This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by Space Grant and the National Aeronautics and Space Administration.

Student: Robert Alland
Mentor: Diane Conner
Project Report

Introduction

This paper describes my work with the Cassini Mission Sequence Subsystem (MSS) team during the summer of 2011. It gives some background on the motivation for this project and describes the expected benefit to the Cassini program. It then introduces the two tasks that I worked on – an automatic system auditing tool and a series of corrections to the Cassini Sequence Generator[1] (SEQ_GEN) – and the specific objectives these tasks were to accomplish. Next, it details the approach I took to meet these objectives and the results of this approach, followed by a discussion of how the outcome of the project compares with my initial expectations. The paper concludes with a summary of my experience working on this project, lists what the next steps are, and acknowledges the help of my Cassini colleagues.

Background

The Cassini spacecraft is currently performing its Solstice Mission, scheduled to run until the year 2017. To successfully perform this mission, the Cassini program requires its hardware and software configurations to match an expected state. Of interest to this project are two areas of system cohesiveness: the configuration of computers that are part of the Cassini team's Ground Data System (GDS) and consistency in the command parameter validation and constraint enforcement performed by SEQ_GEN.

Current methods for monitoring the state of the Cassini GDS have several problems: they can be inaccurate, they are missing some information, they are missing historical information, and they are manual. My work is expected to contribute to the production of an automated asset auditing tool that overcomes the inadequacies of current auditing methods.

Sequence generation for Cassini has some inconsistencies between the behavior of SEQ_GEN and the expected behavior as described in the Cassini Operational Reference Encyclopedia (CORE): there are some cases where checks in SEQ_GEN do not identify invalid commands and there are some cases where checks are too strict and reject valid commands. My work will contribute to an update of SEQ_GEN that improves consistency of the command parameter and constraint checking while helping to avoid commanding errors.

[1] SEQ_GEN is the Cassini Adaptation to the multi-mission SEQGEN program, which validates and models a proposed sequence of commands for the spacecraft

Objectives

One of my tasks this summer was to adapt the Online Asset (OLA) software from the JPL Multi Mission Office, written by Kelly Breed in Perl, for use on the Cassini GDS. This will help Cassini satisfy JPL IT Security Database (ITSDB) requirements and more easily understand and manage the configuration of the GDS at any point in time. The first goal was to complete the adaptation of the existing OLA software and automate production of a single XML, CSV, and TXT (for ITSDB) output file for each report. The next step was to produce at least a basic web interface where all output files could be viewed and downloaded. Advanced features were desired but not expected.

The second task involved modifications to SEQ_GEN, detailed in Cassini Engineering Change Request (ECR) 111742. I corrected the command parameter checking for a subset of Attitude and Articulation Control Subsystem (AACS) commands. This update will increase the number of commands that are allowed for uplink, identify additional commands that are not allowed for uplink, and reduce parameter errors in uplinked commands. Working with me on this task was Jeff Boyer, SEQ_GEN Cognizant Engineer/Programmer for Cassini. There are a total of 66 changes across 53 different commands in the ECR, of which 42 are assigned a high priority based on their impact and the frequency of use of the affected commands. My objective was to implement and test as many high-priority changes as would fit in my schedule and as could be coordinated with the MSS and AACS teams.

Approach

GDS Auditing

The original hierarchy of the asset auditing program consists of several layers. The middle layer, part of the original software package from Kelly Breed, is the 'olaManager' Perl script responsible for collecting audit data from the GDS machines. A single run of olaManager can collect data from a single hostname or IP address, a range of IP addresses, or a single subnet. The top layer was comprised of several Unix shell scripts that convert the output to CSV, create time-stamped log files, and call the olaManager on multiple subnets. Each computer to be audited runs another Perl script, the 'olaListener,' that communicates with the single olaManager using TCP/IP sockets and sends a message containing its own configuration (in XML format) back to the manager. The original hierarchy of the auditing program can be seen in Figure 1.

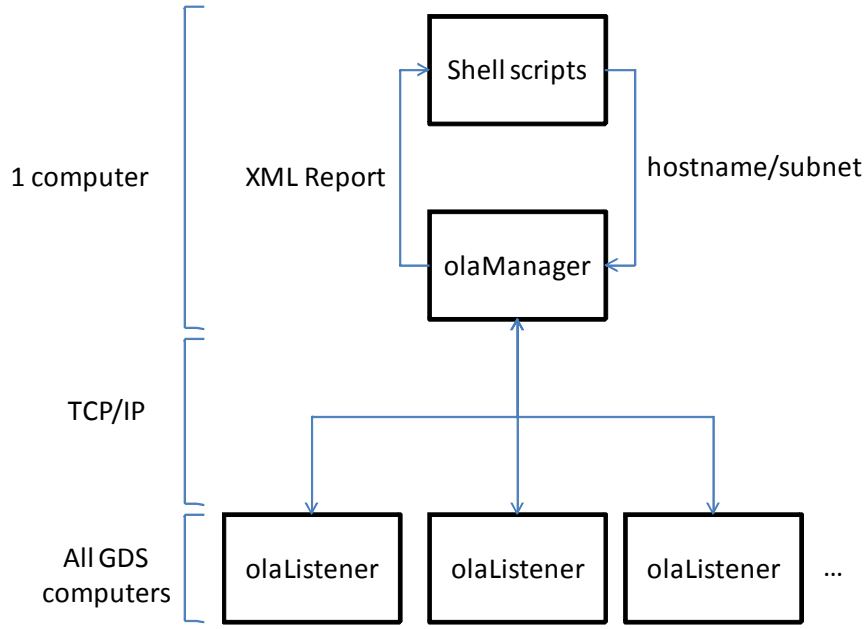

Figure 1 Hierarchy of Original Asset Auditing Program

After first implementing the creation of a combined audit log from each subnet's individual log within the confines of shell scripting, I rewrote these top layers as a single Perl script (the 'olacron'). It uses the Perl LibXML library to combine the output of each call to the olaManager, removing the need for any intermediate files; it also uses an external text file to list each subnet that will be audited, simplifying modification. This script produces a single XML audit log for the entire ground data system with the following format:

- One <LOG> root element
  - One <REPORT> element for each call to olaManager
    - One <OPTIONS> element, listing the hostname, subnet, IP address, or IP address range used to generate the report
    - One or more <MACHINE> elements, containing the configuration data for each machine audited

The olacron uses another Perl script, originally written by my mentor, which required only minor modifications, to convert this to a CSV file. These two output formats, along with a tab-separated format described below, are automatically uploaded to the Cassini internal web server using the 'rsync' command; the entire script is run as a cron job once per day.

A special tab-separated text file, organized by property number, is used to upload system information to the JPL ITSDB. I wrote another Perl script, called 'olaITSDB,' to produce this formatting from the raw XML. It was derived from the CSV-conversion Perl script, with the major addition being the combination of machines with different IP addresses but identical property numbers[2] into a single row of data. The formatting and permitted values for this are described in the ITSDB_help_batch.doc. In most cases this

[2] For example, a single physical machine running multiple virtual machines using Sun's Logical Domains (LDOM)

script uses a default value or simple lookup, provided by my mentor, to fill in the required ITSDB field. Determining the value for a computer's operating system is performed using a multi-level lookup table that is generated by reading in an external text file.

The final component of the asset auditing task is an internal website for hosting all audit logs. To determine desired features of the website, I led a meeting organized by my mentor with future users of the auditing script and its output. At the meeting the following changes were requested:

1. Audit reports:
    a. Only use the local MDAS version
    b. Add uptime
    c. Add serial number
    d. Add host ID
2. Web interface:
    a. Sorting and filtering an audit table
    b. Including/excluding data fields
        i. Checkboxes to individually select fields
        ii. Default template for selected fields
    c. Export a table from website
    d. Calendar for selecting past audits
3. Real time monitoring:
    a. reporting system up/down status

To begin the website, I first created an Extensible Stylesheet Language Template (XSLT) that is used to transform an XML audit log into an HTML table that is much more human-readable. I also used Tablesorter, a plugin for the jQuery[3] JavaScript library, to add sorting functionality to this table. Although this plugin is very extensive, I added new parsers for the specific formats used in the time and memory size columns and fixed several bugs in the IP address parser. Even though a link to this XSL stylesheet can be embedded directly in the XML, I chose to create a separate webpage that performs the transformation using JavaScript[4]. This leaves the original XML file unaltered and allows it to be viewed without the transformation if desired; the page also uses PHP and a URL parameter to dynamically select the XML file to be transformed.

To begin developing the actual web interface, I wrote PHP code to list all files in the directory containing the uploaded logs. To facilitate easy viewing of the logs, this code was then modified to output a log's base filename followed by links to all present formats on the same row. This format can be seen in Figure 2.

Figure 2 Example Listing of an Audit

[3] jQuery is a cross-browser compatible JavaScript library
[4] This page was adapted from http://www.w3schools.com/xsl/xsl_client.asp

After outputting the link to the raw XML, the PHP automatically inserts a link ("[html]") to the XSL transformation page with this XML file as the URL parameter. The most recent log is listed first, followed by a listing of all logs.

I gave the website a vertical navigation bar on the left, written using jQuery, and created a page listing the subnets being audited along with a description of each subnet. Cascading Style Sheets (CSS) were used to provide layout and styling for the website. I also implemented one of the requested features: calendar selection of past audits. Using the jQueryUI[5] datepicker to select inclusive beginning and end dates, the list of audits is filtered and displayed. This is the only additional feature that I had time to implement.

SEQ_GEN Corrections

There are three main components that I used in my task of correcting SEQ_GEN: a spacecraft activity sequence file (SASF) that lists the activities and commands to be performed in the sequence, the AACS spacecraft model file (SMF) that controls the command parameter and constraint checking for AACS commands, and an environment file that identifies the inputs and outputs to the sequence generation (including the AACS SMF) for running SEQ_GEN. After familiarizing myself with editing the environment file to generate the correct sequence, I continued this task by working on simple changes to the SMF. The process for making a change has four steps:

1. Create a custom SASF file that will test the changes to be made.
    a. For constraint checks, this SASF must get the spacecraft in an initial state that will cause the targeted constraint check to occur[6], and then must call the targeted command.
    b. For parameter checks, this SASF must simply call the targeted command
    c. In both cases, the SASF must also call commands that are expected to be unaffected by the changes
2. Incorporate a single change into the SMF
3. Run SEQ_GEN with the custom SASF for both the new and old model files
4. Compare the two outputs, making sure that the desired effect was produced but that there were no other collateral changes

After performing a series of changes to the SMF, I would integrate my version of the file with the version under development by Jeff Boyer, who then committed the changes to AccuRev, the Cassini software configuration management system. We coordinated the work on and completion of changes through a shared Excel spreadsheet.[7]

I began working on corrections that required only slight changes to the wording of error messages, followed by deletions of unnecessary or redundant checks. I then moved onto more difficult changes, beginning with expansion checks: checks that deal only with a command's parameters and not with the

[5] The official user interface library for jQuery
[6] In some cases the input conditions file, which contains the state of the spacecraft model, must also be edited to produce the desired state
[7] ECR111742_spreadsheet.xls, part of ECR-111742

spacecraft model's state. Finally, I performed corrections that involved more complicated logic and the state of the spacecraft. In the process of making these corrections, I had several questions about the necessity or specifics of a desired change. While Jeff Boyer was able to answer many of my questions, some of them were answered by members of Cassini's AACS team.

Results

I successfully adapted the OLA scripts for use with the Cassini Ground Data System. I automated the production of a single XML audit log, along with corresponding CSV and ITSDB-compatible TXT outputs. These files are automatically uploaded to an internal web server, where they can be located and viewed on an internal website. I also implemented filtering of the audit logs based on a desired start and end date. I met the basic expectations of this task, and had time to implement one additional feature that will make the auditing website more useful.

I encountered two, as yet unresolved, main problems in my work on this task. The first is that the OLA software does not correctly capture the presence of virtual IP addresses of a system.[8] My mentor is coordinating investigation of this problem, and potential solutions, with the original author of the scripts (Kelly Breed) and one of the Cassini system administrators (Nick Patel). The second problem is that, for some of the software version fields, the auditing script produces results with different formats on different internal networks, making comparisons and formatted output difficult. For now this problem is accepted, because while the formatting of the results is undesirable they are still correct.

I was able to implement a large portion of the priority '1' SEQ_GEN changes.[9] The only remaining priority '1' changes are awaiting comment and clarification from the Cassini AACS team before they are implemented. While there was no set expectation of the number of changes to complete, my work marks significant progress towards the completion of ECR 111742. My contributions are scheduled to be delivered for operational use in the MSS D14.4 release, which is currently scheduled for delivery in Spring 2012.

Discussion

I knew before beginning my project that I would use for the first time both Perl and the proprietary language[10] used in SEQ_GEN. My expectation that Perl would be relatively easy to learn, given my prior experience in C++ and Java, turned out to be correct; the book "Learning Pearl" by Randal L. Schwartz and Tom Christiansen was very helpful in introducing me to the language. The adaptation used in SEQGEN was easier to learn than I had anticipated as the constructs used are not overly complex, and I was able to mimic preexisting code for many of my modifications.

---

[8] This is caused by the use of the Perl Net library's hostfqdn() function, which only returns the physical fully qualified domain name

[9] Priority 1 changes are those that are considered most important, based on the effect of the change and the level of usage of the command

[10] This is an interpreted language that provides constructs and methods to define, model, and check spacecraft sequences and ground events in SEQGEN

In my work on the web interface for system audits I was introduced to JavaScript, PHP, and CSS for the first time. Although this introduction was not comprehensive, I did not expect learning as much about these concepts as I did; as a result the features and sophistication of the website that I developed were beyond my original expectations. I enjoyed learning more about web development, and am glad that this portion of the project benefits both my own professional development and the Cassini program.

Before I began my project I expected to spend around 80% of my time on the SEQ_GEN modifications and the other 20% of my time on development of the GDS auditing scripts and website. I actually split time about evenly between the two tasks, because the required feature-set of the auditing task that beyond minor adaptations and bug fixes.

Conclusions

In the course of this project I learned the difficulty of managing the cohesiveness of the Cassini GDS and of the Cassini SEQGEN adaptation. The project illustrates the difficulty in understanding the configuration of a system or program as well as the challenge of correcting any inconsistencies in that configuration. I made significant progress in providing the Cassini team with an automated system auditing tool that overcomes inadequacies with previous methods. There are still unimplemented features and several unsolved problems with the auditing scripts, but in its current state the program is functional and produces correct results. I made good progress in performing corrections to SEQ_GEN's command parameter and constraint checking, completing nearly all of the high-priority changes. My work on this task will eventually be delivered for operational use and will improve the consistency and accuracy of Cassini sequence generation. While there is more work to be done for both tasks, I am leaving my work in a usable state that benefits the Cassini team in its GDS management for the remainder of the Solstice Mission. Jeff Boyer will be responsible for completing the SEQ_GEN modifications; Shahen Petrosian, another member of the Cassini MSS team, will be responsible for continued development of the GDS auditing scripts and website.

Acknowledgements

Diane Conner, Cassini Integrated Uplink Systems Manager, for guidance in and help with the GDS Auditing task

Jeff Boyer, Cassini MSS Principal Programmer, for guidance in and help with the SEQ_GEN correction task

Michael Alland and Matthew Hall, Cassini summer interns, for HTML, CSS, JavaScript, and PHP help used in development of the GDS auditing website

Shahen Petrosian, Cassini MSS Senior Programmer, for Perl help used in adaptation of the OLA scripts

Nick Patel, Cassini System Administrator, for help with adaptation of the OLA scripts

Oscar Castillo, Cassini System Administrator, for setup of and help with creating an internal website