Modernization of the Cassini Ground System

Gus Razo[1] and Tammy J. Fujii[2]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, USA, 91109

The Cassini Spacecraft and its ground system have been operational for over 16 years. Modernization presents several challenges due to the personnel, processes, and tools already invested and embedded into the current ground system structure. Every mission's ground system has its own unique complexities and challenges, involving various organizational units. As any mission from its inception to its execution, schedules are always tight. This forces GDS engineers to implement a working ground system that is not necessarily fully optimized. Ground system challenges increase as technology evolves and cyber threats become more sophisticated. Cassini's main challenges were due to its ground system existing before many security requirements were levied on the multi-mission tools and networks. This caused a domino effect on Cassini GDS tools that relied on outdated technological features. In the aerospace industry reliable and established technology is preferred over innovative yet less proven technology. Loss of data for a spacecraft mission can be catastrophic; therefore, there is a reluctance to make changes and updates to the ground system. Nevertheless, all missions and associated teams face the need to modernize their processes and tools. Systems development methods from well-known system analysis and design principles can be applied to many missions' ground systems. Modernization should always be considered, but should be done in such a way that it does not affect flexibility nor interfere with established practices. Cassini has accomplished a secure and efficient ground data system through periodic updates. The obstacles faced while performing the modernization of the Cassini ground system will be outlined, as well as the advantages and challenges that were encountered.

Nomenclature

Ace	=	Mission Controller
AFS	=	Andrew File System
CM	=	Configuration Management
CPU	=	Central Processing Unit
DOM	=	Distributed Object Manager
DMZ	=	Demilitarized Zone
DTU	=	Desktop Units
FRNS	=	File Release Notification and Exchange
GB	=	Gigabytes
GDS	=	Ground Data System
HP	=	Hewlett-Packard
HP-UX	=	Hewlett-Packard Unix
ICA	=	Inventory Change Authorization
IT	=	Information Technology
JIRA	=	Java EE web-based bug tracking and issue tracking application
JPL	=	Jet Propulsion Laboratory
LDAP	=	Lightweight Directory Access Protocol
LDOM	=	Logical Domains
NFS	=	Network File System

[1] Cassini Adaptation Team Lead, Ground Systems Engineering Section, M/S 230-250
[2] Cassini Downlink Ground System Manager, Ground Systems Engineering Section, M/S 230-250

RMI = Remote Method Invocation
RPG = Remote Partner Gateway
SFDU = Standard Format Data Unit

I. Introduction

THE Cassini/Huygens mission launched on October 15, 1997 and arrived at Saturn in June of 2004. Between launch and arrival, Cassini's cruise phase incorporated four planetary flybys: two flybys of Venus, one flyby of Earth and a flyby of Jupiter. In 2004, Cassini's four-year prime mission phase began. During prime mission, Cassini collected its primary science data of Saturn and many of its moons. The Huygens probe, which was attached to the Cassini spacecraft, collected data from Saturn's largest moon, Titan, by entering its atmosphere and landing on its surface in 2005. At the end of prime mission, from 2008 through 2010, Cassini entered its Extended Mission phase. This phase continued critical events such as radio occultation experiments, which measured the size-distribution of particles in the Saturnian rings. In 2010, the Cassini mission entered the Extended-Extended Mission phase. Finding reasonable gaps of time to modernize the ground system between these mission operation events was a challenge. Updating one tool is a less convoluted task than modernizing an entire ground system infrastructure. Since multiple missions share IT infrastructure resources, other missions' critical events can impede software, hardware, and network upgrades. At JPL, during critical events such as launches, orbital insertions and flybys, missions levy an infrastructure freeze, which prohibits changes to shared infrastructure and services. Although ground system modernization is challenge, it maximizes effective operations of irreplaceable spacecraft hardware.

II. Cassini's Architecture and Technology History

Cassini's Ground Data System (GDS) began official operations in 1997 and was formulated several years before preliminary design, when multimission services were introduced. It consists of the ground-based hardware and software components in a distributed environment necessary to support operations at the Jet Propulsion Laboratory (JPL) along with direct contribution from sites in the US and Europe. Cassini's ground system encompasses what is considered multimission legacy software components. Its original hardware was mainly RISC processor based Hewlett-Packard (HP) workstations. In 2001, Cassini moved from HP machines to SPARC processor based Sun Ultra series. The transition to Sun hardware platform workstations was driven by performance and cost effectiveness over the HP platform. From 1997 until 2011, all operational software and user home directories were on each individual workstation. Having home directories and ground system software running on local disks was difficult to maintain from the systems administration perspective.

From the beginning of its operations, Cassini has used the web as the primary means of collaboration. Another main component is the Distributed Object Manager (DOM). The DOM is a general-purpose, extensible, customizable, high-performance, distributed, and object-oriented file cataloging system. The DOM as a catalog system specializes in search and file description, rather than the broader functionality of other database management systems. In addition to general file cataloging and search services, the DOM provides special integrated support for file management using Standard Format Data Unit (SFDU) metadata labels and commercial wide-area distributed file systems. Figure 1 illustrates the Cassini Andrew File System (AFS) DOM configuration.

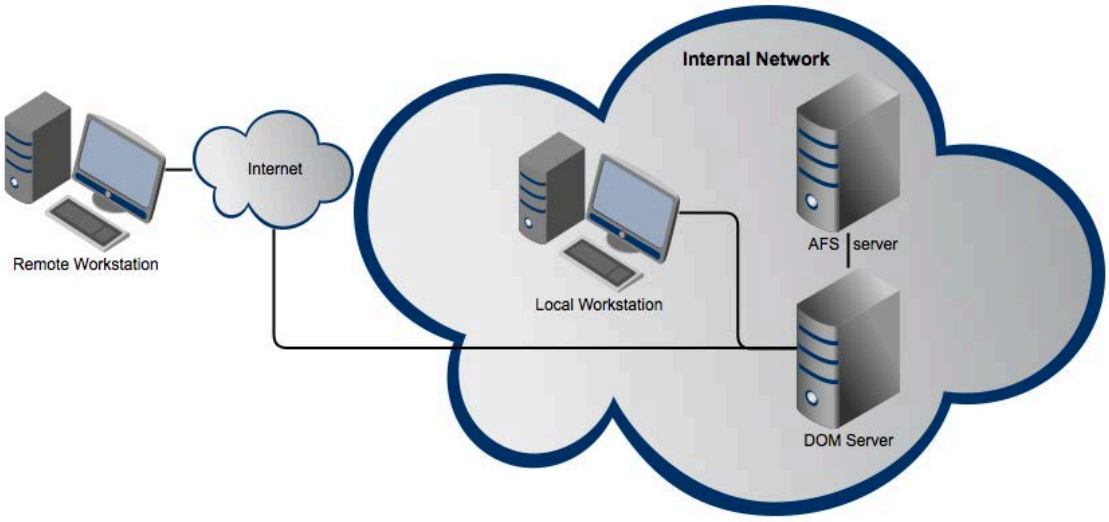

Figure 1. Cassini AFS DOM configuration

Cost effectiveness became the primary purpose of the modernization efforts. The iterative modernization process has been guided by Cassini's change control policy. The Cassini Mission's change control policy has always required appropriate signatures, documentation, reviews and approvals. The original Cassini configuration management guidelines, although comprehensive, focused mainly on major software deliveries. Over the years, the guidelines were restructured to capture more details such as hardware reassignments, minor software deliveries, and documentation. Figure 2 describes Cassini's hardware, network, and infrastructure CM process.

Revising the configuration management plan allowed GDS engineers to redefine roles and responsibilities. The outcome was a clear process to request, approve and document engineering as well configuration changes. Cassini's ground system architecture for operations, test, and development environments were configured as mirrors of each other by design, to provide a platform that supports the configuration management process. Each environment is under different levels of configuration management, supporting the necessary control and flexibility where needed. This supports each modernization cycle through the development and delivery of robust products and services.

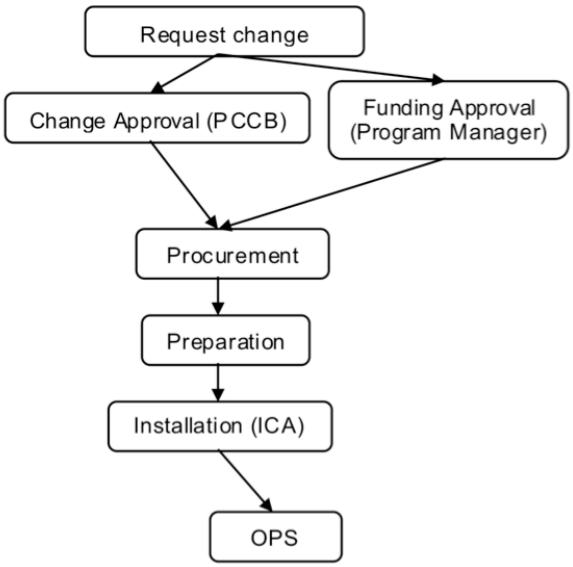

Figure 2. Hardware, Network and Infrastructure change CM process

III. Methodology for Modernizing

The traditional principles of ground data systems engineering focus on a finite project lifecycle and design the system over the projected lifespan. Usually, initial GDS development focuses on the definition of requirements and design, with minimal emphasis on maintenance. However, Information Technology (IT) has changed the profile of GDS maintenance over the last two decades. Ubiquitous computing has forced organizations to place greater emphasis on the management of its IT resources. Organizations today must consider the disruptiveness of innovative IT and must evaluate their change management policies. Therefore, mission operations ground systems managers must consider the appropriate methods to inject modern technologies into their ground systems.

Several methodologies can be used for a repetitive evaluation process focused on the development, production, and deployment of major ground system upgrades. Nevertheless, the agile method for an iterative system development life cycle fits ground system modernization for long-term missions. This system development method frequently uses the spiral model.[6] As described in Figure 3, the spiral model is characterized by a series of revisions, based on feedback. This methodology reduces risk and accentuates continuous feedback, where each cycle builds upon improvements from previous iterations. [1] It also lessens major risks by incremental updates to the GDS.

Unit, performance, system and end-to-end tests were used in each Cassini system upgrade cycle. For hardware upgrades that impacted one particular team, side-by-side performance testing took place to ensure that new hardware environment met the team's needs. The agile method resembles a GDS project lifecycle, by using: the planning phase, development phase, testing phase, and implementation phase. Nevertheless, the iteration of each phase accomplishes significant changes with low risk. The agile approach focuses on creating favorable elaboration of systems with explicit process guidance defining objectives, constraints and alternatives.[1]

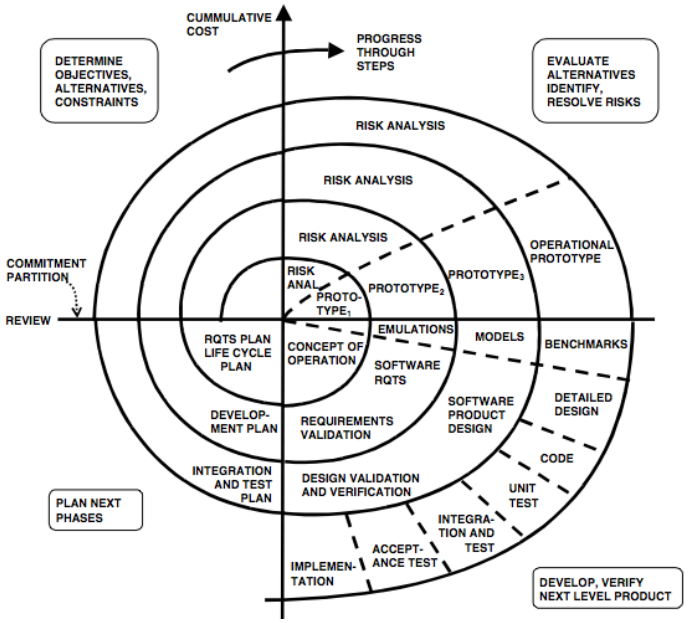

Figure 3. The Spiral Model. This model represents a series of iterations typically used in agile system development methods.[1]

A. Planning Phase

The planning phase starts with the process of identifying needs before evaluating each alternative ground system update. All teams are given the opportunity to provide inputs and specify requirements in order to keep in mind necessary subsystem functionality. Tools, processes and teams impacted are noted in order to ensure that there is ample time for adapting tools, modifying processes and training personnel. Risk, cost, plans and schedules are calculated during this phase. Documentation for the development, testing, and planning are established with buy-in from stakeholders. Aspects such as security, cost, performance, risk, and requirements stay as top priorities for the planning phase. Critical questioning and understanding of policies support the decision making process in the planning phase.

In Cassini, for each modernization cycle the goals and guidelines were clearly defined. These were defined as follows: 1) Increase efficiency; 2) Reduce maintenance cost; 3) Increase security; 4) Streamline operations; 5) Document processes; 6) Enhance overall ground system knowledge; 7) Control risk. This was the framework for the decision support system. All teams were empowered and engaged to collaborate by specifying requirements that ensured harmony of each subsystem (listed in Table 1). It required relying on each team's expertise and creating interfaces functioning together with the technology selected for modernization. Tools, processes and teams impacted were noted to provide ample time for development, adaptation, and training. The GDS engineer coordinated schedules and documentation, in order to maximize results during the ground system development, testing, and deployment phases.

Subsystem
Mission Planning, Sequencing, Command Processing
Telemetry and Tracking Data Processing
Mission Monitoring, Data management and Archiving
Navigation, Science Data Processing, Flight System

Table 1. Cassini Subsystem Functionality

B. Development Phase

During modernization, the development phase is similar to the original development of a GDS with the caveat that there is a working environment that will be enhanced for operations. In the development phase, concept exploration begins by comparing available upgrade options of the existing system.[3] This demands a deep understanding of the existing environment, roles and responsibilities in order to develop the system-level architecture. Understanding the interacting components of the ground system assists defining the integration, test and deployment plans. In order to avoid breaking critical systems and processes, the GDS engineer guides the development ensuring compliance with standards, practices, and quality. Also, because changing processes can be cumbersome for teams, the approach to the ground modernization relies on maintaining principles of existing tools, processes and environments when possible.

C. Test Phase

The test environment allows for broader identification of flaws missed in the planning and development phases. It also helps define details regarding deployment in the operational environment. It verifies the complete integration, including interfaces of the system for operational mode. The test phase involves participation from all the impacted teams. The GDS engineer coordinates the test phase by maintaining a list of tools, systems, and processes to be tested by each team. The test phase can be as little as unit testing of tools, all the way to end-to-end testing of workflows, and final products. During the test phase, if any issues arise, they are noted, and resolved when possible.[3] In many cases, another cycle of development and testing must be implemented to ensure all issues have been understood and addressed.

During each testing phase, the two main questions to ask are "Have we built the right system?" and "Have we built the system right?" In this case, the first question is to make sure that the ground system has been improved. The second question is to make sure that the ground system does what it supposed to do. The systems engineers verify that the requirements are satisfied and fit the current state of the mission. Testing official operations methods allows inspecting, and demonstrating mission-expected outputs as well as performance of the system. The GDS Engineer leads the implementation and deployment of the testing environment. A comprehensive list of functions and tools traces validation results in chronological order to mimic the actual real-time operations. The results are documented by capturing the environment, inputs, outputs and anomalies.

D. Implementation Phase

The modernization of the ground system is a gradual development process and so is its implementation. Implementation of the transition plan to deploy upgrades to the operational ground system takes place after testing has been completed. This requires an execution plan, training and documentation of update processes. The implementation phase is planned as building blocks to continue the foundation of verification. This approach allows troubleshooting issues, while avoiding adding extra variables that could delay or add complexity for teams involved in implementing the changes. This phase not only completes one cycle, but also begins a new one, due to feedback, that is continually received from the stakeholders. The feedback drives development of requirements for the next modernization cycle.

IV. Cassini Agile Method Cycles

The Cassini historical ground modernization was comprised of several iterations. Most cycles were completed while the project was fully staffed. The Cassini project has capitalized on the spiral model to succeed in its various iterations of modernization. Changes such as transitioning from standalone workstations to implementing a server/client environment benefited teams by providing large storage and more computing speed.

A. Cassini Modernization Stages

Based on initial performance evaluations, priorities and needs, critical components became the main design focus. The original Cassini ground system was revolutionary and fully controlled by the mission. As newer disruptive technology became available, security and performance challenges surfaced for Cassini's ground system. Gradually, the mission evolved its ground components through cyclical upgrades that provided significant advantages and satisfied changing IT requirements. The upgrade cycles are illustrated in Figure 4. The most significant accomplishments are listed in black and expanded in the modernization journey.

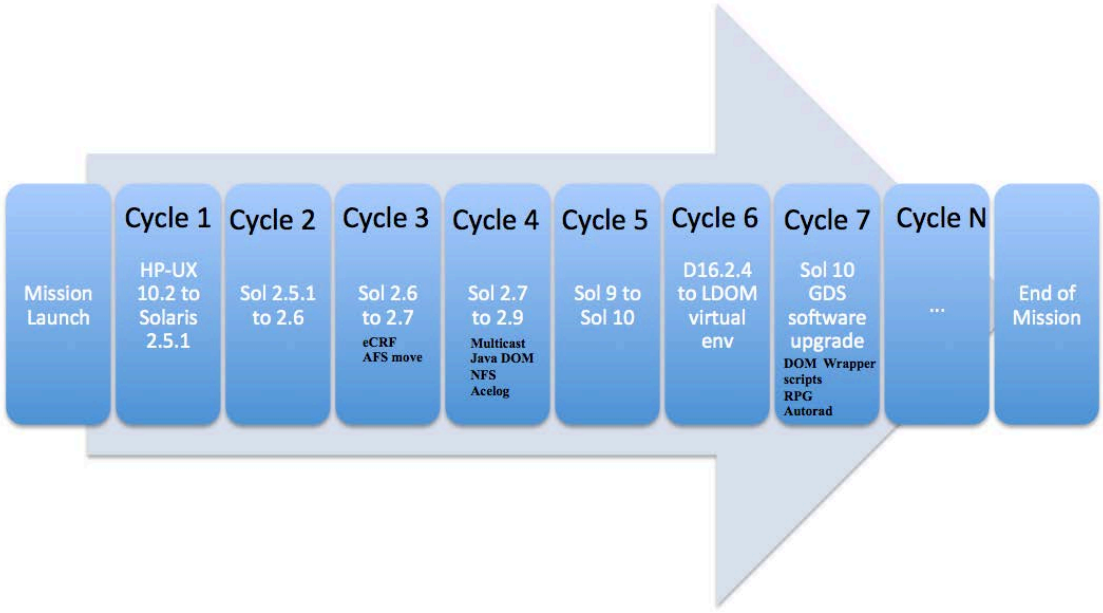

Figure 4. Cassini's modernization cycles. Some significant updates are listed in black below the major modernization cycles, which are listed in white.

B. Modernization Journey

1. *Cycle 1*

The first significant upgrade was the transition of system hardware and operating system from HP-UX 10.2 to Solaris 2.5.1. This modernization effort required rather important changes to mission critical software. The mission's software applications source code had to be rewritten and recompiled for the Sun

hardware CPU architecture. The overall accomplishment was to maintain functionality of the ground system during this major change. Also, it became significant source of knowledge regarding Cassini software dependencies and the existence of user-developed tools that gradually became embedded in critical operational functions.

2. *Cycle 2*

The first maintenance upgrade to enhance security was the transition from Solaris 2.5.1 to Solaris 2.6. The transition addressed remnant source code transition fixes left from the transition from cycle 1. During this cycle, the Lightweight Directory Access Protocol (LDAP) service became part of the Cassini web platform. This was an innovative approach at the time because neither the institution nor any other mission had adopted an authentication system as a service. Eventually JPL adopted LDAP as the institutional directory service, using lessons learned and knowledge from Cassini.

3. *Cycle 3*

The OS upgrade from Solaris 6 to Solaris 7 introduced the electronic Command Request Form (eCRF). The eCRF tool transformed a paper-based process of identifying a command data file and transmitting that information to all mission parties involved before sending it to the spacecraft, to an automated process accessed through a web interface.[7] Another event was the transition from an AFS server in a limited access network to an AFS server located in a more open network.

4. *Cycle 4*

This cycle was an OS maintenance update from Solaris 7 to Solaris 9 for the workstations and servers. This entailed a significant applications redevelopment for Cassini, because the move from Solaris 7 and Solaris 9 was major operating system revision. In this cycle, Java DOM, NFS DOM, and web-based tools became prominent part of mission operations. The Java DOM and NFS transitions are discussed further in cycle 7.

The transition from broadcast to multicast also took place in this cycle. Telemetry data had been distributed within the JPL operations network via broadcast for several years. As technology evolved and the reliability of broadcasting services deteriorated, multicast became an option that the mission took as an opportunity to improve data delivery to users. Many legacy projects did not make this move, but because Cassini had already been through significant modernizations cycles, multicast was conveniently understood and implemented. Multicast services reduced the network traffic and allowed data to flow across networks.

5. *Cycle 6*

In cycle 6, the Logical Domain (LDOM) technology introduced virtualization to Cassini's ground system. LDOM offered server virtualized and partitioned virtual environments accessed via a "thin client".[4] Cassini success relies on the dependability of the file system supporting operations, including its data availability. For the Cassini mission, reducing downtime due to failure and maintenance has always been a priority.

As the LDOM architecture became part of the ground system, NFS became a viable alternative to local storage. The selection of NFS was based on testing results, historical statistics, storage capacity, and vendor support. NFS and the "thin client" configuration based on Sun Ray servers and DTU clients delivered numerous layers of data recovery. This allowed replacing workstation local disk storage with NFS. Local disks constantly failed and recovery of data on such devices was not always possible. This was a major factor that encouraged users to fully support the transition to NFS. Cassini expanded the use of NFS services as part of the Solaris 9 to Solaris 10 modernization iteration. NFS provided data centralization, reliability through redundancy, and cost effectiveness. Cassini's high availability NFS service is shown in Figure 5.

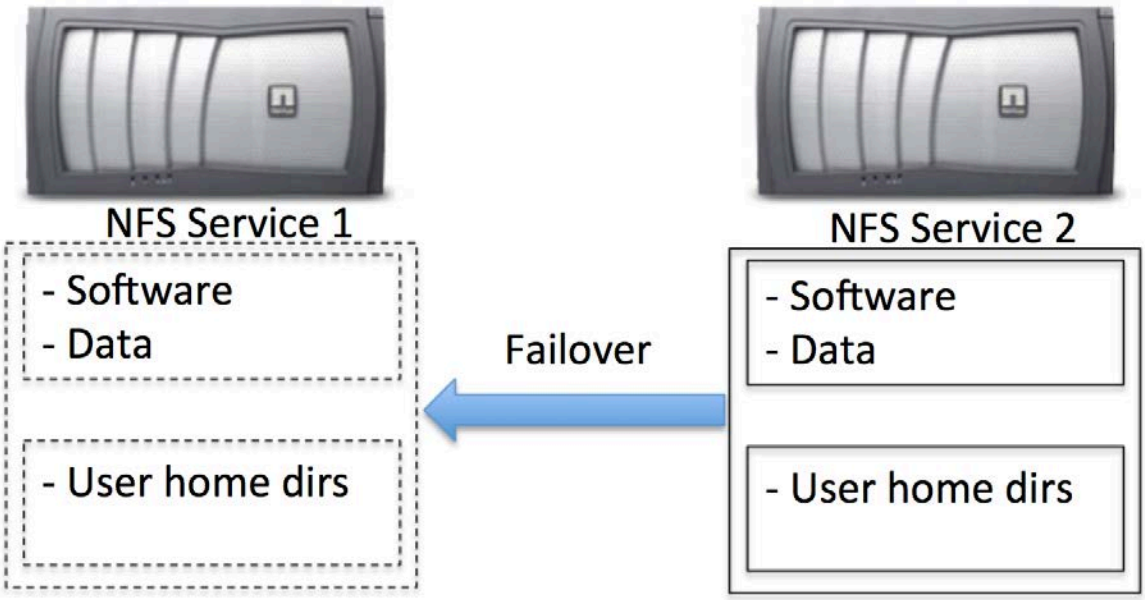

Figure 5. Cassini NFS high availability configuration.

This modernization cycle also incorporated software changes such as the File Release Notification Services (FRNS) that were made possible due to the upgrade of DOM. The file manager system in the server side was enhanced by the message reactor services. The message reactor is a Java Message Service, which sends messages to users on specified events. The message reactor facilitated significant capabilities to enhance processes and reduce the time required to generate and distribute products. The message reactor concept was new to Cassini's subsystems, but the teams immediately explored its features and made it part of their processes.

6. *Cycle 7*

One system upgrade in particular, had a domino effect on infrastructure and applications. That system is AFS DOM, the project's sequence file repository system. JPL removed the AFS server from the operations network to a less restrictive network. This compelled the mission to make the move during the modernization cycle 3.

A couple of years after this transition, JPL AFS services were gradually downgraded. The original DOM system was a C++ language based tool, which worked well with AFS. The AFS provided convenient accessibility for remote users. The AFS required users to authenticate before accessing the DOM, in order to grant the capability to traverse through the file system. As AFS maintenance cost became expensive and its support limited, moving to NFS for DOM became necessary. As its name says it, NFS is better fitted for access from internal network clients. The C++ DOM tool did not support (Remote Method Invocation) RMI, which was needed for remote sites to access the NFS DOM. In the NFS configuration, the remote client's access control was based on the Java RMI, whether the workstation was inside the operations network or not. Cassini was compelled to take a new DOM tool version written in Java, which supports RMI.

A multitude of various Solaris and web-based tools were deeply embedded into the C++ DOM. An end-to-end test became necessary, as well as weekly coordination meetings to ensure that the various tools and scripts were thoroughly adapted, tested and trained on the Java DOM. The initial findings were that the Java version of the DOM client did not have the same behavior and functionality as the C++ version. Some discrepancies in the Java version were: asterisks "*" no longer being considered NULL values, external users could not longer see the file system, and inconsistent metadata in files. This required the Cassini mission to develop work-around software patches to avoid breaking embedded processes.

Moreover, members reported that Java DOM performance was significantly slower. This prompted Cassini to examine the DOM hardware configuration. It became evident that the transition from AFS to NFS was a downgrade in regard to technological advantages. In fact, this change was driven by a directive to adopt a more widely used and less expensive technology.

At this point, end-to-end testing played a significant role. Cassini GDS engineers scheduled a test of every tool related to DOM and documented findings. This required Cassini teams to run their processes in chronological order. The coordination of teams and activities in the test environment determined if the update was going to become a viable option or if the GDS engineers needed to go back to the drawing board. In the end several compromises were made in order to move into this environment, with the clear intent of fixing flaws in the next ground system revision cycle.

The Solaris 10 GDS software upgrade cycle became a significant maintenance upgrade as some Java DOM bugs detected in the D16.2.4 version were addressed. This meant that the original software patches developed in the previous cycle could be eliminated or required modification. Compromises made in D16.2.4 were well understood and documented, which facilitated a smooth system development life cycle. The interrelated processes were already clear and the general understanding of the system was well spread in the mission's personnel. Once again, this cycle required development, testing, and deployment phase involving every stakeholder.

In cycle 7, LDOMs architecture was further enhanced to address critical performance issues. After the LDOM deployment in the previous cycle, it was discovered that the virtual storage configuration was subject to instability due to frequent disk failure and required a significant effort to maintain. In order to reduce the intensive system administration and configuration management effort, the storage was reconfigured to only utilize physical hard drives. For every revision cycle, the embedded functions of every subsystem producing products were revised, and if possible enhanced.

In this cycle, a layer of security was added to older equipment of high criticality to the mission. This equipment could not be updated and was based on vulnerable protocols of communication. In order to meet IT security requirements, the equipment was isolated behind the remote partner gateway (RPG). The RPG is a demilitarized zone (DMZ) segregated network, whose machines are excluded from the corporate internal network, adding security.[6] This effort is an example of making dynamic adjustments to continue operations.

C. Compile feedback and revise prototype

During and after every cycle of the agile system development method, Cassini has captured findings through email conversations, tracked via the Java EE web-based bug tracking and issue tracking application (JIRA), and Inventory Change Authorization (ICA) records. Although many flaws or inconsistencies are found before testing, many cannot be fully addressed in the revision cycle. Hence, keeping a comprehensive list of needed improvements has assisted the management of resources for future development life cycles. Capturing details of each issue and the interrelated tools and processes has reduced the analysis effort for each ground system revision. A significant evolution during capturing feedback was the creation of the Cassini configuration management tool, which replaced a paper-based process and eventually became an institutional multimission tool.

V. Modernization Advantages and Challenges

Modernization of the GDS can vary from changing a mission operation's processes to upgrading the entire hardware, network, and software. For Cassini, modernization was necessary due to the need to increase processing power on machines, revamp legacy software, reduce cost, enhance network speeds, replace services, and perform software upgrades. At times it can be due to security requirements, a reduction in personnel, need for automation, or end of life support from vendors.

A. Advantages

GDS Modernization can be highly beneficial to a flight project. It offers the opportunity to decrease computing time, increase security, reduce cost, and enable superior software capabilities. Modernization of a ground system can potentially provide new and better features such as faster processing, automation, and real-time notifications.

Cassini's mission has innovated in its ground system modernization and along the way new tools have been developed due to its endeavors. It has driven the development of new processes by providing teams the opportunity to upgrade their tools, adding much needed enhancements, and creating compelling opportunities to streamline processes and make operations more reliable. The ultimate objective of a GDS Engineer is to provide an integrated

suite of subsystems supporting mission's critical events: launch, cruise and encounter (operations) phases.[2] Most of the time, this means not all the requirements are fully agreed upon as the mission develops. Modernizing the ground system provides the opportunity to clean up, find or remove flaws, and educate the mission personnel regarding the system. Also, this effort becomes a knowledge creation process particularly for long-term missions. As the work force changes over time, knowledge retention weakens when engineers and scientists move to other missions or retire and become replaced by younger generations. In addition, despite plenty of documentation, hands-on experience is the main source of knowledge. Therefore, knowledge creation, acquisition and transmission are significant outcomes of ground system modernization efforts.

B. Challenges

With any development, there are several challenges to overcome. Unfortunately, some dependencies that were known when the ground system was formulated, were not fully documented, understood or communicated. This deterred mission members from accurately identifying the tools and processes that could be affected by ground modernization efforts. This became evident with each examination of the ground system during initial modernization efforts.

Some of the cons of modernization are that it can break existing tools and processes, which can interrupt project operations. Change is difficult in general, and more so when it is done on a mission that performs critical science with intensive navigational maneuvers. The user community does not necessarily buy-in until the final product is delivered. Consequently, strong leadership and communication from the GDS engineer is important. Another challenge is resistance from teams, due to the different time zones covered by the ground system and the operations teams. Therefore, GDS engineers play a critical role in this effort by thoroughly vetting the overall system from end to end. Moreover, modernization is a lengthy process that requires reviews, coordination, and downtime without interfering with the spacecraft operations. In consequence, addressing conflicting schedules and resolving resource divergences challenge modernization. Software as well as hardware upgrades can take away capabilities depending on the method and reasoning for modernizing. Hence, it is critical to effectively lead, communicate and provide a vision of the ultimate goal for the mission.

VI. Conclusion

From its inception, the Cassini/Huygens mission provided scientists the opportunity to explore Saturn. The mission timeline will cover an expanse of two decades upon completion. The longevity of the mission and constant IT changes have compelled Cassini's systems engineers to rely on the agile method for ground system modernization. In recent history, IT services have become centralized, forcing missions to refocus efforts on modernization in order to remain functional.

The modernization of a mission's ground system is a challenge due to its interrelated components working together. However, significant modernization becomes reality when it is implemented gradually with deep analysis, involvement, clear communication, and teamwork.

Unless a ground system is meant to support a short mission that has full control of its IT resources, a standard system development life cycle might work; otherwise an agile approach is recommended. In the case of Cassini, modernization has been gradual, allowing for automation, clarification, and redefinition of teams' responsibilities. It has streamlined processes, enhanced data recovery, and updated the IT infrastructure. Each ground software upgrade has addressed external and internal requirements in an incremental manner, without adding significant risk to mission operations.

The success of developing and implementing the ground system modernization is measured by the several challenges faced before its accomplishment. Cassini success can be summarized by the fact that it has adopted technology that eventually has become part of other missions or the JPL institutional services. Consequently, Cassini's achievement serves as a model to follow for other long-term and future missions, which might shy away from the agile ground system modernization model.

Acknowledgments

The ground system modernization was funded, planned, designed, developed and implemented by the Cassini mission system engineering teams and system administrators at the Jet Propulsion Laboratory and the remote sites, under a contract with the National Aeronautics and Space Administration (NASA).

References

[1]Boehm, B., "Spiral Development: Experience, Principles, and Refinements," Carnegie Mellon University, Pittsburgh: Carnegie Mellon University Software Engineering Institute, 2000.

[2]JPL. Basics of Space Fight Section III. URL: http://www2.jpl.nasa.gov/basics/bsf16-1.php [cited 20 March 2014)]

[3]Kossiakoff, A., and Sweet W. "Systems Engineering Principles and Practices," Hoboken, New Jersey: Wiley-Interscience, 2002.

[4]Microsystems, Sun. "Sun Ray Server Software 4.0 Administrator's Guide for Solaris," URL: http://docs.oracle.com/cd/E19634-01/820-0411/overview.html [cited 19 March 2014]

[5]Shelly, G. B., and Rosenblatt, H. J., "Systems Analysis and Design, Video Enhanced," 8th. Boston, Massachusetts: Course Technology Ptr, 2010, Chap. 1

[6]Shinder, D. (2005, June 29), "SolutionBase: Strengthen network defenses by using a DMZ," from TechRepublic: http://www.techrepublic.com/article/solutionbase-strengthen-network-defenses-by-using-a-dmz/ [cited 12 March, 2014)]

[7]Wong, C. K. (2006), "Development and use of a web-based automated command request application in a distributed operations environment for the Cassini Saturn Mission," *AIAA 9th International Conference on Spacecraft Operations (SpaceOps)* (pp. 1-10). Pasadena: Jet Propulsion Laboratory.