Fomenting Change
(Instigating Change in a Risk Averse Environment)

Barbara Streiffert
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109 USA

Diane Conner
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California 91109 USA

CAST Grand Rapids, Michigan August 3 – 5, 2015

Abstract:
Many companies are risk averse, especially when the risk is to human life and/or involves significant financial impact. The Jet Propulsion Laboratory (JPL) builds, launches and operates spacecraft that flyby, orbit and roam planets, moons and asteroids in our Solar System. Loss of a mission and the scientific knowledge that can be obtained by the mission is the highest concern, and engineers focus their talents on obtaining high science return by ensuring spacecraft safety. In addition, because spacecraft are unique and can cost billions of dollars to build and operate, the loss of a spacecraft is very costly. The development and operations environments for missions are process heavy and risk averse, yet change is necessary to reduce costs and improve the quality of software used for the missions. It is difficult to marry change and risk aversion. An approach for changing software, software testing and processes include:

1. understanding the business use cases
2. brainstorming ideas
3. performing changes incrementally
4. obtaining approval or asking for forgiveness instead of permission
5. understanding what "NO" means
6. implementing the changes

This paper describes how changes have been implemented in software and software testing at JPL as well as describing the processes that has led to those changes.

I. Introduction

Jet Propulsion Laboratory (JPL) comes up with new and novel solutions to successfully send one-of-a-kind spacecraft to planets, moons, comets and asteroids. Currently JPL is experimenting with small helicopters (figure 1) that will go with rovers to Mars. The small helicopters will survey the area to help direct the rover safely to a location of interest. The helicopter's blades must spin faster than on Earth due to the lower density atmosphere, and have low mass so that it can lift off the surface without much power. Finally, it must perform 2 to 3 minutes of survey daily including the seven seconds of terror for take offs and landings. Two and a half years ago Mars Science Laboratory, Curiosity, landed on Mars using the sky crane harness (Figure 2) to actually put a "Mini-Cooper" sized rover on Mars. The sky crane was an innovative approach to landing and generated the video "7 Minutes of Terror" that can be seen on www.youtube.com . All of these innovations would seem to make initiating change a common place occurrence at

JPL. However, the innovations that have been discussed are all hardware. When it comes to software development, testing, and processes, change is more difficult because it is necessary to ensure that the spacecraft will not be harmed by the commands that are sent to it.

There are approaches that have been successful in instigating change in these areas in this risk averse environment and they will be discussed in this paper. The examples will be related to software development and testing processes. This paper describes two different examples that have been successfully implemented. One deals with using existing software in a new way and the other is a new testing development process.

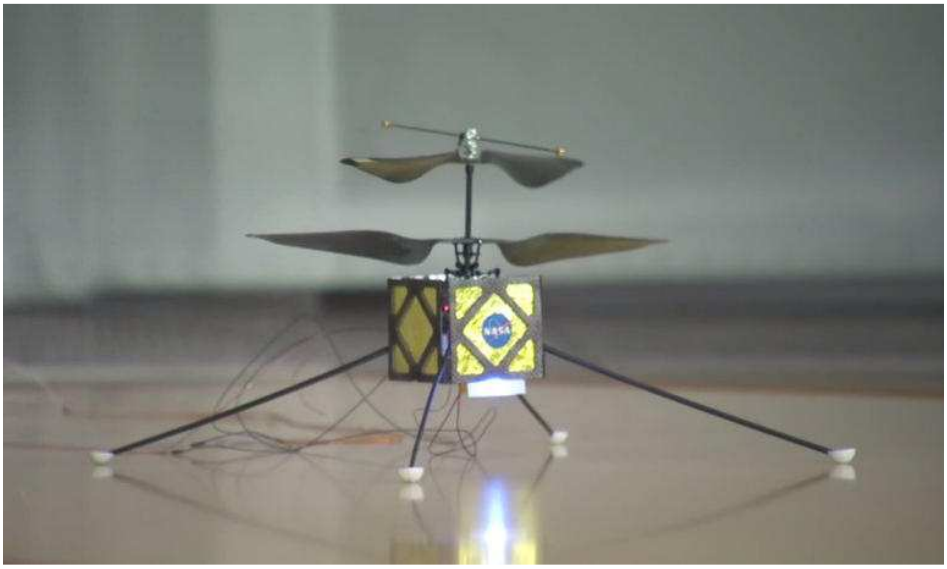


Figure 1: Prototype Mars Helicopter


Figure 2: Mars Science Laboratory (Curiosity) Sky Crane

II. Understanding the Business Use Cases

Perhaps the first step in instigating change is to understand the characteristics that are important to the business. For some companies profit margin is the most important factor. With companies that deal with the public, customer satisfaction is a significant factor since this affects profit margin. In other companies such as air traffic control or power generation, public safety is of major importance. By understanding what is important to the company, a business case can be made for change. For JPL, spacecraft safety is an imperative, because JPL sends expensive one-of-a-kind spacecraft to places that are not yet accessible by humans. These spacecraft are used to collect valuable science data about the solar system. To collect science data the spacecraft must remain healthy and ground software simulators must identify possible commanding errors. JPL has started to explore using a new type of smaller spacecraft called CubeSats. These spacecraft are comprised of "units" that are 10x10x11 cm cubes that are put together by using 1 or more cubes. A one unit (1U) CubeSat is only 1 cube. Standard configurations are 1U, 2U, 3U, 6U or even 12U. Figures 3 and 4 show a 6U CubeSat called Mars Cube One (MarCO) that will demonstrate CubeSat operation in deep space. CubeSats are simple spacecraft in comparison to spacecraft like the Mars Curiosity Rover (Mars Science Laboratory) or the Cassini Saturn orbiter. They are less expensive and commands are validated and verified on an engineering unit in a lab on Earth rather than through software simulation. However, it is still necessary to ensure that the commands to be sent to the spacecraft and their corresponding parameters are valid. If there is inexpensive and easy-to-use software to perform this task, it can be used to help ensure command integrity.

The Multi-Mission Planning and Sequencing (MPS) group provides much of the software that is used to develop and package the commands that are sent to spacecraft. The software is complicated because the spacecraft command sequences can be complex. A study recently sought to understand the needs of the CubeSat community and to provide the teams of these small spacecraft with out-of-the-box software to create command sequences. The study was funded because MPS came up with a new business arena for the Multi-Mission software. Before the proposal to perform the study, the team got together to discuss ideas and new approaches.

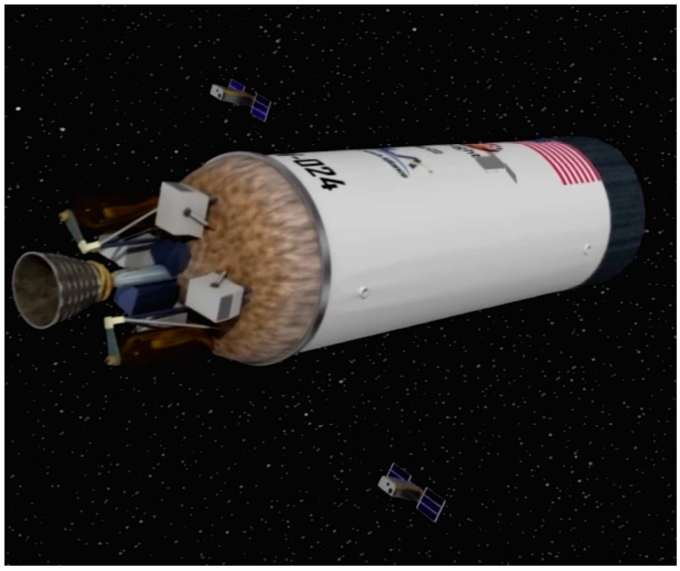


Figure 3 Mars Cube One (MarCO) 6U CubeSat illustration with InSight Mother Ship

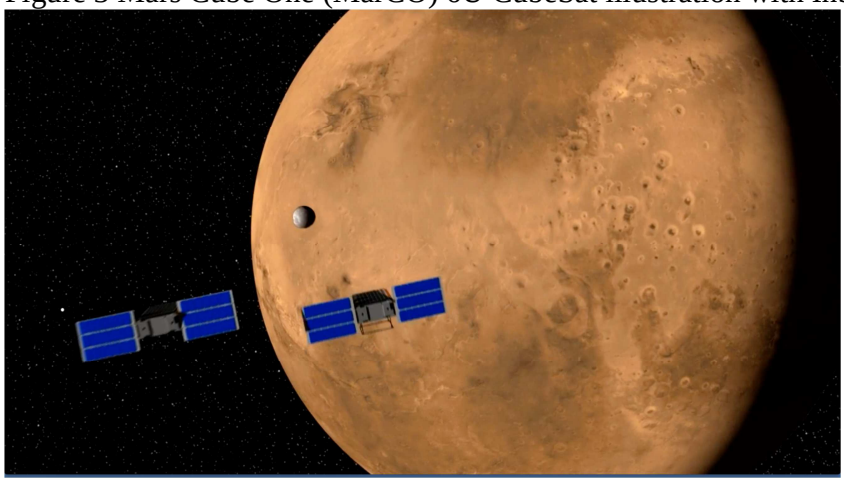


Figure 4 Mars Cube One (MarCO) dual 3U CubeSats Going to Mars

III. Brainstorming Ideas

Making changes in software development and testing processes usually starts with ideas from one or more team members who work with the software in some way. Some companies including Google and 3M have their workers dedicate a certain percentage of their time to developing new ideas. At JPL, people have ideas all the time: some of them are good, some are o.k. and some of them are not workable. Brainstorming ideas helps refine good ideas and modify o.k. ideas or not workable ones into good ones. Brainstorming helps to see an idea from fresh perspectives and determine if it is feasible. Brainstorming an idea inside one's own head first helps to refine the concept and how to communicate with others. Thinking about an idea and how it will work typically translates into the development of operational use cases. With the operational use case and overall concept in hand, it can be easily communicated to others and

help to promote buy-in. As the concept is discussed, and questions asked and answered, the original concept matures and becomes more viable and eventually a fully developed concept. To make changes happen, it is generally best to have this well-thought out concept.

The MPS team met to discuss how to approach using existing MPS software for CubeSat missions. It was realized that an MPS specialized editor can be used in a simple mode that fits the needs of the CubeSat community. The next step was to propose a study to discover more about CubeSats to determine what needed CubeSat capabilities were missing in the MPS software.

Another example involves software testing where MPS found that software testing was often left to the last minute. For the test team it was difficult to determine the new software capabilities that were being delivered. Determining those new capabilities and writing tests for them was risky to do at the last minute. The group had recently begun using the Scrum agile process and one of the testers came up with the idea to add a test sub-task to each new capability or bug fix task. The test tasks are scheduled the same way that the coding tasks are scheduled. In this way the test team can begin to develop the automatic tests at the same time that the developer is implementing the new capability or bug fix. It allows for continuous exploratory testing. Figure 5 includes two "Verify" test sub-tasks associated with a coding refactoring parent task and a coding bug fix parent task.

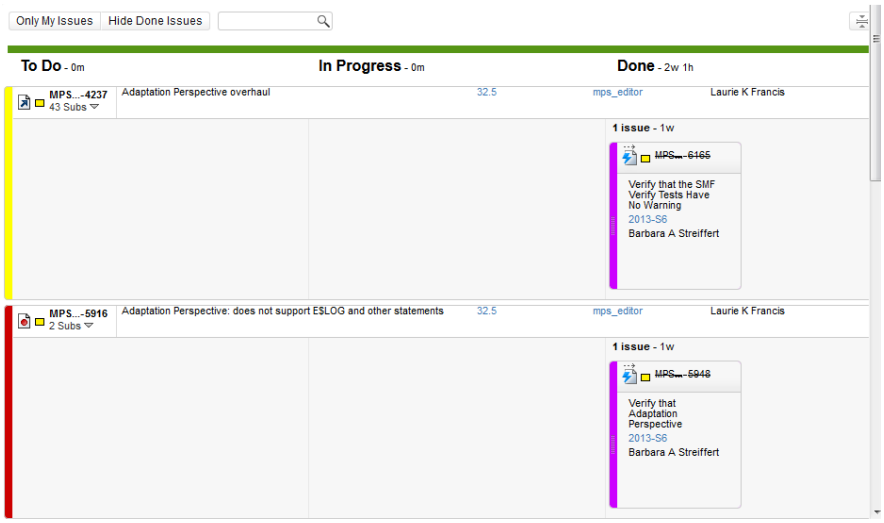


Figure 5 Test tasks associated with coding implementation tasks

IV. Performing changes incrementally

Incremental changes are sometimes easier, more cost effective or have less schedule impact and can allow the implementers to understand additional improvements. Often managers believe that incremental change involves less risk and are more open to making the changes. Small changes are almost always done at one time, but larger changes can involve an incremental approach.

When a change can be broken down into smaller steps, it makes sense to implement the change incrementally. The CubeSat study has been proposed to be performed in three steps:

1. An initial high level study to determine the basic needs of the small spacecraft
2. An in-depth study to look at the individual needs of the small spacecraft and build a prototype using existing software
3. Look at future needs of more complicated small spacecraft and prototype those needs using existing software

Figure 6 is a sample of the quad chart proposal for the CubeSat study (note: proprietary information has been removed). The initial study has been funded and interviews of CubeSat mission personnel are underway.

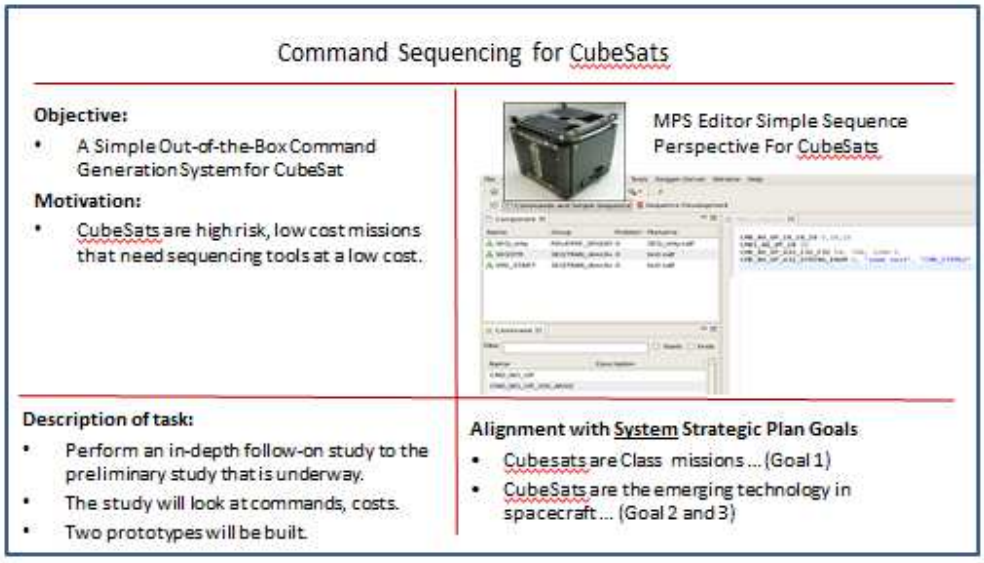


Figure 6 A Quad Chart for the CubeSat study

Sometimes incremental change happens unexpectedly. The change that added the “Verify” test sub-task into scrum process ended up being able to be used for another purpose. Software Quality Assurance (SQA) asked for the changes in the automatic tests that verify each bug fix to be documented. The test team came up with the idea to use the “Verify” test sub-task to describe the changes to the automatic tests. In this way the “Verify” test sub-task concept was repurposed so that it provided two capabilities. The change was implemented incrementally by creating a dual purpose for the original change.

V. Obtaining approval or asking for forgiveness instead of permission

It is often difficult to decide when to make proposals, how to make them, and who to make them to. There are several conditions that can govern these aspects of change. If the change involves a single person and if this person can make the change and still meet his/her schedule and responsibilities, then the change can happen as soon as the person wants to implement it. However, new ideas/changes generally involve one or more of the following:

1. Time to implement the change
2. Others are impacted by the change
3. Deliveries are impacted by the change
4. The implementation of the change costs money or resources
5. The change involves risk

When one or more of these aspects is impacted, then typically the person who is interested in making the change must make a proposal in order for the change to take place. The request/proposal is generally made to closest level of management to the change unless the company has specific times set aside for making proposals to higher level managers. Often it is necessary to make the proposal to multiple managers. In the case of the CubeSat study the proposal has been made to Multi-Mission Ground Systems and Services (MGSS) management since they are the funding source for MPS, and at JPL changes require approval of resources. MGSS must provide funding for something that will take a year to implement. On the other hand the "Verify" test task has been a change that could be implemented easily and even though others in MPS have been impacted by the change, the impact has been minimal. This change only required the MPS manager to approve it. However, sometimes it is necessary to implement part or all of the change to show managers that the change is beneficial. In those cases people have used their own time or short lulls in their workload to implement the change. Recently MPS started using GIT as their repository. The test team had not used a repository in the past, and several aspects of the testing framework needed modification to fit into the structure of the MPS implementation of GIT. One member of the test team, in-between various testing tasks, made the required structure changes. He did not make a proposal nor ask for permission. He made the needed changes and told everyone afterward. In this case nothing was impacted negatively and

the test framework can now be put into GIT – a definite positive outcome. If a change has not been approved through some sort of approval process, then probably the safest approach is to implement the change on one's own time making sure that the change can be reversed if not successful. Perhaps using a playground area is useful when making any type of change whether it is approved or not.

VI. Understanding what "NO" means

Sometimes a proposal is not successful and the submitter is told that he/she is not allowed to proceed. In most cases it is beneficial to find out the reason why the proposal was not successful. In other words 'what did "NO" mean'.

It could mean any one (or more) of the following reasons:

- The manager did not think the proposal would succeed.
- The manager thought the proposal would take too much
  - Time
  - Money
  - Personnel
- Two or more of the above
- The manager thought the proposal was too risky
- The manager thought the proposal did not meet future company needs
- The manager thought the proposal was not aligned with the company's goals
- The manager thought the proposal was not a good idea, well-written, or well-researched
- The manager thought the proposal contained inadequate justification
- The manager didn't understand the proposal
- The manager thought the proposal should be handled by another department
- The manager fell asleep during the proposal because of an overnight flight
- …

There are many managers and just as many reasons why a proposal doesn't succeed. It is important to find out why a proposal is rejected and apply lessons learned to the next attempt. If the proposer has socialized the proposal, by getting as many valid opinions as possible, and has answered the questions that those people raise, then there are usually fewer reasons for the proposal to be rejected. Being rejected is part of the process, and is a learning opportunity. Proposal writing like anything else requires practice. Learning about the audience who will evaluate the proposal is also invaluable. Of course if the proposal is selected, successfully implementing it is equally as important.

VII. Implementing the changes

Once the proposed change has been funded or time allocated to perform the change, work can begin. Building a track record of successfully implementing proposed changes is important in building confidence in the proposer. The following are suggestions to help make the proposal successful:

1. Make plan for the work. If the work has a final report, create an outline for the report

2. Make a realistic schedule with deliverables

3. Provide management with the plan and the schedule

4. Divide the work into small chunks so that accomplishments/progress can be reported

5. Provide status to management at least weekly. (Note: the status does not have to be lengthy, but should show progress and obstacles)

6. Provide demonstrations at appropriate intervals. Usually demonstrations are only performed when the work has some maturity.

For the CubeSat study an outline for the final report has been made and is Figure 8. Status has been reported weekly. Figure 7 is the template of the status report. A demonstration of the prototype work will be scheduled at the end of the task. The study has been assigned as an additional task and fits in-between the normal work of testing deliveries. It has been important to let management know the work that has been completed in-between the deliveries.

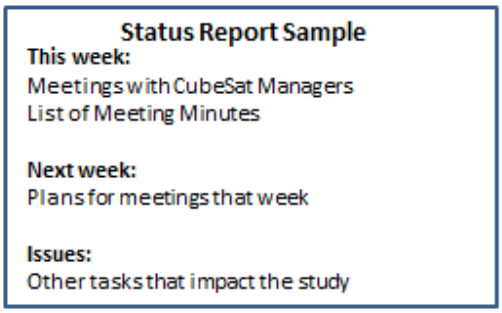


Figure 7 A Sample Status Report

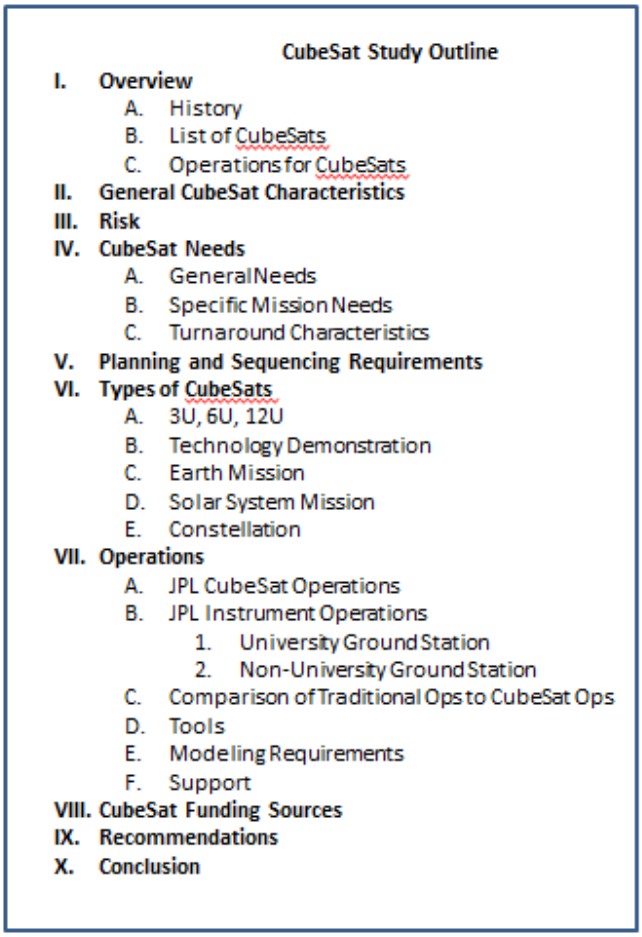


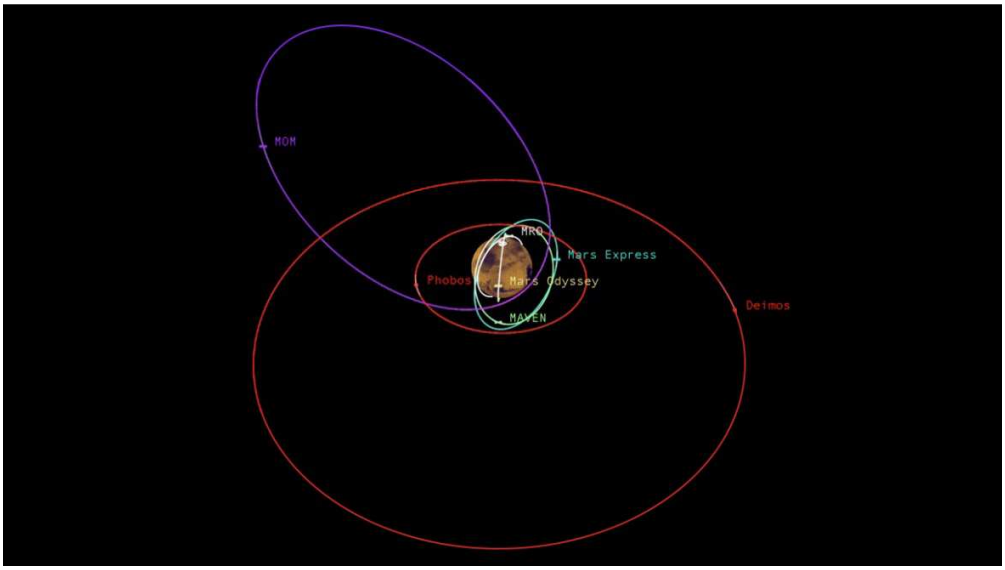
Figure 8 CubeSat Study Final Report Outline

VIII. Conclusion

All of the suggestions in this paper are common sense. To recap the following are the suggestions for making change happen:

1. understanding the business use cases
2. brainstorming ideas
3. performing changes incrementally
4. obtaining approval or asking for forgiveness instead of permission
5. understanding what "NO" means
6. implementing the changes

Each of the examples in the paper has used these suggestions in order to instigate change. This process has been successful in creating change for software development and testing processes in the risk averse JPL environment. Perhaps the most important aspect for fomenting change is the individual – change occurs when the individual proposes changes and implements them. Andy Warhol said, "They always say time changes things, but you actually have to change them yourself."



Traffic around Mars: Phobos and Deimos (Mar's Moons), Mars Express (European Space Agency), Mars Orbiter Mission (MOM—India), Mars Express, Mars Odyssey and Mars Reconnaissance Orbiter – MRO (NASA) and soon InSight Lander with Mars Cube One (MarCO) twin CubeSats. (Rovers on Mars are Opportunity and Curiosity)

IX. Acknowledgments

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work is funded by Multi-Mission Ground Systems and Services (MGSS).