

# **Final Report**

Justin D. Jones

Jet Propulsion Laboratory

Major: Electrical Engineering and Computer Engineering

USRP Spring Session

Date: 05-02-2011

# Mars Science Laboratory Flight Software Internal Testing

Justin D. Jones<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*

The Mars Science Laboratory (MSL) team is sending the rover, Curiosity, to Mars, and therefore is physically and technically complex. During my stay, I have assisted the MSL Flight Software (FSW) team in implementing functional test scripts to ensure that the FSW performs to the best of its abilities. There are a large number of FSW requirements that have been written up for implementation; however I have only been assigned a few sections of these requirements. There are many stages within testing; one of the early stages is FSW Internal Testing (FIT). The FIT team can accomplish this with simulation software and the MSL Test Automation Kit (MTAK). MTAK has the ability to integrate with the Software Simulation Equipment (SSE) and the Mission Processing and Control System (MPCS) software which makes it a powerful tool within the MSL FSW development process. The MSL team must ensure that the rover accomplishes all stages of the mission successfully. Due to the natural complexity of this project there is a strong emphasis on testing, as failure is not an option. The entire mission could be jeopardized if something is overlooked.

## I. Introduction

The Mars Science Laboratory (MSL) flight software team develops flight software (FSW) for use by the MSL mission to conduct in-situ exploration on the surface of Mars. The Mars Science Laboratory is a rover that will assess whether Mars ever was, or is still today, an environment able to support microbial life. In other words, its mission is to determine the planet's "habitability." [1]

The Mars Science Laboratory is due to launch in the month of November, 2011. A picture of what the Mars Science Laboratory would look like on Mars is displayed in Fig. 1. The landing site location will be decided relatively soon. Many landing site locations have been proposed for MSL, most of which have been addressed or dismissed in the MSL Landing Site Workshops and now only four landing sites remain. The final four candidate landing site locations for MSL include Gale crater, Mawrth Vallis, Holden crater, and Eberswalde crater. [2]

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

- Launch
- Cruise
- Entry-descent-landing
- Surface operations of the rover

---

<sup>1</sup> NASA USRP Spring Intern, Jet Propulsion Laboratory, University of Michigan - Dearborn Undergraduate Student

<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

I am currently apart of the FSW Internal Test (FIT) Team who are responsible for designing, implementing, and executing FSW acceptance tests to verify proper FSW behavior. FIT is intended to focus on verifying FSW functionality at module- and inter-module levels of testing. [3]

## II. Flight Software Internal Testing for the MSL

I have been involved with testing of the flight software modules that have already been developed or are under development and require more thorough and detailed testing. I have been assigned the functional tests of DIMU, RIMU, PTY, LATCHV, PWR, AUT and IMG. The result is a completed and executable test procedure which verifies the required functionality of a specific flight software domain.

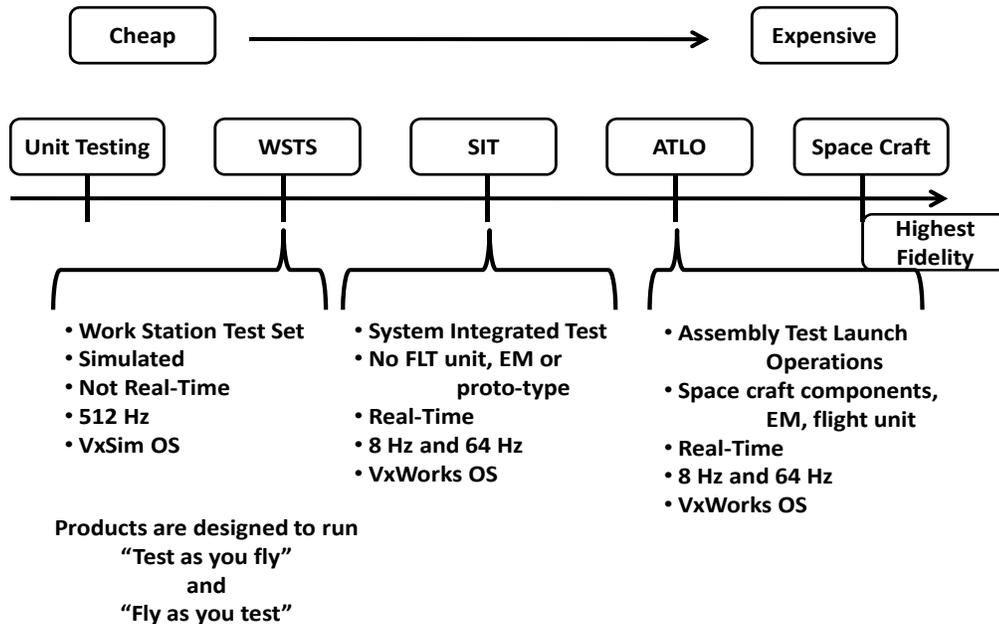
The FSW itself is created by the FSW developers. Every night the entire work done by the FSW developers is compiled into a Nightly Cron (NC). A similar but more thorough process is done every couple weeks, known as a build, and every month known as a release. I create test scripts written in the Python programming language that works with the simulation software known as the Work Station Test Set (WSTS). The FIT team can accomplish this with WSTS and the MSL Test Automation Kit (MTAK). MTAK has the ability to integrate with the Software Simulation Equipment (SSE) and the Mission Processing and Control System (MPCS) software which makes it a powerful tool within the MSL FSW development process. 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. The way that the FIT test scripts themselves relate to the simulation software is depicted in Fig. 3.



**Figure 1: NASA's Mars Science Laboratory, a mobile robot for investigating Mars' past or present ability to sustain microbial life. This picture is an artist's concept portraying what the advanced rover would look like in Martian terrain, from a side aft angle. [1]**

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 my case a program called WSTS, which simulates the rover. I make sure that all of the telemetry has been sent successfully and properly. An example of checking to make sure the telemetry is properly sent is the case where we want certain telemetry not to be successful and thus we must expect failure to complete the command. There are many instances at which commands are modified or deleted. During these cases, I must contact the FSW developer in regards to the command update.

In general, each test is split up into different sections such as variables, setup procedures, main sections, sections to test and a test matrix. The main sections to test have been previously defined within the Functional Design Description (FDD) documentation. In a large number of these sections, the script has either previous existing errors, sections that are incomplete, or sections that are manually verified. I run through these scripts and fix the errors, complete the incomplete sections and automate the manual sections.



**Figure 2:** This figure illustrates the process that the FSW goes through before reaching the Space Craft. The goal of FIT 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:

- Became familiar with the project objectives
- Reviewed the Functional Description documents and level 4 requirements to get an understanding of what the module to be tested is expected to do
- Reviewed the FSW module documentation and talking with SW developer to gain an understanding of how the module to be tested is implemented.
- Reviewed the various test operating manuals and work with integration and test engineers to gain an understanding of the test infrastructure and processes.
- Developed and present test plans describing the specific functionality and methods that the student proposes to implement.
- Write, test cases, usually using python scripting language that will exercise the module to be tested.
- Build the test executables and execute the test cases.
- Evaluate test results, debug unexpected behavior with the primary objective to find and report defects in the flight software so that they can be corrected before causing problems.
- Document tests conducted and results.

### III. Updating FIT Test Scripts

Two tests that I have been working on the majority of my time here are DIMU and PWR functional tests. DIMU has not been updated in quite some time and thus is outdated.

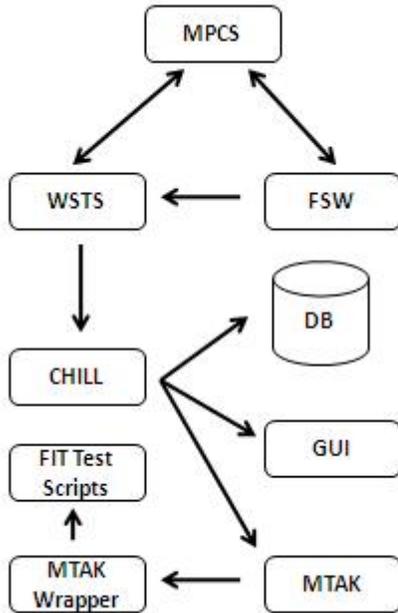


Figure 3: This diagram shows the system at which the FIT test scripts are related.

more of a challenging task in the fact that I needed to know what was supposed to result from the DPs in order to verify that the contents were correct. I resolved this issue by becoming familiar with how the DPs are structured and speaking with the DIMU FSW developer, Clayton Williams<sup>3</sup>.

Within the other sections I have been updating in such a fashion that allows them to run faster and future viewers of the scripts can read them better. With less unnecessary function calls and conditional statements, the tests look cleaner and run faster too. Updates to the software have taken care of the requirements within these conditional statements and thus they are no longer required. Within other instances, the code can be optimized to have less conditional statements and still accomplish the same task. This affected most sections for all scripts.

There are cases where the venue, in my case WSTS, is limited and cannot perform the task that is required. I must insert an exception within the test to only test what I can due to this limitation.

When I was updating the AUT script, there were many older functions that required user input for the test to run successfully. With newer functions, I was able to automate the script, so the user does not have to be present while the script is running.

### C. The Final Steps

Some previous testers did not have enough time to finish some of the sections that require testing. After I have taken care of the previous tasks, I complete the sections that were not started. These sections require much more

<sup>3</sup> DIMU FSW Developer, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109.

### A. The Approach

When I first approach a test, I begin by making sure that everything works properly. Many different issues may arise during a test. The WSTS may be preventing certain commands from performing successfully. The current state at which a device is in may cause a test to fail. In many cases it may just be a command which was modified or deleted. After diagnosing the problem, I either figure it out on my own or contact the FSW developer.

DIMU had the case where one of the DIMU commands had been deleted. This required me to go through the test and modify it in such a fashion that the newly deleted command does not affect the rest of the test. With many other tests, certain CBM commands have been modified a couple times which require multiple updates.

### B. Intermediary Steps

The next step in the process is to identify all sections that require manual verification of success and to automate them to pass successfully without manual verification. Within this step I needed to parse and then verify the contents of the DPs. This was

knowledge of the given device. Given a certain test section description, I must come up with the commands and conditions for the test.

There are some cases in which the FIT tests are so outdated that they no longer fit the new template that has been set up. Each test is split up into different sections such as variables, setup procedures, main sections, sections to test and a test matrix. Some of the older tests simply run through everything sequentially, while the newer tests run through the main section, then through the test matrix which calls the other sections.

There is a previously made set of requirements for each module. My responsibility is to verify that the given module passes this set of requirements. With some of the tests like RIMU and AUT, the infrastructure is outdated and these tests do not include some of the requirements. Using a template and a listing of all the requirements, I created a new python script that is updated with all the requirements. More than likely these requirements have not yet been coded, and thus I would go through and write the code to test these requirements.

#### **IV. Challenges Regarding MSL FIT Testing**

The major challenge was at the beginning. Being introduced into this complex project that has dozens of modules, many thousands of lines of code, and all custom to their own system has quite the learning curve. Unlike other programming internships that I have had, I cannot just search the internet for a reference like I can for web design with php scripts. I have however overcome these challenges and feel much more comfortable with the environment.

A challenge with evaluating the test results is determining whether the test script needs to be updated or that there is an issue elsewhere such as the flight software. Whenever something like this arises there is always someone there to ask for assistance.

I have mentioned some challenges already with the exception of my greatest challenge. WSTS has encountered a fatal EVR and shortly thereafter crashing during the PWR test. During my initial run of the PWR test, commands began to fail continuously after a certain point. I managed to narrow it down to the section, and soon after the function. After manual verification of the function, I was able to narrow it down more specifically to what the error was.

I worked with Cindy Oda<sup>4</sup> regarding this problem. I created a session report, backed up the temporary files related to the run, and spoke with other experts before ultimately sending in an anomaly report. After creating the session report I was able to identify the specific fatal EVR. I compared the session report to previous reports that ran properly. Eventually, Cindy Oda helped me find a way to run the script successfully. In this case, I required additional set up for the script to be successful.

#### **V. Conclusion**

As of this moment, all of my test scripts are up to date and run without errors. The MSL team must ensure that the rover accomplishes all stages of the mission successfully. Due to the natural complexity of this project there is a strong emphasis on testing, as failure is not an option. The entire mission could be jeopardized if something is overlooked.

#### **Acknowledgments**

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by NASA Undergraduate Student Research Program and the National Aeronautics and Space Administration.

---

<sup>4</sup> MSL FIT Lead, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109

## NASA USRP – Internship Final Report

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. Marc Pack and Cindy Oda have been a substantial help with whatever questions that I have had. I would like to thank the Jet Propulsion Laboratory and the NASA Undergraduate Student Research Program for offering me this extraordinary internship experience at NASA's Jet Propulsion Laboratory as I continue my degree in Computer Engineering and Electrical Engineering at the University of Michigan – Dearborn. I would like to thank other members of the MSL team: Benjamin Cichy, Constantine Chen, Clayton Williams, Vandi Verma and the rest of the MSL team for assisting me during my stay.

### References

1. Mars Science Laboratory URL: <http://mars.jpl.nasa.gov/msl/mission/overview>
2. Mars Exploration Rovers URL: <http://marsrovers.jpl.nasa.gov>
3. Mars Science Laboratory Wiki URL: <http://msl-wiki:8080/JSPWiki/Wiki.jsp?>
4. Mars Science Laboratory Fact Sheet URL: [http://mars.jpl.nasa.gov/msl/news/pdfs/MSL\\_Fact\\_Sheet.pdf](http://mars.jpl.nasa.gov/msl/news/pdfs/MSL_Fact_Sheet.pdf)