The Integration of COTS/GOTS within NASA's HST Command and Control System

Thomas Pfarr[1], and James E. Reis[2]

[1] Computer Sciences Corporation,
Space Telescope Science Institute,
Baltimore, Maryland, 21218, USA
pfarr@stsci.edu
[2] National Aeronautics and Space Administration,
Goddard Space Flight Center,
Greenbelt, Maryland, 20771, USA
jreis@hst.nasa.gov

Abstract. NASA's mission critical Hubble Space Telescope (HST) command and control system has been re-engineered with commercial-off-the-shelf/government-off-the-shelf (COTS/GOTS) and minimal custom code. This paper focuses on the design of this new HST Control Center System (CCS) and the lessons learned throughout its development. CCS currently utilizes more than 30 COTS/GOTS products with an additional ½ million lines of custom glueware code; the new CCS exceeds the capabilities of the original system while significantly reducing the lines of custom code by more than 50%. The lifecycle of COTS/GOTS products will be examined including the package selection process, evaluation process, and integration process. The advantages, disadvantages, issues, concerns, and lessons learned for integrating COTS/GOTS into the NASA's mission critical HST CCS will be examined in detail. This paper will reveal the many hidden costs of COTS/GOTS solutions when compared to traditional custom code development efforts; this paper will show the high cost of COTS/GOTS solutions including training expenses, consulting fees, and long-term maintenance expenses.

1 Introduction

The Hubble Space Telescope (HST) is NASA's flagship astronomical observatory. HST was originally designed in the 1970s and was launched on April 24, 1990 from Space Shuttle Discovery (STS-31). HST continues to be a state-of-the-art telescope due to on-orbit service calls by Space Shuttle astronauts. The telescope is designed to be modular which allows the astronauts to take it apart, replace worn out equipment, and upgrade instruments. These periodic service calls make sure that HST produces first-class science using cutting-edge technology.

The HST is a low Earth orbiting (LEO) satellite. It is located 320 nautical miles above the surface of the Earth. Each day, HST archives between 3 to 5 gigabytes of data and delivers between 10 and 15 gigabytes to astronomers around the world. HST has a resolving power calculated to be 10 times better than any Earth-based telescope. The telescope has taken more than 330,000 separate observations and has observed more than 25,000 astronomical targets. The telescope has created a data archive more than 7.3 terabytes. HST circles the Earth once every 95 minutes and has traveled more than 1.5 billion miles. Approximately 11,000 telemetry parameters are received from the telescope for 82 minutes of each Earth orbit. HST has received more than 93 hours of on-orbit improvements within three successful servicing missions.

2 Vision 2000 Project

HST has been producing extraordinary scientific results since its launch in 1990. In the mid-1990s, the life of HST mission was extended to 2010. HST Project staff recognized that significant improvements to spacecraft operations and ground system maintenance were needed to maintain the quality of science return and to ensure health and safety of the spacecraft. In 1995, HST Project staff instituted the Vision 2000 Project to reengineer the ground-based control system for HST. The main purpose of the Vision 2000 Project was to significantly reduce the costs of operating the telescope for the life of the mission without impacting ongoing scientific observations. The Vision 2000 Project is organized into four Product Development Teams (PDTs) including Planning and Scheduling (P&S), Science Data Processing System (SDP), Flight Software (FSW), and Control Center System (CCS). This paper focuses on the design of this new HST Control Center System and the lessons learned throughout its development.

The CCS Product Development Team was chartered to create the new Control Center System by reengineer the existing business processes and computer. The CCS Product Development Team successfully utilized new government practices by using government and contractor personnel within a badgeless and co-located facility. The success of the PDT approach was due, in part, to the integration of domain and technology experts within a cohesive team dedicated to a common goal. These domain and technology experts included end users, developers, testers, network administrators, security engineers, and system engineers. Major challenges facing the CCS Product Development Team were to incorporate new technology into the control center, to integrate commercial-off-the-shelf (COTS) products with legacy software, and to provide worldwide access from any remote location. In addition, the new Control Center System had to be modular and scalable such as to support single-server configurations for standard test facilities as well as to support multi-server configurations (~20 machines) for mission-control operational configurations.

3 CCS Overview

The Control Center System is the new command and control system for the HST. The CCS provides a unified architecture for commanding, engineering data processing, data archiving, data analysis, spacecraft and ground system monitoring, and simulation. The CCS is currently installed on 4 operational control center multi-server strings, 25 operational test facility single-server strings, and 4 development multi-server strings. The new Control Center System for HST's Space Telescope Operations Control Center (STOCC) is operated at the Space Telescope Science Institute (STScI) in Baltimore, Maryland.

The Control Center System is a data-driven scalable architecture. The Control Center System is partitioned into six major subsystems including the Graphical User Interface (GUI), Command Processing, Front End Processing (FEP), System Monitoring, Data Management, and CCS Management. See Figure 1, CCS Functional Architecture. A critical Middleware layer is also utilized for inter-process communication. The CCS Middleware layer provides a suite of services for message and data transport between application and COTS/GOTS products while executing on a variety of hardware platforms.

The Control Center System is segmented into secure network levels that accommodate remote engineering data access while protecting the command and control system. See Figure 2, CCS System Architecture. Network and security functions have been integrated into the CCS to provide worldwide access from any remote location. The Control Center System receives spacecraft events from P&S at STScI based on the upcoming science observations. The CCS receives and processes engineering telemetry data. This telemetry is provided real-time to client workstations as well as stored in a life-of-mission long-term archive. The engineering telemetry is also provided to the SDP for use in calibration and analysis of science observations. CCS interacts with the FSW facility to validate and uplink the content of the spacecraft's onboard computer programs.

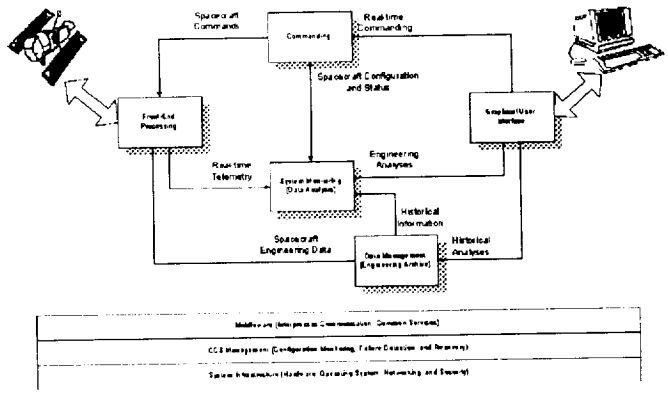

Fig. 1. CCS Functional Architecture

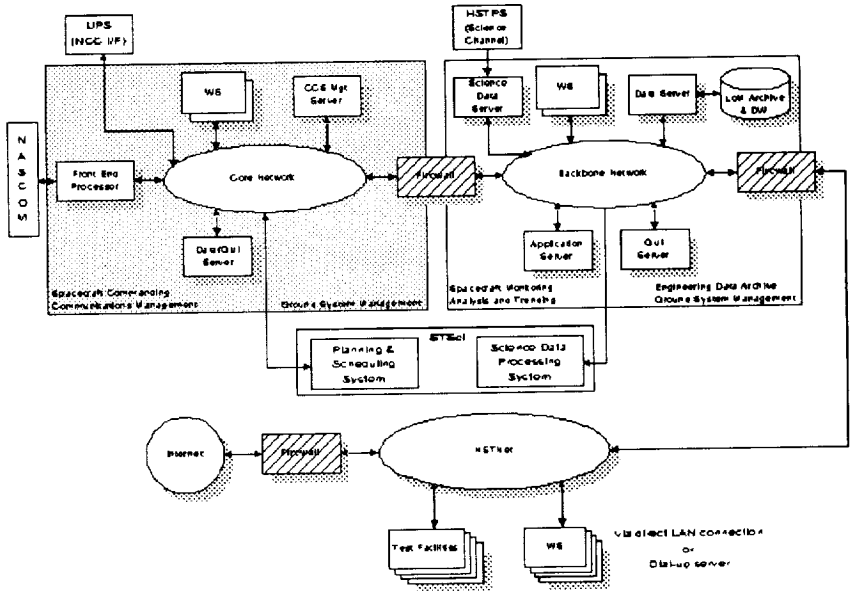

Fig. 2. CCS System Architecture

3.1 Graphical User Interface

The CCS GUI is implemented in the Java programming language. Java is a viable, highly productive, platform-independent software technology. The CCS vision to use Java applications running in web browsers was established as our GUI in 1995. All CCS tools and functions are available from a user's PC. To the user, the CCS is yet another tool that coexists with e-mail, web browsers, office applications, etc. The Control Center System software can be executed from any location that has an Internet connection (office, home, road, etc.). The original concept to run the Java applets under a web browser proved to be unrealistic considering the complexity of the applications. Separate virtual machines (VM) are now executed for each application to provide reliability and performance. Browsers do not support this capability and therefore the decision was made to run the applets as applications with a custom built class loader and packager. Netscape was abandoned as the primary browser due to numerous software problems and lack of support. Internet Explorer and the Microsoft VM have proved to be the best product for this COTS product selection.

3.2 Command Processing

The CCS spacecraft commanding is the most critical function of the ground system. The security requirements were very strict and the software is written specifically for HST. Typically a custom scripting language forms the core of these types of systems. For CCS, the Tool Command Language (TCL) was chosen as the core scripting language with custom extensions used to integrate with legacy applications. TCL has proved its worth in providing a modern interface while maintaining a backward compatible interface for existing scripts. A one time translation of the legacy scripts to TCL scripts was performed and validated in an automated test facility. The CCS Product Development Team also identified and integrated a database oriented GOTS product to manage the spacecraft event data and mission timeline; the Mission Operations Planning and Scheduling System (MOPSS) is developed and supported by many missions at the Goddard Space Flight Center (GSFC).

3.3 Front End Processing

The Front End Processor (FEP) is the communications interface for telemetry, command, and external communications management. A COTS hardware/software product, VEDA Omega series, processes the incoming telemetry data including removing communications protocol artifacts, time-tagging, de-commutation, and conversion of the raw data into engineering units. The FEP distributes the telemetry to the archive and publishes the telemetry for real-time GUI displays. A GOTS product is used for real-time telemetry data distribution; the Information Sharing Protocol (ISP) is developed by the Johnson Space Center. ISP servers exchange telemetry on a demand basis for changes-only data at one-half second intervals.

3.4 System Monitoring

The System Monitoring subsystem provides engineering data analysis tools for general plotting, trending, reporting, and analysis. The automated real-time monitoring component performs fault detection and isolation. COTS products have been used for spacecraft monitoring via an expert system, Rtworks, and for data analysis and plotting, PVWave. System Monitoring subsystem also provides the infrastructure to report and distribute CCS system level event messages. These event messages are used for analysis of spacecraft and ground system anomalies. The System Monitoring subsystem also provides spacecraft sensor analysis tools that perform attitude determination, sensor bias and calibration comparisons, and sensor data analysis algorithms. These functions are provided by legacy Fortran software that has been integrated into the CCS architecture. The CCS data warehouse will contain over 15TB of information that is fully indexed and saearchable.

3.5 Data Management

The Data Management subsystem provides data storage, cataloging, retrieval, and database services. The engineering data archive was designed with three major components: a short-term all-points online archive, a long-term offline all-points archive, and a changes-only online data warehouse. COTS products were selected for system database requirements, Oracle, and for data warehousing capability, Redbrick. Redbrick supports rapid complex query support of telemetry and spacecraft event data.

3.6 CCS Management

The CCS Management subsystem manages the overall ground system by providing system configuration, startup/shutdown control, monitoring, and failover capabilities. Several COTS products were integrated with a custom central server. The Control Center System monitors itself at the network, hardware, and software application levels to determine its state and to aid in the detection/resolution of problems. Custom software and COTS products such as Patrol are used to monitor hardware and software. A knowledge-based COTS product, Tivoli T/EC, is used to collect this hardware and software monitoring information. Tivoli has been configured to interpret the events and determine the appropriate action (e.g., restart a process, failover to another node, notify a system administrator, etc.).

4 COTS/GOTS Products

The CCS Product Development Team has integrated over 30 commercial-off-the-shelf/government-off-the-shelf (COTS/GOTS) products, developed over ½ million lines of custom code, maintained interfaces with legacy subsystems, and established a common Middleware protocol for subsystem communications. The new CCS exceeds the capabilities of the original system while significantly reducing the lines of custom code by more than 50%. CCS staff has delivered 10 major releases in 48 months with a total of 1½ million lines of code; each release has been deployed to approximately 4 operational multi-server strings, 25 operational test facility single-server strings, and 4 development multi-server strings. Onsite consultants were used in critical areas to assists teams in initial development efforts. Object-oriented design experts participated in all initial design reviews.

The CCS Product Development Team established and used an 80-20 rule as the basis for COTS/GOTS product selection. This rule states that if a COTS/GOTS product meets 80 percent of the total requirements for a function this product would be acceptable pending final approval from HST Project staff to defer implementation of the remaining requirements. This 80-20 rule approach reduced the amount of time and effort needed to perform COTS/GOTS trade studies. CCS development staff was trained with the selected COTS/GOTS products and technologies using

rapid just-in-time training sessions. The quick COTS/GOTS product selection process and rapid just-in-time COTS/GOTS training allowed the CCS Product Development Team to meet aggressive schedules.

The primary reason for the rejection of a COTS/GOTS product from consideration was mostly security related issues, particularly the ability to work through commercial firewall products. In some cases this was not possible, in other cases custom interfaces were developed to circumvent port issues at the firewall. Even today, few commercial products are designed with firewalls in mind.

The CCS operational COTS/GOTS software refers to the software that runs within the Control Center System (multi-server or single-server) on a daily basis. CCS administrative COTS software refers to the software that is used by system, security, network, or database administrators to keep CCS working but is not part of the day-to-day operational system. CCS development COTS software refers to the software used by developers to enhance the operational system software. See table 1 for a list of major operational COTS products in use by CCS. See table 2 for a list of major operational GOTS products in use by CCS. See table 3 for a list of major development and system administration products in use by CCS staff. In addition to the COTS/GOTS products listed below, CCS uses a large number of freeware, shareware, or low-cost COTS products including gnu-compilers, gnu-make, gnu-ftp, gnu-zip, tcl, tk, ntp, snmp, Netscape browser, Internet Explorer browser, Apache Server, Adobe Acrobat, and GhostScript.

Table 1. Operational COTS Software

Vendor	Product	Subsystem	Copies	Yr cost
SGI	Irix	operating system	many	$415k
Hewlett-Packard	HP-UX	operating system	site	$22k
Compaq	Open VMS	operating system	site	$13k
Sun	Solaris	operating system	25	$500
Roguewave	Rogue Wave	Middleware	4	$13k
Microsoft	Java VM	GUI	many	free
Microsoft	Java RTE	GUI	many	free
Veridian	ITAS	FEP	many	$112k
Visual Numerics	PVWave	System Monitoring	47	$100k
Talarian	RTWorks	System Monitoring	15	$40k
Acceler8 Technology	Transl8	System Monitoring	site	$14k
Oracle	Oracle, SQLplus	Data Management	site	$94k
Informix	Redbrick Warehouse	Data Management	4	$60k
ADIC	AMASS	Data Management	36	$24k
Checkpoint Software	Checkpoint/1	CCS Management	many	$75k
IBM	Tivoli/TEC	CCS Management	3	$75k
BMC	Patrol Agent	CCS Management	84	$14k
DataFellows	SSH	CCS Management	275	$11k

Telemon	Telalert	CCS Management	4	$2k

Table 2. Operational GOTS Software

Vendor	Product	Subsystem	Copies	Yr cost
GSFC	MOPSS	Command	6	1 FTE
JSC	ISP	FEP	many	free

Table 3. Development/Administrative COTS Software

Vendor	Product	Subsystem	Copies	Yr cost
Legato	Networker Client	System Admin	24	$14k
Bud Tools	Bud Tools	System Admin	1	free
McCabe	TruChange	Development	1	$33k
McCabe	Purify	Development	site	$13k
Mercury Interactive	WinRunner	Development	13	$9k

The majority of the operational COTS/GOTS products listed in table 1 and table 2 are critical to the success of Control Center System. Some COTS/GOTS products have been integrated with minimal effort while others have required substantial resources. Some COTS/GOTS products selected have met all of our requirements while others have met only a portion. A smaller number of COTS/GOTS products selected have required substantial resources to integrate or have required a reselection. Examples of products that meet our requirements and have worked well include Java, ISP, Rogue Wave, and Patrol. Examples of products that only meet a portion of our requirements and have required substantial resources to successfully integrate include TCL, Oracle, Redbrick, PVWave, and RTWorks. Other examples of products that have met the requirements but had not integrated well and eventually required a reselection include the Netscape browser and a suite of Tivoli tools that were never delivered by the vendor.

A recent review of the COTS/GOTS software products in use by CCS reported that many of the products in use are still quite adequate for operation of the system but that some products should be reevaluated. The review concluded that in many cases where better products are available today, the cost to procure and effort to modify the existing system architecture generally exceeds the benefits of these new products. There are some COTS products currently in use by CCS that are prime candidates for replacement based on an increasing lack of support and responsiveness from the vendor. Current COTS products that have been identified to be re-assessed include Tivoli/TEC, TruChange, Checkpoint/1, and Networker. Alternatives selections for these COTS products still need additional research and formal recommendations need to be made with respect to their replacement.

Java: The Java COTS product was selected in 1995. This product is successfully integrated within the Control Center System. The CCS GUI is written completely

using Java. The Java product would be very difficult to replace but the Java code itself is very easy to maintain.

ISP: The Information Sharing Protocol (ISP) GOTS product was select early in the CCS development phase and is also successfully integrated within the Control Center System. The ISP product is tightly coupled in the CCS architecture and replacing it with another product would be infeasible. JSC owns the ISP product but if needed we could maintain our own baseline; the source code for ISP is available and ISP code itself is easy to maintain. The ISP product meets all of the CCS requirements.

Rogue Wave: The Rogue Wave COTS product was selected early in the CCS development phase and is successfully integrated within the Control Center System. This product is used throughout the CCS architecture and replacing this product would be very difficult. Other COTS products are now available that would most-like be better selections if our decision were made today. The source code for Rogue Wave is available and is easy to maintain.

Oracle: The Oracle product was selected early in the CCS development phase and is successfully integrated within the Control Center System. The Oracle product required substantial resources to properly integrate this product with CCS applications. A large number of resources were used to write and maintain an Oracle Application Programmer Interface (API). In addition, the Oracle product has shown itself to be problematic over the last few years when running on the SGI platform. Technical support for this product has not been highly responsive to problems because of the SGI/Irix hosting. However, our recent COTS review concluded that it would not be cost-effective to replace the Oracle product with another product unless it's cost were to increase significantly in the near future.

Redbrick: The Redbrick product was selected to store spacecraft telemetry and orbital events data online for immediate retrieval. The Redbrick product required substantial resources to properly integrate this product within the CCS architecture. Training of development and system administration staff was expensive and keeping in-house experts is difficult. New hardware and software technologies will continue challenge the Redbrick product selection. Although the Redbrick product meets the requirements, the long-term sustaining costs may drive a re-assessment of this COTS selection.

PVWave: The PVWave product is a programmable data-visualization tool used to present spacecraft engineering data using plots and graphs. Other products of this type exist; however, PVWave has been sufficiently integrated into the CCS architecture as to make it very costly to replace. PVWave is a fine product that meets all of the CCS requirements.

RTWorks: The RTWorks product provides a rule-based environment for the specification and execution of spacecraft monitoring functions within CCS. There

are other products that can provide this functionality, but few are as tailored to support real-time processing. RTWorks is also a fine product that meets all of the CCS requirements. CCS only uses a small portion of the RTWorks product capabilities and the RTWorks capabilities could be rewritten using custom code. Although the RTWorks product meets the requirements, the long-term sustaining costs may drive a re-assessment of this COTS selection.

Netscape: The Netscape browser product was selected early in the CCS development phase to drive the CCS GUI. The Netscape browser product was problematic running on the SGI and Windows NT platforms and vendor was not highly responsive to problems. The Netscape browser is used in conjunction with CCS for standard web browsing function but was dropped for it's original purpose to run CCS. The stand-alone Java application replaced the Netscape browser product for executing the CCS application. Internet Explorer and the Microsoft VM have proved to be very reliable and efficient.

TruChange: The TruChange product provides configuration management functions for the CCS source code and related products. Over the last year, the vendor has become considerably less responsive to problems and issues regarding the product. Other comparable configuration management products exist that could be used by CCS staff. It has been recommended that a more detailed assessment be performed to determine the long-term cost effectiveness of migrating from TruChange to another product.

Checkpoint/1: The Checkpoint/1 product is used to provide security firewall functionality across all development, operational, and various test facility CCS configurations. Since the original selection of the Checkpoint/1 product, numerous other respectable firewalls have become available. In addition, over the last year, the vendor has become increasingly less responsive to requests for information. It has been recommended that alternate systems be reviewed to replace the existing firewalls.

Networker: CCS system administrators use the Networker product to backup the operational system data across the CCS strings. The product meets our current needs but occasionally has problems that require system administration staff support to recover. It has been recommended that a periodic review of similar products be assessed to determine if a more stable product can be found to replace the Networker product. Procedures for activating, transferring, and maintaining licenses are difficult and also support justification for product replacement.

5 Conclusion

NASA's Vision 2000 Project has successfully reengineered the Hubble Space Telescope ground-based command and control system. The new Control Center

System utilizes more than 30 COTS/GOTS products with an additional ½ million lines of custom glueware code. The CCS Product Development Team successfully incorporated new technology into the control center by integrated commercial-off-the-shelf products with legacy software. The success of the new Control Center System is due in part to the careful selection of COTS/GOTS products. The CCS Product Development Team was able to meet their aggressive schedules by developing successful processes to quickly select and provide training for COTS/GOTS products. Once selected, each COTS/GOTS product has an individual life cycle; some products have been successful from their initial selection, some products have been failures, and a larger number of products fall somewhere in between these two extremes. As a part of the CCS maintenance activities, the long-term success of the Control Center System will depend on periodic re-assessments and re-evaluations the COTS/GOTS product selections.

References

1. Barrett, L., Lehtonen, K.: Culture Management on the NASA Hubble Space Telescope Control Center Reengineering Project (1999)
2. Barrett, L., Speigel, D.: Vision 2000: Radical Reengineering of the Hubble Space Telescope Control Center System
3. Dougherty, A., Garvis, M., Whittier, W.: Re-engineering of the Hubble Space Telescope to Reduce Operational Costs
4. Friedman, B.: Deploying the Control Center System into Hubble Space Telescope Test Facilities
5. http://hubble.gsfc.nasa.gov