

# Mars Science Laboratory Boot Robustness Testing

Payam Banazadeh<sup>1</sup>, Danny Lam<sup>2</sup>

*NASA Jet Propulsion Laboratory, California Institute of Technology, Mail Stop 321-151, 4800 Oak Grove Drive, Pasadena, CA, 91109, USA*

**Mars Science Laboratory (MSL) is one of the most complex spacecrafts in the history of mankind. Due to the nature of its complexity, a large number of flight software (FSW) requirements have been written for implementation. In practice, these requirements necessitate very complex and very precise flight software with no room for error. One of flight software’s responsibilities is to be able to boot up and check the state of all devices on the spacecraft after the wake up process. This boot up and initialization is crucial to the mission success since any misbehavior of different devices needs to be handled through the flight software. I have created a test toolkit that allows the FSW team to exhaustively test the flight software under variety of different unexpected scenarios and validate that flight software can handle any situation after booting up. The test includes initializing different devices on spacecraft to different configurations and validate at the end of the flight software boot up that the flight software has initialized those devices to what they are suppose to be in that particular scenario.**

## I. Introduction

The Mars Science Laboratory (MSL) flight software team is in charge of developing flight software (FSW) for use by the MSL rover to conduct in-situ exploration on the surface of Mars. The main goal for MSL is to assess whether Mars ever was, or is still today, an environment able to support microbial life. It is hoped that MSL would be able to determine Mars’ “habitability.”[1] MSL is scheduled for launch in November of 2011 and is expected to arrive on Mars on August 5 of 2012. MSL or “Curiosity” will land at the foot of a layered mountain inside the planet’s Gale Crater. The target crater spans 96 miles (154 kilometers) in diameter and holds a mountain rising higher from the crater floor than Mount Rainer rises above Seattle.[2] A picture of the Mars Science Laboratory is displayed in Fig. 1.

The scope of the flight software covers all mission phases which include:

- Launch
- Cruise
- Entry-Descent-Landing (EDL)
- Surface operations of the rover

---

<sup>1</sup> NASA USRP Summer Intern, Jet Propulsion Laboratory, The University of Texas at Austin – Aerospace Engineering

<sup>2</sup> MSL BIT/FIT Team Lead, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109

Key Functional areas of the flight software include:

- Command and telemetry processing
- Guidance and control during cruise
- Entry-descent-landing through Martian atmosphere
- Rover deployment
- Rover navigation and mobility
- Robotic arm control
- Science payload interfaces



Figure 1: NASA’s Mars Science Laboratory, a mobile robot for investigating Mars’s past or present ability to sustain microbial life. [1]

The FSW itself is created by the FSW developers. Every night the entire work done by the FSW developers is compiled into a Nightly Cron (NC). A similar but more thorough process is done every couple weeks, known as a build, and every month known as a release. Before software is released to a user outside the software development team, it is necessary for the software developers to test their product extensively and make sure there are no misbehaviors. The developer meets this requirement by implementing and executing hundreds of tests in different environments and scenarios. One of the areas that would especially need extra testing is the flight software booting process. I worked with two other interns to develop a test environment that allows FSW to be tested while booting up. The testing that we do is low fidelity, but much cheaper than the other testing procedures higher up the ladder as shown in Fig. 2.

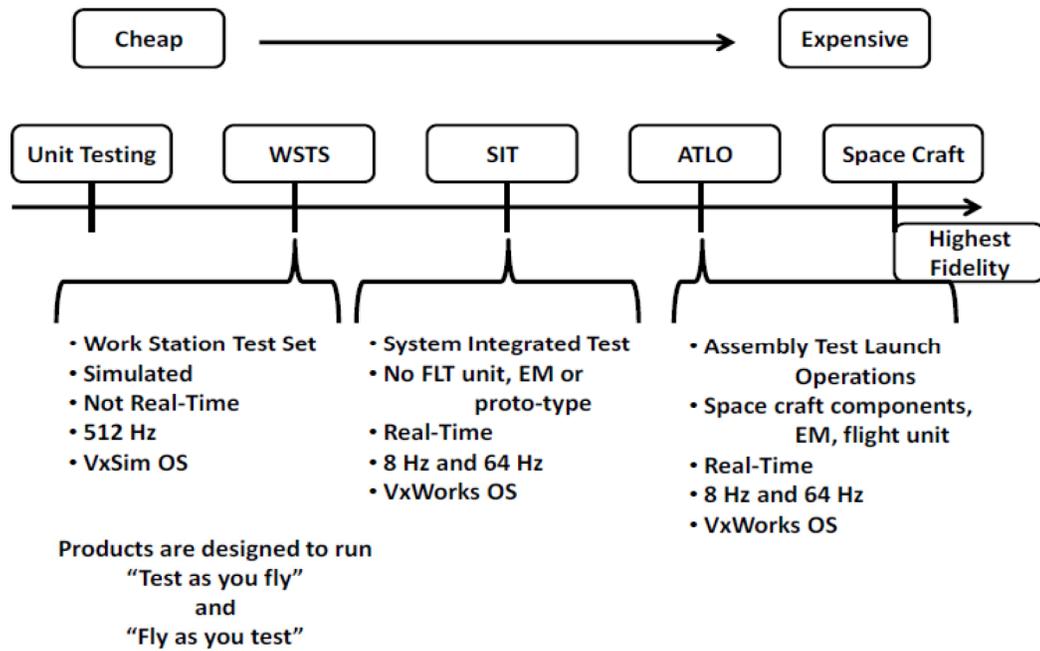


Figure 2: This figure illustrates the process that the FSW goes through before reaching the Space Craft. The goal of boot testing is to discover FSW defects early prior to delivering to SIT or ATLO.

The basic objectives that I set out to accomplish at the beginning of the term were as follows:

- Become familiar with MSL
- Become Familiar with MSL FSW
- Review FSW module documents and various test operating manuals
- Meet with appropriate MSL personnel as required
- Develop and present test plans
- Develop the test environment
- Perform test
- Write status report
- Write abstract
- Continue developing tests and testing FSW as required
- Write status report
- Write final presentation
- Write final report

## **II. Flight Software Boot Robustness Testing**

The objective of the MSL FSW Boot Robustness testing is to boot the flight software in a variety of hardware and software configurations to verify that initialization is successful and control is regained correctly. This test seeks to demonstrate that the flight software will use the hardware and software states to correctly determine and command the spacecraft configuration through an exhaustive range of different scenarios. The goal of this task is to gain improved confidence that the complex and critical boot process is robust to hardware and software scenarios that may be seen during the mission. Before the tests can be run, a system for performing these tests needs to be developed. The tests will be run under a software simulation through Work Station Test Set (WSTS) with the flight software built for Linux. Because of the large number of possible boot scenarios, however, it is desired that the testing be automated to the extent that:

1. Starting configurations are provided by the tester
2. FSW boots under each configuration autonomously
3. After specified duration, output is recorded and verified against expected configuration

In addition to learning about MSL hardware, flight software, and boot process, the major components to this task were: to define an input and output vector format; to generate custom non-volatile memory files containing software state on the Rover Compute Element (RCE) and Rover Remote Engineering Unit (RREU); to generate a list of SSE(support simulation equipment) commands to configure the hardware simulation before boot based on the input configuration; to create an execution framework that facilitates running the tests.

### **A. The Approach**

The FSW, during boot, uses hardware and software states to determine and command the appropriate vehicle configuration. This is a complex process that needs to be tested thoroughly in order to have a successful mission. This Boot Robustness Testing will demonstrate that the FSW will boot successfully under a wider range of different scenarios.

The general approach was as follows:

1. Parse an input vector of hardware and software states and create input files and SSE command lists
2. Star Linux WSTS using these input files
3. Send the SSE commands to configure hardware states

4. Start FSW, and run a fixed number of time steps (e.g. 20 minutes simulated time )
5. Stop the simulation
6. Parse the output files for results
7. Verify the results against the output test vectors
8. Cleanup files

## B. Implementation

The first step was to come up with input and output configurations. These configurations had to specify different hardware and software states of devices such as health states, prime states, operating state, etc. Since the goal was to be able to test hundreds of tests automatically, an input/output configuration generator needed to be developed. For this matter, a script was written that would generate the output configuration file base on the input configuration file. The second step was to create a program that would generate the following:

1. nand.dat image, this is the non-volatile software state which is resident on each RCE
2. srm.dat image , this is the non-volatile software state which is resident on each RREU and is shared between RCEs
3. SSE command list, a list of SSE commands needed to configure the initial state of the simulated hardware

The next step was to develop an execution framework that would be able run the FSW by taking the SSE commands so we could manipulate the simulation and configure different devices. This execution framework would need to be able to tap into SSE simulation environment before running FSW and establish the initial hardware states by sending SSE commands to the running Linux session. After running the FSW the execution needs to be stopped after duration of time specified by the input configuration and the state information needs to be saved in different log files for post processing. The last step is to verify that the FSW did what it was suppose to do. This post processing step was divided into three different parts:

1. Create a script that captures SSE hardware state information, this includes memory and register values
2. Create a script that extracts state data from updated nand.dat and srm.dat images
3. Create a script to verify output vector

The post processing script would open up the output configuration file, read in everything and save different states in their appropriate arrays and then compare them with the log files and device states extracted from the simulation. At the end of this process the post processing script would generate a report that says which condition wasn't satisfied and why it wasn't satisfied. This way the developer can go back and figure out whether there is a problem in the FSW or the simulation.

There are three forms of telemetry known as Event Records (EVRs), Engineering, Health & Accountabilities (EHAs), and Data Products (DPs). We receive these forms of telemetry after sending commands to the rover, or in our case WSTS, which simulates the rover. Part of the post processing script's task was to make sure some certain EVRs happened during the FSW boot up. EVRs are important because they tell you whether the test is failing because of a problem in your scripts or because the FSW didn't do what it was suppose to do.

## C. Challenges

After developing the framework and being able to run a complete end to end test without any problems we started to generate input and output configurations for different scenarios that the FSW team was interested to test. Generating the input configuration is easy because they could be generated completely randomly but coming up with the correct output configuration for each input configuration and what the FSW needs to do in each scenario requires knowledge of all the FSW components. The way we came up with output configurations was to talk to experts in the area and also run hundreds of test with a pre-defined output configuration and elaborate on one by one case and update the output configuration. This process was time consuming and very complicated. Another

challenge is how long it would take to run each test. On average it takes between 4-7 minutes to run each test and if we were to run hundreds of tests it would take hours or days to complete a set of tests. For this matter, if the FSW team decided to run thousands of tests at the same time they would need to use a super computer. Since we were aware that generating a set of output configuration would be very complicated we have added a feature to the post processing script that would check for some basic but important configurations no matter if an output configuration is provided or not.

### **III. Conclusion**

We were able to add test capabilities that the MSL FSW team didn't have before. The MSL FSW team can now test their FSW in all different software and hardware modes and configurations and make sure that the FSW is doing what it is suppose to do. Due to the natural complexity of this project there is a strong emphasis on testing, as failure is not an option. The entire mission success is dependent on thorough testing of all the components such as FSW.

### **Acknowledgments**

I would first like to thank my mentor, Danny Lam, for allowing me to experience first-hand what it is like to work on the MSL team. I would also like to thank Steve Scandore, the project manager for the Boot Robustness Testing, for his time and patience with me which made my experience very joyful. I would also like to thank Dee Leang, Cindy Oda and the entire MSL FSW team for being a substantial help with whatever questions that I have had. This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by undergraduate Student Research Program (USRP).

### **References**

1. Mars Science Laboratory URL: <http://mars.jpl.nasa.gov/msl/mission/overview>
2. Mars Exploration Rovers URL: <http://marsrovers.jpl.nasa.gov>