Cassini's Test Methodology for Flight Software Verification and Operations

Eric Wang[1] and Jay Brown.[2]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 91109

The Cassini spacecraft was launched on 15 October 1997 on a Titan IV-B launch vehicle. The spacecraft is comprised of various subsystems, including the Attitude and Articulation Control Subsystem (AACS). The AACS Flight Software (FSW) and its development has been an ongoing effort, from the design, development and finally operations. As planned, major modifications to certain FSW functions were designed, tested, verified and uploaded during the cruise phase of the mission. Each flight software upload involved extensive verification testing. A standardized FSW testing methodology was used to verify the integrity of the flight software. This paper summarizes the flight software testing methodology used for verifying FSW from pre-launch through the prime mission, with an emphasis on flight experience testing during the first 2.5 years of the prime mission (July 2004 through January 2007).

Acronyms

AACS	=	Attitude and Articulation Control Subsystem
AFC	=	AACS Flight Computer
ATLO	=	Assembly, Test, and Launch Operations
CATS	=	Cassini AACS Test Station
CDS	=	Command & Data Subsystem
FP	=	Fault Protection
FSC	=	Flight Software Change
FSDS	=	Flight Software Development System
FSTB	=	Flight Software Test Bed
FSW	=	Flight Software
ITL	=	Integrated Test Laboratory
JPL	=	Jet Propulsion Laboratory
MRO	=	Memory Readout
MTA	=	Mono-propellant Tank Assembly
RCS	=	Reaction Control System
RWA	=	Reaction Wheel Assembly
SFP	=	System Fault Protection
SOI	=	Saturn Orbit Insertion
ΔV	=	Change in Velocity

[1] Software Engineer, Guidance and Control Software and FSW Testing, M/S 230-104, 4800 Oak Grove Dr., Pasadena, California 91109. Email: eric.wang@jpl.nasa.gov
[2] Sr. Software Engineer, Guidance and Control Software and FSW Testing, M/S 230-104, 4800 Oak Grove Dr., Pasadena, California 91109. Email: jay.brown@jpl.nasa.gov

I. Introduction

THE Cassini mission description and spacecraft design concepts have been well documented and can be referenced[1]. The mission to Saturn and Titan has encompassed over twenty-five years of planning, design, development, testing and flight operations. Figure 1 provides the Cassini mission time line from the launch phase to the end of cruise phase. The Attitude and Articulation Control Subsystem (AACS) Flight Software (FSW) implementation started in 1993. The launch FSW load was completed in 1997 with a successful Cassini launch on 15 October of the same year. This however was not the end of AACS FSW development nor was it the end of testing. The roles of testing and the methodology used to address testing during all milestones of a spacecraft flight project are keys to the success of the project and mission. Testing will follow the software development cycles and is constantly intertwined during a project's inception to completion. However, it is also important to emphasize that testing must lead during anomaly investigations, prototyping, trade studies, verification and validation of flight software and sequence design.

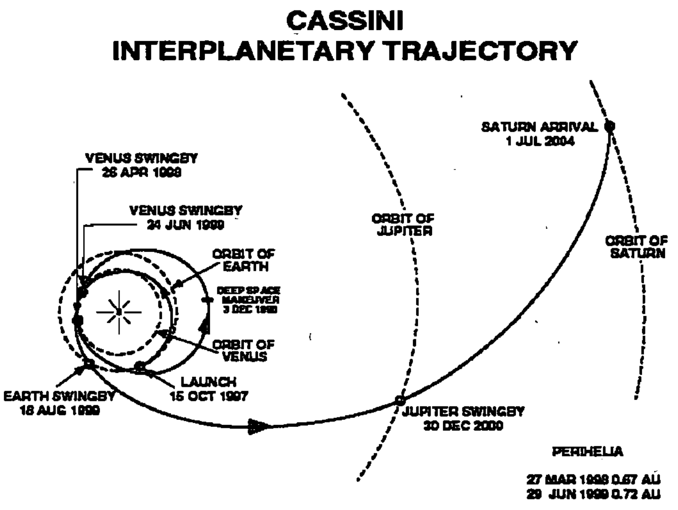

Figure 1. Cassini Spacecraft Mission Timeline. *This time line describes the Cassini trajectory to Saturn from its launch to the beginning of the prime mission.*

For Cassini, the test methodology established during the pre-launch phase of the mission standardized the baseline testing performed during post-launch and operations phases. Planning for nominal and anomalous scenarios during the operations phase provided opportunities to enhance the testing philosophy to account for additional testing methods. The continual monitoring of spacecraft performance provides the opportunities to update the FSW to meet the changing needs of the mission. Each and every change to FSW must be subjected to the same overall test philosophy to ensure mission success.

This paper describes the test methodology utilized throughout the Cassini AACS FSW design, development, and operations. The focus will be on the prime mission operations test methodology for FSW verification. The level of detail for pre and post launch methodologies will be limited to a brief high-level description to establish test milestones and baseline philosophies. The pre-launch mission phase encompassed preliminary FSW design to spacecraft launch.

II. Pre-launch AACS FSW Test Methodology

Cassini FSW development and testing for the AACS has kept to strategies described in JPL's Software Development Requirements plan and Flight Project Practices. Moreover, Cassini FSW development and testing spanned over six years for pre-launch and another 6.7 years for post-launch development to focus on SOI and the Huygens Probe release and relay critical sequences. Currently, modifications to AACS FSW are ongoing to meet current mission objectives for the prime and extended mission phases. The pre-launch methodologies will reveal the test philosophy used to validate the software used in one of the most successful inter-planetary missions. Overall, the software test process contained and adhered to the following key milestones:

1) Requirements trace matrices development: To ensure that requirements are captured in testing.
2) Test plan development: To ensure that FSW requirements are testable and that test methods, strategies, and environments are identified.
3) Test case design: To ensure test designs and compatibility with test environments.
4) Test trace matrices development: Tracing test cases to requirements to ensure full test coverage.
5) Test case and procedure development: Develop test cases and pass/fail criteria to ensure proper test coverage.
6) Test readiness review: Review of test environments, procedures, test cases, and personnel to ensure that testing can meet timelines and requirements.

7) Test case execution: To ensure that testing is accomplished in the desired test environment.
8) Test anomaly reporting: To ensure that anomalies are properly documented.
9) Test analysis: To ensure that test results met pass/fail criteria, fulfill trace matrices, suggest workarounds and updates to Test Trace Matrices.
10) Test result reporting: Test results reported and test trace matrices presented and reviewed to ensure proper visibility.

The Cassini unit test plan described the methodology in detail on how to test FSW functions, modules, and objects. Test execution, test set-up, hardware and software environments, and test cases were all identified in the unit test plan. AACS control architecture was broken down into objects which included commanders, controllers, estimators, and hardware mangers. The FSW was mode and state driven which established the method for how unit testing was performed and what the verification methods would be for pass/fail criteria. Unit testing always involved low-level verification of functions, monitoring state or mode transitions, and boundary condition testing. Future FSW interfaces were stubbed to allow the proper verification of each unit test. Interfaces were un-stubbed once they became available. Analyzing the scope of each unit test was also vital in determining if the proper test methodology was exercised. All unit test results were peer reviewed for content, functionality validity, and results.

The Cassini AACS Software Test Plan was broken down into integration and acceptance test planning, test requirements, test procedures, and test reports. The focus was on testing interfaces between objects and requirement conditions specified in the Software Requirements Document, Software Specification Document, and Failure Mode and Effects Analysis. Requirements could then be traced to the specific scenario or function tests. Integration testing was tied to the planned number of FSW builds.

At the completion of each integration test phase, an integration test baseline was established. The baseline was configuration controlled which included tested software, test drivers, and test stubs. Regression testing consisted of rerunning selected unit test for the objects and rerunning selected scenario and/or functional tests from the integration testing.

Acceptance testing started with the verification of the Requirement Test Matrix to the requirements for FSW. Any requirements not validated by scenario testing had one or more functional tests developed to validate them. The final approval of acceptance testing was the successful completion of scenario and functional testing. Just as important as testing were the reviews for test readiness and reviews of test results. These reviews established the readiness of ground and flight support, facilities, plans, processes, procedures, and verification & validation status to meet the mission objectives.

III. Test Environments

There were three main test environments that were utilized during testing, verification, and validation: FSDS, FSTB/CATS, and pre-launch Assembly, Test, and Launch Operations (ATLO) which became the post-launch Integration Test Laboratory (ITL).

A. FSDS

FSDS is a simulation environment for the Cassini AACS subsystem. It provides the user a command line interface for visibility into the spacecraft simulation, the flight software, and the ground interface. Furthermore, FSDS is a high fidelity, faster than real-time, subsystem-level simulation test environment. FSW and simulations coexist on the same UNIX platform. The role of the FSDS test bed simulator throughout the Cassini mission has been well published.[2]

Over the years, FSDS has performed extremely well supporting Cassini operations. However, in order to facilitate additional new testing in support of the critical events such as SOI, "superscript" was created. Superscript is an extension of the AACS FSDS with a simulated system fault protection logic engine and a simulated critical sequence engine (with mark point and rollback capabilities). It was initially developed to test the full functionality and interaction for SOI, Probe Release, and Probe Relay.

B. FSTB/CATS

Flight Software Test Bed was used during pre-launch testing. It consisted of a Virtual Machine Environment (VME) chassis with Tasco PACE 1750A commercial boards for FSW, Heurikon 68040s for the hardware modes and interface for the dynamics simulation, memory, and reflective memory boards. A Sun Sparcstation was used for the star tracker simulation, and Ethernet boards were utilized for communication. FSTB provided a more realistic simulation test bed because FSW and simulations were running on separate computers. During post-launch testing, it was replaced with the Cassini AACS Test Station (CATS). This provided a real-time subsystem level test environment.

C. Pre-launch ATLO to Post-launch ITL

The pre-launch system mode test environment was ATLO. After launch, ATLO was replaced by ITL. As a result, ITL became the system and sub-system mode hardware-in-the-loop test bed for post-launch FSW testing.[3] ITL is a high fidelity real-time system-level test environment. The Cassini project emphasizes the importance of having a ground test bed system that is identical to that of the spacecraft. It is imperative that the system configuration on the ground matches the configuration on-board the spacecraft. Thus, ground test bed system such as ITL must reflect the actual spacecraft's on-board software and hardware configurations and update them accordingly when different. The use of ITL test bed system for operation testing is vital in identifying potential problem(s) that may occur in-flight. ITL is composed of AACS hardware and software, AACS support equipment and software, Command and Data Subsystem (CDS) hardware and software, CDS support equipment and software, Power and Pyrotechnic Subsystem (PPS) hardware, PPS support equipment, a data management system (itlarc), Advanced Multi-mission Operations System Ground System components, which includes Test Telemetry and Command System, Data Monitoring and Display, Telemetry Delivery Subsystem, Telemetry Input Subsystem, and Test Telemetry Input Subsystem. Table 1 describes which test beds contain software simulated hardware models and which ones contain actual or engineering unit hardware.

Table 1. Comparisons between Test Environments

Hardware	ITL System Mode	ITL Sub-System Mode	FSTB/CATS	FSDS
AACS Flight Computer (2)	Real	Real	Real	Sim
CDS Engineering Flight Computer (2)	Real	Sim	Sim	Sim
Accelerometer (1)	Real/Sim	Real/Sim	Sim	Sim
Inertial Reference Unit (2)	Real/Sim	Real/Sim	Sim	Sim
Reaction Wheel Assembly (4)	1 Real + Sim	1 Real + Sim	Sim	Sim
Backup ALF Injection Loader (1)	Real	Real	Real	Sim
Power & Pyrotechnics Subsystem	Sim	Sim	Sim	Sim
Stellar Reference Unit (2)	Real	Real	Real	Sim
Sun Sensor Assembly (2)	Real	Real	Sim	Sim
Main Engine Valve Driver (2)	Real	Real	Sim	Sim
Engine Gamble Actuator (8)	2 Real + Sim	2 Real + Sim	1 Real + Sim	Sim
Solid State Recorder (2)	Real	Sim	None	None

IV. Post-launch AACS FSW Test Methodology

Per pre-launch plan, two FSW builds were developed during the cruise phase to support the execution of critical sequences. New FSW functionality was unit tested. FSW went through sub-system and system level testing before being uplinked to the spacecraft. Uplink Readiness Reviews were the major milestones to approve FSW for uplink. In these reviews, the FSW development status, FSW testing (both on the sub-system and system level), FSW test reports, and uplink procedures were presented and reviewed.

1) Builds A7 (Cruise): These builds contained FSW fixes and updates to support the seven-year cruise to Saturn. Regression, scenario, and functional testing were performed to verify and validate changes to FSW.
2) Builds A8 (Critical Event Sequences and Prime Mission Support): A8 was divided into several planned version releases. A8.6.7 supported Saturn Orbit Insertion, A8.7.1 supported Probe Release and Probe Relay, and A8.7.2 (and its variants) supported the remainder of the prime mission. The same A7 build test philosophy was followed. However, additional testing was focused on the critical events of SOI and Probe Relay. Several papers have been written on the efforts of FSW development and testing for these events.[4-6]

The flight software test methodology for post-launch activities followed the pre-launch methodology. The method consisted of incrementally increasing complexity and functionality through the phase of integration testing for each build, which culminated with scenario and functional testing of the complete set of software. To support the post-launch stage of the mission, additional testing had to be considered for full and partial FSW uplinks and contingency planning. Testing and validation of these activities became critical additions and major foci for the operations testing philosophy.

V. Prime Mission AACS FSW Operations Test Methodology

The prime mission of Cassini started in July 2004. Figure 2 illustrates the first 2.5 years of the prime mission's Saturnian trajectory. Although there are many different methodologies in testing, the pre/post-launch test practice has been embraced as the general common test methodology for ensuring the health and safety of the spacecraft. Along with a common testing practice, reusable test environments have also been practiced. Test bed environments have undergone several changes over the years to incorporate new and improved capabilities. Simulation, analysis, and comparison for each test are performed and used as the basis for validating changes in the software and testing processes before the product is sent to the spacecraft. System testing first occurs in the development environment but eventually is conducted in the operation environment. Functionality and performance testing are designed to catch bugs in the system, unexpected results, or other behaviors in which the system does not meet the stated requirements. FSW and FP engineers create detailed simulation scenarios (stress tests) to test the strength and limits of the system by attempting to break it when possible. The testing provides reviews of the uplink products and procedures to not only correct typographical errors and incorrect file contents, but also improves the system's overall process and usability for future use.

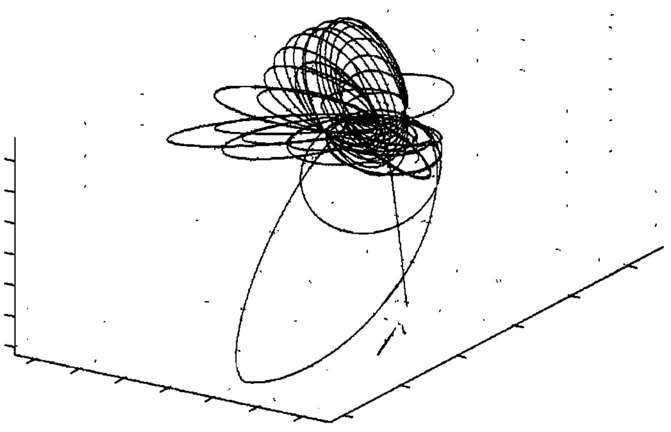
Figure 2. Cassini Spacecraft Prime Mission Timeline. *This time line describes the first 2.5 years of the Cassini prime mission Saturnian trajectory. The Cassini image marks the approach to Saturn.*

Although FSW uploads have occurred throughout various stages of the mission, its upload philosophy has evolved to some extent during latter part of the mission. The overall perception on what goes in the FSW load has changed since entering the prime mission phase. Even with a revised philosophy on all future FSW loads, the test methodology still remains the same. With all the major FSW development occurring during the pre-launch and cruise phases of the mission, full FSW uploads have become virtually non-existing in the prime mission. Project management has adopted a more conservative approach and avoided changing the FSW logic/code unless deemed absolutely necessary. Any FSW logic code change will require a full flight software upload. All executed FSW updates in the prime mission have been parameter updates. As a result, the FSW patch load has become the choice method for maintaining the health and safety of the spacecraft. Three major reasons for the project's conservative approach are:

1) AACS FSW is performing nominally – Prior to the start of the prime mission, FSW A8.7.1 was uploaded in preparation for the critical activities. To date, the spacecraft is operating nominally.
2) Prevent the chance of having any FSW oversight – FSW patch load vs. full FSW load.
3) Avoid the occurrence of AACS flight computer resets – In order to lessen the chance of an anomaly and lessen the impact to sequence operations, the prime AFC should not be reset unless necessary. A full FSW upload would have to reset the prime AFC.

However, even with a revised philosophy in FSW updates, FSW testing is still required. No matter whether it is a full FSW upload or a FSW patch load, every step of testing must be performed. In addition, reviews for test readiness and test results must be conducted. These reviews are required to establish the readiness of ground and flight support, facilities, plans, processes, procedures, and verification and validation. The operations test philosophy can be broken down into five major categories:

1) First-Time Events
2) FSW Patches
3) FSW Patch Loads
4) Sequences
5) Anomalies

A. First-Time Events

In order to ensure the spacecraft's health and safety during first-time events, extensive and thorough testing must be performed to identify all issues that the spacecraft may encounter. There are five major first-time events during the prime mission: SOI, Probe Release, Probe Relay, Mono-propellant Tank Assembly (MTA) recharge, and low-altitude Titan flybys. Due to the extent of the testing that was required to perform hundreds of test scenarios within a tight time schedule, FSDS and Superscript were the primary test bed environments utilized in verifying all the critical events such as SOI. Due to this substantial number of test cases that must be performed within the given time constraint, it was essential that a set of pass/fail criteria was established first before continuing with the testing. Without having a well-defined set of pass/fail criteria, there would be little direction as to how the tests were to be performed and be analyzed. With pass/fail criteria defined, regions of vulnerability were meticulously identified and selected to effectively test various fault scenarios. The design and approach for testing Probe Release and Probe Relay are illustrated in Fig. 3 and 4, respectively.

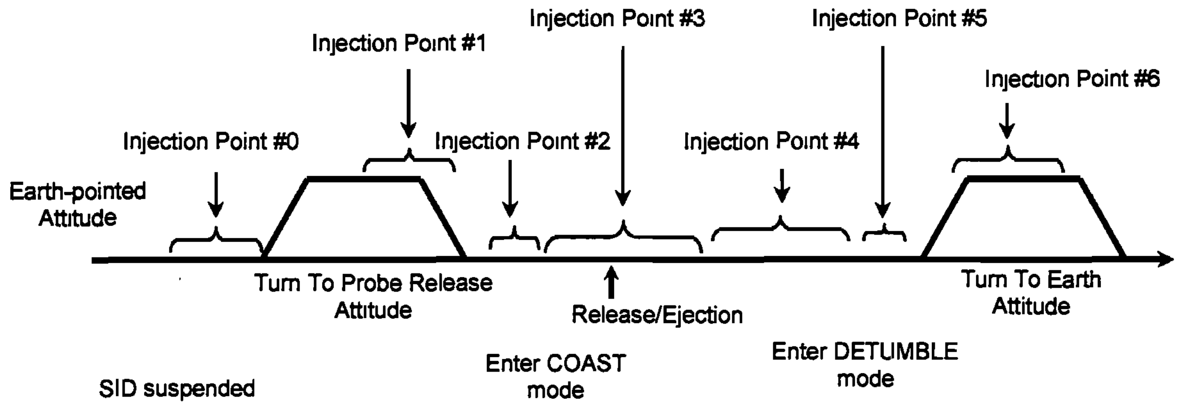

Figure 3. Strategic Probe Release Testing Approach. *The figure captures the spacecraft turn profile and sequence of events for the Huygens probe release. Injection points indicate test areas of interest for fault injection scenarios.*

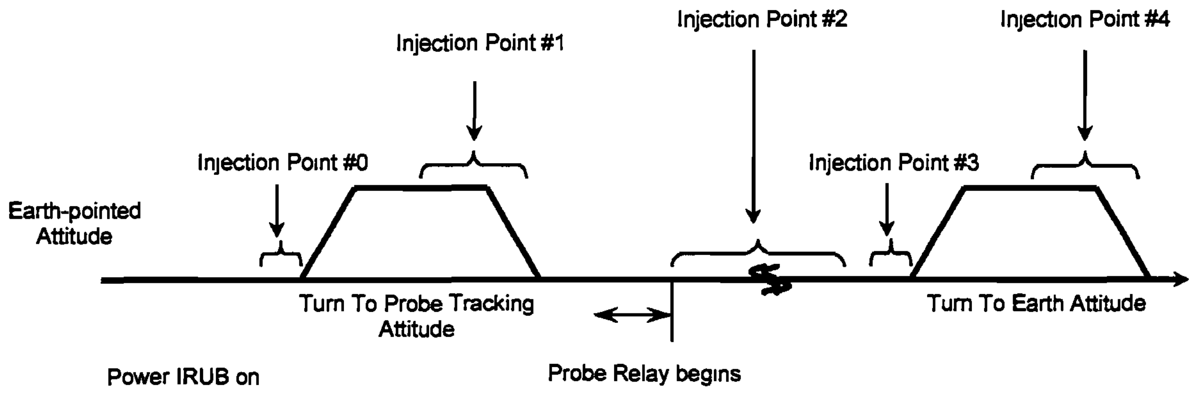

Figure 4. Strategic Probe Relay Testing Approach. *The figure captures the spacecraft turn profile and sequence of events for the Huygens probe relay. Injection points indicate test areas of interest for fault injection scenarios.*

There were twenty-four Titan flybys during the first two-and-a-half years of prime mission. Out of these twenty-four Titan flybys, seven were classified as low-altitude Titan flybys. Furthermore, each Titan flyby is different in flyby altitude, spacecraft orientation, and science observation. In order to support the simulation of these low-altitude Titan flybys, both FSDS and ITL had to be modified to add in a Titan atmospheric model. Many tests were performed to verify and validate the atmospheric models before testing the science sequences of these Titan low-altitude flybys. Test data from FSDS were also used to support the reconstruction of the atmospheric density of Titan.[7]

The MTA recharge was executed in April 2006. It was the first and only time that the spacecraft was to perform re-pressurization of its hydrazine tank. The MTA recharge brought the spacecraft's thrusters up to a higher thrust level to ensure safe low-altitude Titan flybys. After the MTA recharge event, the FSW was updated to match the new thrusters' thrust level. While some of the MTA recharge activities were part of FSW patch load verification process, a majority of the MTA recharge activities involved contingency testing using both FSDS and ITL test bed environments. Regions of vulnerability were identified and tested by inserting fault cases to simulate fault scenarios during the MTA recharge simulation.

B. FSW RAM Patches

FSW RAM patches are primarily used for updating certain FSW parameters that cannot be changed using pre-defined commands. Instead, these FSW parameters can only be updated by sending memory-write commands to the AACS flight computer. The memory-write command will directly override the previous value at the intended memory location with the new value. FSW parameters are categorized into two groups: variables and constants. Memory write command shall only be used when updating variable values. Changing a constant will result in changes in the FSW checksum. Hence, when performing FSW patch updates, it is imperative that the intended FSW parameter is a variable and not a constant. FSW RAM patches are used to improve the accuracy of the data and maintain the health and safety of the spacecraft. Obsolete FSW parameters in the system could lead to the triggering of spacecraft anomalies. With the updated FSW parameters, spacecraft anomalies are prevented and the health and safety of the spacecraft is preserved. At times, during expected events, fault protection related FSW parameters are patched to prevent FP activities such as relaxing FP operational thresholds due to aging equipment or known events which may unnecessarily trigger FP. All memory write commands are always tested thoroughly in ITL before they are sent to the spacecraft. When performing verification of the memory-write commands, Memory Readout (MRO) commands are always sent to capture both the pre-patch and the post-patch values. Whenever appropriate, modified FSW parameters are also incorporated into a check-out sequence and the sequence is executed in-flight to verify FSW integrity.

C. FSW Patch Uploads

There were three FSW uploads, A8.7.2, A8.7.4, and A8.7.5, during the first 2.5 years of the Prime Mission. All these FSW uploads involved updating the FSW parameters that were constants. As a result, the FSW patch load process was used in order to update the FSW checksum that corresponds to the FSW constant parameter changes. For each FSW patch upload, extensive testing on the ground was required for software verification. In order to validate the validity and the reliability of a new FSW load, three well-defined types of testing were and still are performed for software verification.

1) Unit Testing/FSC Testing
2) FSW Regression Testing
3) System-level Testing.

The unit testing is conducted during the early stage of the development phase in the software cycle. These test cases are created to test each object's specific functionality individually or with other units. However, unit testing is designed to cover small area of functionality rather than the system as a whole. This allows the flight software engineers to conduct a swift first round testing to eliminate any software errors before advancing to the next stage of testing. During the prime mission, unit testing is replaced by Flight Software Change (FSC) testing for more selective functional testing. Like unit testing, FSC tests are created to test the object's specific functionality individually. The FSW patch load involves only parameter changes in the corresponding FSW objects. Instead of performing unit testing on all the objects, FSC test cases are uniquely created to only test the affected objects.

When performing software verification for a FSW patch load, a standard set of regression tests is performed. Regression testing is conducted to verify the functionality of the latest FSW build using the FSDS simulation test bed. Its test suite is composed of a collection of Tcl test scripts created during the original software development and shortly after launch. The execution is driven by the FSDS with commanded sequences and by fault-induced predetermined scenarios. These tests are utilized to catch and identify any unexpected errors or events caused by the new software changes. The goal is to validate the interaction between the Flight Software and FSDS/ITL using the following four sets of test cases:

1) Functional Regression Tests (performed using FSDS): The functional regression tests are intended to verify the spacecraft's AACS capabilities (RCS and RWA attitude control, RCS and Main Engine ΔV, low altitude Titan flyby, etc.) and to catch undesirable changes or interactions in FSW.

2) Backup-RAM Regression Tests (performed using FSDS): The backup-RAM regression tests are intended to verify the basic functionality which validates the execution of fault protection commanding, telemetry, as well as the repair, isolate, and get and replacement of peripheral assemblies.
3) Fault Protection Acceptance Tests (performed using FSDS): The fault protection acceptance tests are intended to validate backup AACS Flight Computer commanding, telemetry messages, state table data, service requests, and heartbeat messages.
4) Worst Case Timing Tests (performed using CATS or ITL utilizing hardware-in-the-loop): For every FSW build, the worst-case timing test is performed in CATS or ITL subsystem-mode. The worst-case timing simulations were carried out to provide a more in-depth verification of the adequacy of the Remote Terminal Input/Output Unit design. The simulations were used to provide verification of appropriate timing within the Remote Terminal Input/Output Unit and also of appropriate functionality.

Over the course of the mission, some of these regression test cases have become obsolete and since retired while new cases have been added as a result of better and improved test bed environments. Ultimately, the successful execution of the regression test suite is vital in validating the correctness of the incorporated changes for each FSW patch build as well as validation that other FSW functionality is unaffected.

The system-level testing is mainly performed using ITL. At the end of the testing phase, a test report is generated to document the results. The readiness of ground and flight support, facilities, plans, processes, procedures, and verification and validation status to meet the mission objectives provide full confidence in the test methodology that has been used since pre-launch.

D. Sequences

During the prime mission, there is an extensive amount of sequence testing needed to verify the correct implementation of commands and activities. A sequence consists of commands for both engineering and scientific activities within a specific time-span. In order to validate its design and commanding, FSDS is used. Any design problems with the sequence will be identified and captured by FSDS. In addition to sequence verification, FSDS simulation of the sequence can provide insightful information on how the spacecraft will react or respond. This information can be proven useful by other sub-systems such as Navigation for data analysis.

E. Anomaly Investigation and Resolution

When a spacecraft anomaly occurs, both FSDS and ITL test bed environments are utilized to get a better understanding on what had happened. In order to understand the anomaly, the event must be duplicated on ground. FSDS and ITL are used to simulate the uploaded sequence design and condition at which the anomaly occurred. Although ITL has a higher fidelity in term of software and hardware configurations than FSDS, the FSDS simulation test runs faster; thus, FSDS can provide a quicker outlook of the event or problem. In addition to provide insight of the anomaly, FSDS and ITL are used to find test candidate solutions through testing.

VI. Lessons Learned

Flight software testing is critical in all development phases of a spacecraft. Without an effective testing methodology, unforeseeable software problems may go undetected and be uploaded onto the spacecraft causing an irreversible effect. Selected lessons learned through testing of Cassini AACS FSW are given in the following sections.

A. Stress Testing

Stress testing is always performed when updating parameter(s) in the FSW. During the initial FSW patch load testing, an issue was uncovered while performing stress tests using the ITL test bed. One of the parameter updates was not robust when the parameter was subjected to extreme scenarios. The parameter turned out to be not robust enough for the time duration of its expected latency in the mission. Additional testing was performed and a new strategy in the approach of updating this particular parameter was established. Consequently, a new FSW patch load was created to address the issues that were revealed during initial stress testing. This proves the importance of having stress testing incorporated in scenario testing to ensure the validity of the FSW changes.

B. MRO Verification

Memory Readout is a method used to check a state of the spacecraft that is not telemetered as well as when performing any RAM patching via memory-write commands, thus, making it an important tool for software verification. Every time a memory-write command is used, the verification practice is always to perform MROs on both the effected and the surrounding variables three times before and after the memory-write command. During the FSW A8.7.5 system-level testing in ITL, an issue with the design of the patch verification in the uplink procedures and files was uncovered. The procedure misinterpreted the address range in one of the MRO verification commands. The intended memory address was not fully verified and only captured a portion of the intended memory region. The procedure was updated. Although this was not a threat to the safety of the spacecraft, unnecessary time and effort by the operations team would have been required for contingency meetings and briefings explaining the issue/problem and additional MROs and command files might be needed, which could cause delays to other spacecraft activities. Thus, it is critical that everything is tested as a whole, from uplink procedures to uplink products, under a test bed environment like ITL in order to flush out any issues.

C. Simulation Environments

Simulation environments such as ITL and FSDS are important. All verification testing are performed using the simulation environments. However, discrepancies in the test results are sometimes attributed by the followings:

1) Simulation test bed environments are not always perfect. Often, they are reliable; nevertheless, there are flaws and limitations to them. It is the responsibility of the tester and the operator to know what the test beds are capable and not capable of doing.
2) Documentation of the test bed design is vital and must be kept up-to-date.
3) Test bed configuration must be consistent with what is flown in space. While attempting to verify one of the FSW A8.7.5 updates, a discrepancy between FSDS and ITL results was discovered. It turned out that the test bed configuration for FSDS was different from that of ITL when the tests were performed. The difference in configuration was due to FSDS's inability to emulate the patch table and memory-write commanding. Subsequent command change provided AACS team with a more robust FSW parameter change design.

VII. Conclusions

Ten years have passed since the launch of Cassini on October 15, 1997. In this time-span, Cassini has undergone countless events from its launch, to its encounters of various planets (Venus, Earth, and Jupiter), SOI, release/relay of Huygens probe, and finally to its quest to observe Saturn and its moons. The smooth operation of the Cassini AACS flight software is one reason for the success of the Cassini-Huygens mission. This paper describes test methodologies that were used to verify the integrity of the AACS FSW builds before they were uploaded to the spacecraft. The AACS team will continue to embrace and practice the very test methodology that it has created and employed since pre-launch to the end of the Cassini mission.

Acknowledgments

The authors thank Casper Sagoian for his contribution of providing in-depth historical knowledge of ATLO and ITL. Cliff Lee provided the prime mission Saturnian orbit trajectory plot, which helps illustrate the amount of time Cassini has gathered unprecedented science data. The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1]Lee, A. and Hanover, G., "Cassini Attitude Control System Flight Performance", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA-2005-6269, San Francisco, CA, Aug. 15-18, 2005.

[2]Brown, J., Lam, D., Chang, L., Burk, T., and Wette, M., "The Role of the Flight Software Development System Simulator throughout the Cassini Mission", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA-2005-6389, San Francisco, CA, Aug. 15-18, 2005.

[3]Badruddin, K., Hernandez, J., Brown, J., "The Importance of Hardware-In-The-Loop Testing to the Cassini Mission to Saturn", *IEEE Aerospace Conference*, Track 12.0503, Paper 1231, Big Sky, MT, Mar. 3-10, 2007.

[4]Lam, D., Friberg, K., Brown, J., Sarani, S., and Lee, A., "An Energy Burn Algorithm for the Cassini Saturn Orbit Insertion", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA-2005-5994, San Francisco, CA, Aug. 15-18, 2005.

[5]Cervantes, D., Badaruddin, K., and Huh, S., "Integrated Testing of the Cassini Saturn Orbit Insertion Critical Sequence", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA-2005-6272, San Francisco, CA, Aug. 15-18, 2005.

[6]Allestad, D., Standley, S., Chang, L., and Bone, B., "Systems Overview of the Cassini-Huygens Probe Relay Critical Sequence", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, AIAA-2005-6388, San Francisco, CA, Aug. 15-18, 2005.

[7]Feldman, A., Brown, J., Wang, E., Peer, S., and Lee, A., "Reconstruction of Titan Atmosphere Density using Cassini Attitude Control Flight Data", *17th AAS/AIAA Space Flight Mechanics Meeting*, AAS-07-187, Sedona, AZ, Jan. 28-Feb. 01, 2007.

End of File