VCEWSTS: A Closed-Loop Flight Avionics Simulator for Mars 2020 Flight Software Development

Aaron Huang
Mentored by Johnny T. Chang and William J. Jay

Abstract

Flight software development for the Mars Science Laboratory (MSL) mission was supplemented by the WorkStation TestSet (WSTS). WSTS was a completely software-based non-real-time flight avionics simulator that saw at least an order of magnitude more usage than MSL hardware testbeds and eventually evolved into a verification and validation (V&V) platform for system engineers. Due to the success and utility of WSTS for MSL, we have decided to repurpose WSTS for the upcoming Mars 2020 mission. In this paper, I will describe the general layout of the Vision Compute Element WorkStation TestSet (VCEWSTS) and my role in its development.

Introduction

Designing and planning for a rover's Entry, Descent, and Landing (EDL) onto another planet is a challenging and uncertain task to say the least. Exotic environmental anomalies can negatively affect spacecraft hardware performance and cause all sorts of unforeseen behavior, posing an immense threat to the success of a mission. Engineers on MSL strove to plan for and overcome all possible anomalies by rigorously testing and validating relevant flight software given many permutations of off-nominal hardware behavior. The original WSTS was developed to provide an entirely software-based, non-real-time flight avionics simulator for flight software development[1]. WSTS for MSL was developed according to three core design principles.

First, flight software executing in WSTS should not be able to discern whether real or simulated hardware is being used. So far as the flight software is concerned, data generated from hardware should be completely consistent between hardware simulated through WSTS and real hardware used on a testbed. Consequently, this means that nearly all of the hardware must have a corresponding software model that accurately represents its real behavior. To this end, the second design principle mandates that hardware specification manuals should be used as reference to ensure software models were developed as faithful representations of hardware. This allows for a more modular and flexible testing procedure by which real and simulated hardware can be mixed according to real hardware availability while still producing accurate and useful test results. The last design principle stipulates that WSTS must support versatile fault injection capabilities for hardware testing in off-nominal situations. In this area, WSTS asserts a critical advantage over hardware testbeds as mission engineers are usually not willing to purposefully cause hardware to fail for the sake of testing. WSTS is able to inject faults from a functional level (e.g. forcing an incorrect number of desired camera subexposures) to a very basic hardware level (e.g. corrupting a particular bit in a peripheral's output data packet).

Over the course of the MSL flight software development timeline, WSTS averaged an order of magnitude more usage than hardware testbeds and evolved into a V&V environment suitable for systems engineers, systems integration and test engineers, and

mission operations engineers[1]. As WSTS was clearly a valuable and successful tool for flight software development on MSL, we've decided to recreate WSTS for the Mars 2020 mission in the form of VCEWSTS. My specific role in VCEWSTS development was to develop hardware interface models and integrate existing hardware peripheral models into VCEWSTS.

Mars 2020 EDL

When Curiosity touched down on the Martian surface, it used a new, "high-accuracy" EDL system to place Curiosity within a 20 by 7 km landing ellipse. The final landing site ended up being about 2.4 km away from the targeted landing site[2]. As landing inaccuracies implicitly incur significant financial costs over the length of time it takes to arrive at the targeted landing site, it is in our best interests to develop and test a more accurate EDL system. JPL has been developing Terrain Relative Navigation (TRN) technology for more precise EDL on rocky planets. TRN compares images taken from an onboard camera of the rapidly approaching surface against a preloaded onboard map to navigate and guide the descending spacecraft to more interesting and science-rich sites with less risk compared to traditional methods.

TRN will be used by the Mars 2020 rover as the first demonstration of the technology on a real mission[2]. The additional hardware required for TRN is housed in the Lander Vision System (LVS) which mainly consists of two elements. The first element is the Vision Compute Element (VCE) which takes the form of a chassis that houses two important cards – the Computer Vision Accelerator Card (CVAC) and the BAE RAD750 flight processor board, and the LVS Camera (LCAM).

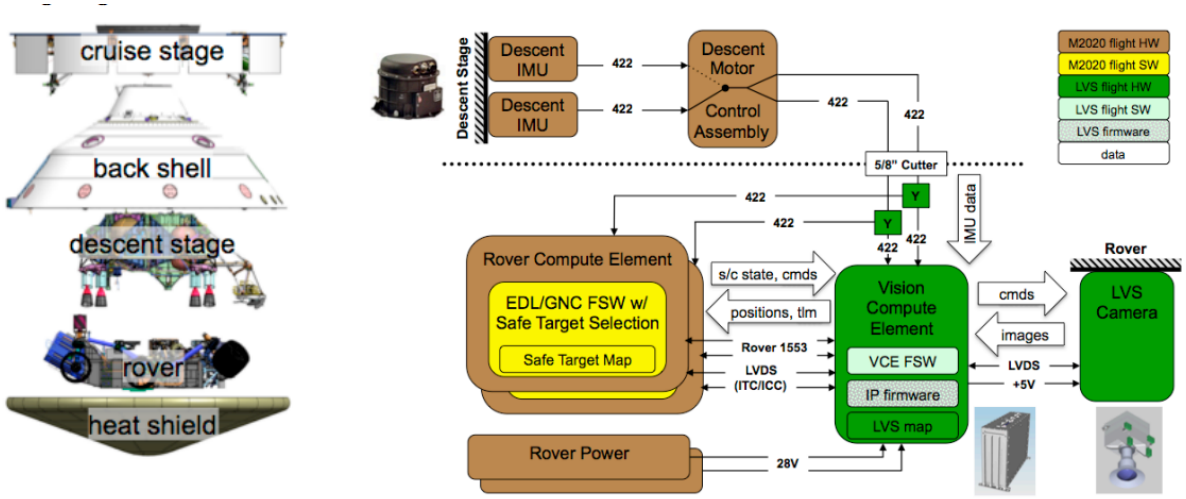


Figure 1: A block diagram of LVS and its peripherals[2]

Vision Compute Element

The Vision Compute Element was designed to inherit as much architecture from MSL as possible[2]. The VCE contains a Compute Element Power Conditioning Unit (CEPCU1) which provides power to the CVAC and RAD750. The RAD750 provide general processing capability and runs the Vision Compute Element Flight Software. The CVAC is a new card developed specifically for LVS that handles all data interfacing for the VCE (including DIMU, LCAM, 1553, ITC/ICC). My main responsibilities for JPL this summer were to develop simulation models for the CVAC/peripheral interfaces.

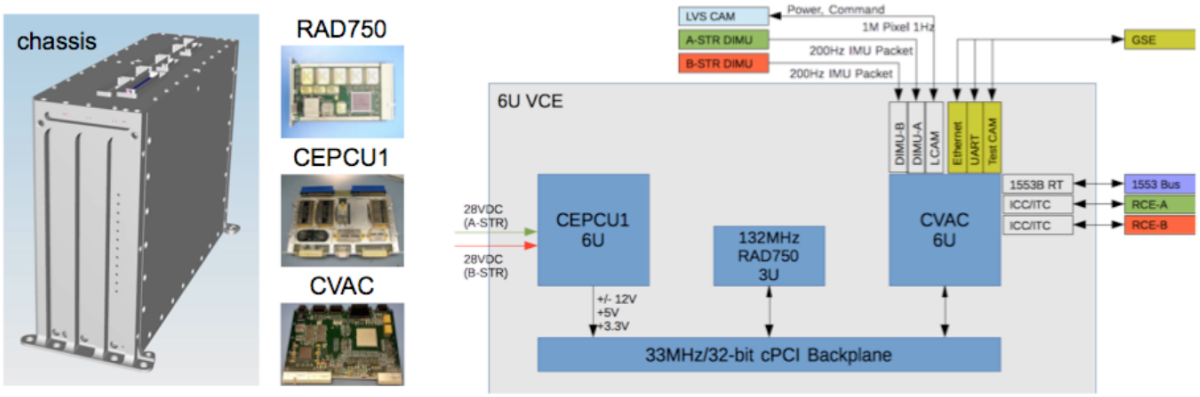


Figure 2: VCE Block Diagram and Chassis[2]

CVAC – DIMU Interface

Every 200 Hz, the Descent Inertial Measurement Unit (DIMU) sends a data packet to the CVAC containing attitude, position, velocity, and other pose measurements. The CVAC must then repackage each packet into an augmented measurement pack and store it into RAD750 memory in a DMA fashion. Each 20 byte raw DIMU packet is repackaged into a 32 byte measurement package that includes packet ID number, spacecraft time in seconds and 2^{-20} seconds resolution, and a descriptive measurement packet flag.

CVAC – LCAM Interface

The CVAC – LCAM interface's main job is to schedule exposures at the behest of flight software and store rendered images into DDR2 memory. Flight software writes the start time of the exposure to a register that the interface then reads and acts upon accordingly. The details of each exposure are dictated through a LCAM configuration file that gives users control over things like dust effects and number of subexposures.

I was able to generate enough images such that going through each one of them individually would have been quite painful. Luckily, I was also able to figure out how to throw these image sequences together in a stop motion video. The video can be found in the appendix.

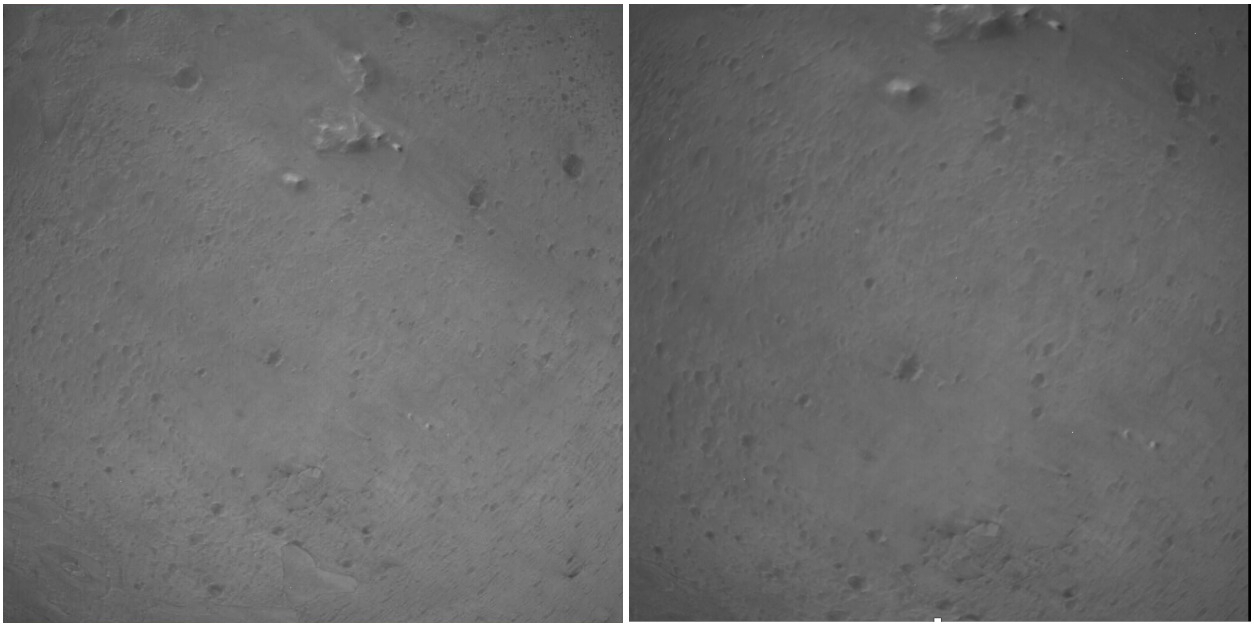


Figure 3: Images rendered by the LVSS LCAM module of the Martian surface near the Jezero crater

LVSS Models

Developing a DIMU model from scratch that accurately generates data packets with realistic noise and other effects would be a monumental task far beyond the scope of a summer internship. Luckily, the Lander Vision System Simulation (LVSS) team has kindly shared a couple of their hardware peripheral models that generate realistic DIMU data, LCAM images, and a couple of other things we need[2]. A block diagram describing data flow for VCEWSTS can be found in the appendix.

VCEWSTS

The VCEWSTS application is heavily based upon the original WSTS simulation for MSL. From a front-facing perspective, the three key elements of the WSTS simulation are the Simulator Console, MSL Simulator, and VxWorks Simulator (for flight software).

The next page depicts a screengrab of the VCEWSTS GUI. The three console windows draw direct parallels to the original WSTS architecture. The Master window directly descends from the Simulator Console, providing a command line interface for users to direct commands e.g. running the simulation for a set number of time slices. The WSTS window is analogous to the MSL simulator and contains all hardware model output. In the window shown below, validity checks from the DIMU model and general exposure information from the LCAM model are displayed. Finally, the FSW window is the VCEWSTS version of the VxWorks Simulator in WSTS. The FSW window shown below is displaying DIMU data read from the DMA buffer in RAD 750 memory.

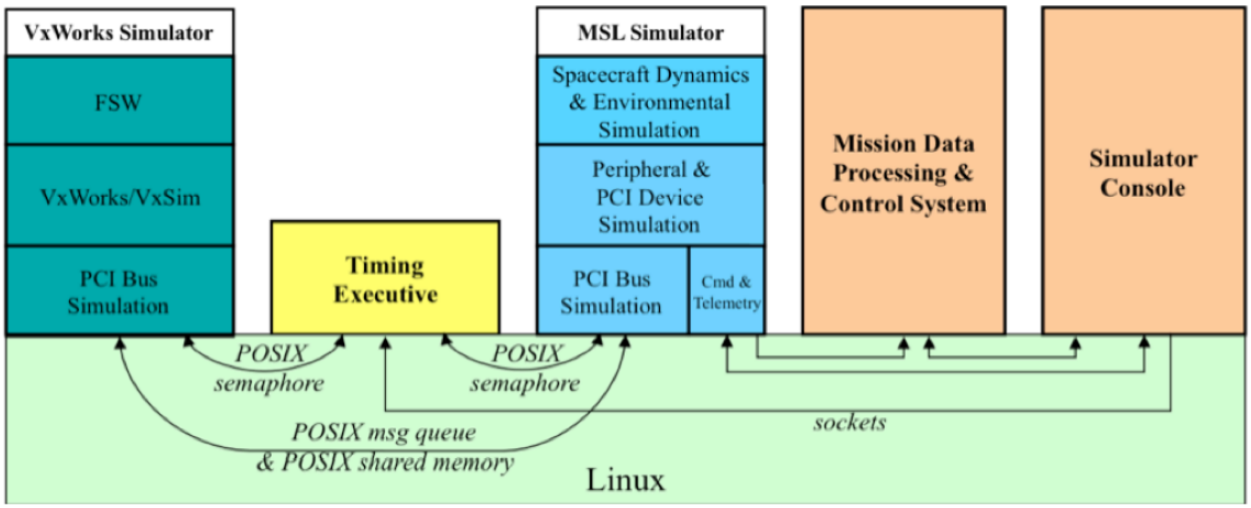


Figure 4: WSTS Architecture[1]

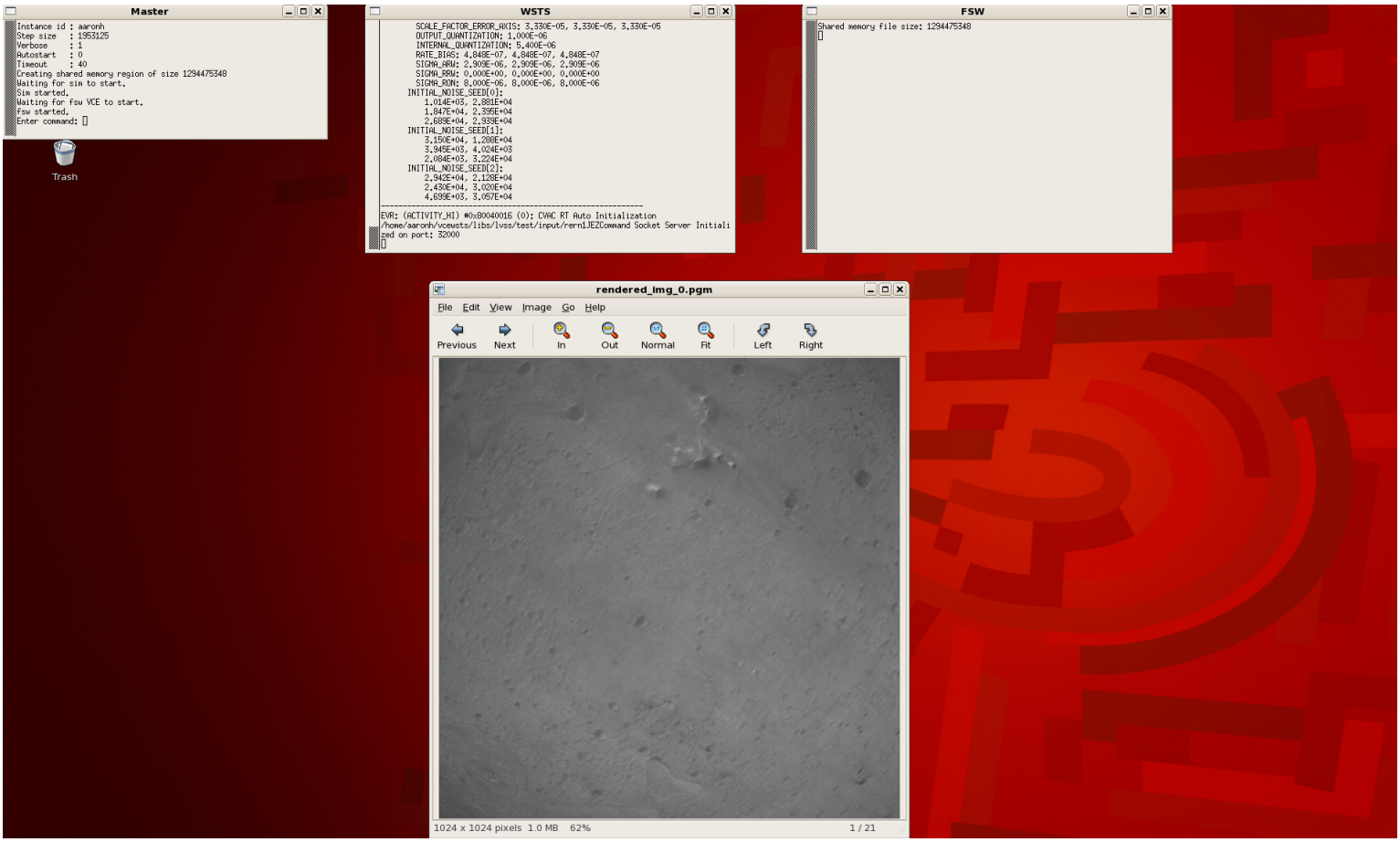


Figure 5: Screengrab of the current VCEWSTS build

Improvements and Future Direction

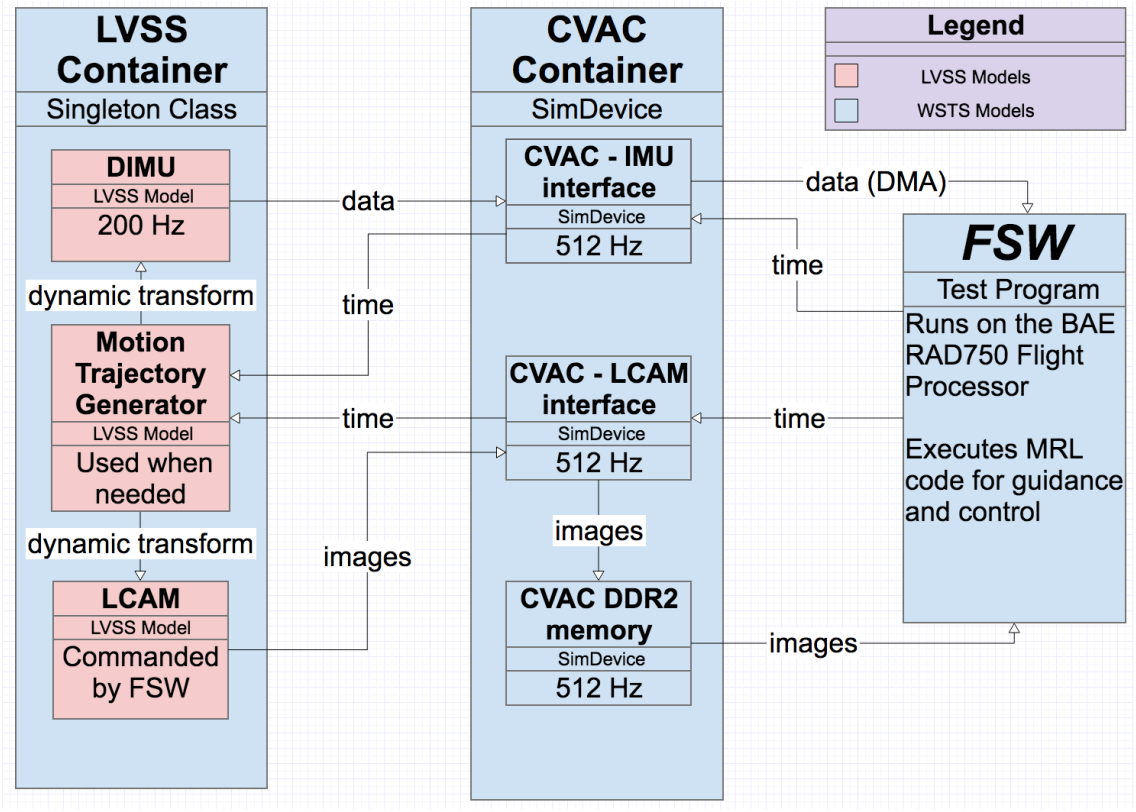
While many of the VCEWSTS components have been created, we have yet to completely integrate all of the components. For instance, LCAM image transfer from interface to DDR2 memory has yet to be put through the 1553 communication model. Of all the disparate models, the only ones that have been incorporated are the CVAC interfaces, integrated LVSS models, and a couple of operational necessities e.g. timing unit. Further integrating the hardware models will bring VCEWSTS closer to its end goal of being a comprehensive VCE simulation. Finally, adding more user interfacing capabilities in the form of command line arguments or easier configuration modification will improve user experience and speed up FSW development and testing.

Acknowledgements

I would like to thank my mentors Johnny Chang and Will Jay for their immensely helpful guidance and support throughout my internship. Without their knowledge and willingness to sit down and help me resolve my programming bugs and gaps in higher level understanding of VCEWSTS, I would never have gotten nearly as far as I did. I also want to thank the LVSS team for creating fantastic peripheral models and Seth Aaron in particular for helping me integrate them into VCEWSTS. Lastly, I'd like to thank Mitch Ingham and Jeffrey Levison for putting me in touch with Johnny and introducing me to the invigorating and intellectual JPL community.

Appendix

A stop motion video of what the LCAM sees during a simulated trajectory descent above the Jezero Crater



A block diagram depicting data flow from LVSS models through CVAC interfaces to RAD 750 for flight software usage

References

[1] D. Henriquez, T. Canham, J. T. Chang, E. McMahon, “Workstation-Based Avionics Simulator to Support Mars Science Laboratory Flight Software Development”

[2] A. Johnson, S. Aaron, J. T. Chang, et al., “The Lander Vision System for Mars 2020 Entry Descent and Landing”