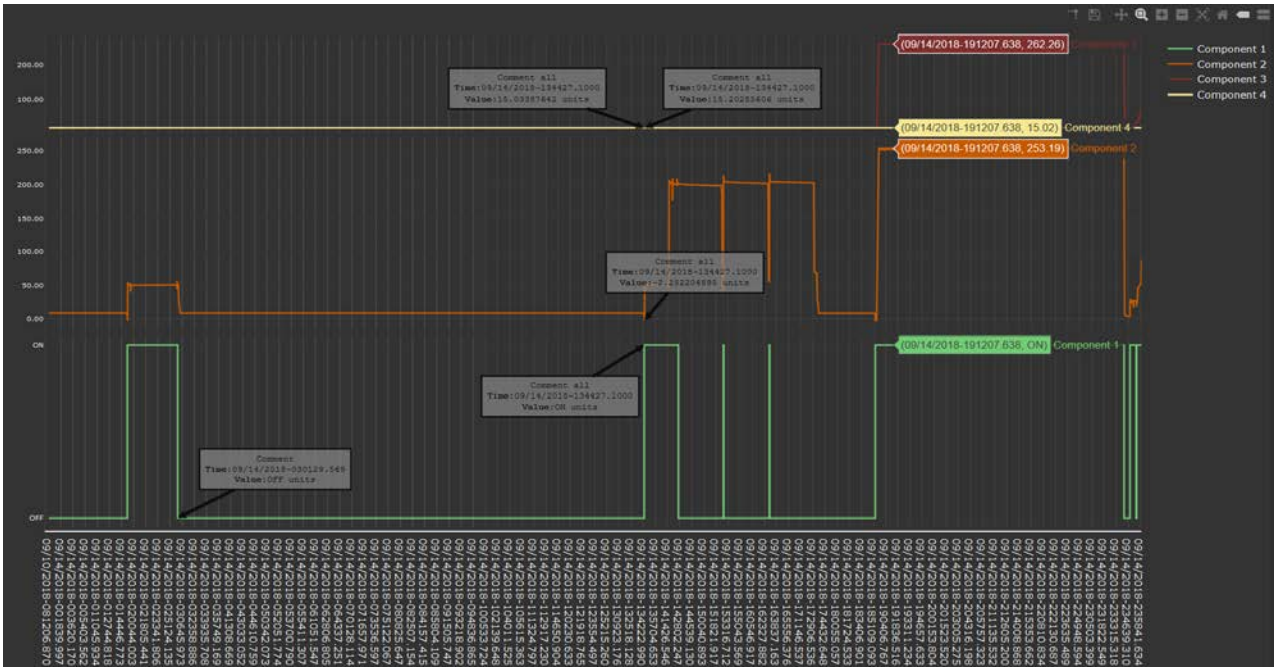

Figure 1. An example graph

D. Future Work

The interactive offline graphing application is currently still under development and screens preceding the graph need work. The design plan is to start with a file upload screen were you either drag-and-drop or select the input vehicle data file. After this, the times and component names are processed. The next screen would show a timeline and checkboxes next to every component name. The user would be able to select which components to graph and what time range they would like to see. These steps are necessary because Plotly's responsiveness decreases with an increase in data. So if the user only cares about 3 components in the last day of data, then the user can select these settings and Plotly will be able to provide a much smoother user experience.

IV. Conclusion

The interactive offline graphing application will provide a new method for exploring vehicle data. This will have a number of different uses for the HYPERS team; they will be able to easily create clear representations of data generated through many of their operations. The tool will hopefully enable the team to more efficiently investigate data whenever necessary.

LCS displays are necessary for remotely monitoring Orion's Crew Module (CM) and Service Module (SM). These displays will need to be continuously modified as requirements might change as well as unique identifier nomenclature. Hopefully, the work I did for the XML to CSV converter will make operations more efficient and allow the engineers to better allocate their time.

This software engineering opportunity has been a once in a lifetime experience. I'm extremely grateful to have been able to leverage my Computer Science fundamentals and apply them to real problems here at the Kennedy Space Center. This internship has also given me the opportunity to explore open source technologies. Plotly is an open source library with a lot of support from an active developer community. Using this library in the Offline Interactive Graphing Application project led me to becoming familiar with reading open source code and building upon that. This is an essential skill to have as many programming projects build on the great work of others!

Acknowledgments

I would like to thank my mentor Joey Parkerson for his guidance and help throughout the internship as well as my supervisor Pablo Aguayo. I would like to thank the HYPER/Ground and Flight Application Software Team (GFAST) for all the help along the way! I'd also like to give a final thank you to the KSC Education Office for this wonderful opportunity.